UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA - UEFS

Claudio Ari Bergossi Santos

Mapeamento dos processos de desenvolvimento ágeis em relação ao Modelo de Melhoria do Processo de Software do Brasil (nível G)

FEIRA DE SANTANA

2010

Claudio Ari Bergossi Santos

Mapeamento dos processos de desenvolvimento ágeis em relação ao Modelo de Melhoria do Processo de Software do Brasil (nível G)

Dissertação apresentada como requisito parcial à obtenção do grau de Bacharel em Engenharia de Computação pela Universidade Estadual de Feira de Santana

Orientador: Professor Mestre José Amâncio Macedo Santos

Feira de Santana

2010

Monografia de Final de Graduação sob o título *"MAPEAMENTO DOS PROCESSOS DE DESENVOLVIMENTO ÁGEIS EM RELAÇÃO AO MODELO DE MELHORIA DO PROCESSO DE SOFTWARE DO BRASIL (NÍVEL G)"*, defendida por Claudio Ari Bergossi Santos e aprovada em janeiro de 2010, em Feira de Santana - Ba, pela banca examinadora constituída pelos professores:

Prof. Msc. José Amancio Macedo Santos
Departamento de Tecnologia - DTEC/UEFS
Orientador

Prof. Msc. David Moises Barreto dos Santos
Departamento de Exatas - DEXA/UEFS
Examinador

Eliana Márcia Borges
Petrobrás - Bahia
Examinador

Dedico esta monografia a minha família,
pelo apoio fornecido,
aos meus amigos,
por tornarem os dias mais leves,
e para minha noiva, meu refúgio.

Agradecimentos

Primeiramente a DEUS por iluminar minha vida.

A minha família, pelo constante apoio e carinho que sempre dedicaram em qualquer ocasião. Em especial, minha mãe e minha vó, Simone Bergossi e Zulmira Bergossi respectivamente. A vocês sou muito grato por tudo. Sem vocês, não estaria neste momento escrevendo estes agradecimentos. Vocês me deram a vida, a dignidade e a força para continuar caminhando rumo ao sucesso profissional e principalmente o sucesso pessoal, obrigado.

A minha futura esposa Joyselles pelos momentos de carinho, alegria e constante apoio. Muito Obrigado.

Aos amigos na UEFS Danilo, Ângelo, Ademar, Luiz Henrique, Nara, Davi, Milton, Ronald, Ive, dentre outros..., assim como aos amigos/companheiros/irmãos de longa data João Ricardo, Alex, Daniel, Marinaldo e Nehru, por todos momentos de risadas, conversas, revoltas e cumplicidade que passamos juntos.

Aos demais colegas de UEFS, sem discriminar semestre, pelas contribuições na minha vida acadêmica.

Ao Professor Mestre José Amâncio pela grande apoio e auxílio prestado para a conclusão deste trabalho, além da sua amizade. Obrigado por tudo, principalmente pela paciência.

Um muito obrigado a todos!

Resumo

Empresas de desenvolvimento de *software* buscam uma maior agilidade no processos de desenvolvimento utilizando metodologias ágeis. Devido a competitividade do mercado brasileiro, as empresas estão buscando a adoção de modelos de qualidade de *software*, onde no Brasil, modelo de qualidade existente é o MPS.BR. Este trabalho insere-se no contexto descrito realizando uma analise sobre o MPS.BR e as metodologias ágeis, discutindo e identificando as relações entre estes dois temas, dando subsídio a empresas que utilizam, ou pretendem utilizar metodologias ágeis, a identificar o quão distante (ou não) estão do selo de certificação do nível G do MPS.BR.

Palavras-chave: Engenharia de *Software*. Metodologias Ágeis. *Extreme Programming*. *Scrum*. *Feature Driven Development*. MPS.BR.

Abstract

Business software development are seeking greater flexibility in the development process using agile methodologies. Due to competitive market, companies are seeking to adopt new software quality, which in Brazil, quality model is the existing MPS.BR. This work is described in the context performing an analysis on the MPS.BR and agile methodologies, discussing and identifying the relationships between these two subjects, giving subsidies to companies using or intending to use agile methodologies to identify how far (or not) are the seal of certification level of G MPS.BR.

Keywords: Software Engineer. Agile methodology. *Extreme Programming*. *Scrum*. *Feature Driven Development*. MPS.BR.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ABES	Associação Brasileira de Empresas de *Software*
ABNT	Associação Brasileira de Normas Técnicas
AP	Atributos de Processo
ASQC	*American Society for Quality Control*
CMM	*Capability Maturity Model*
CMMI	*Capability Mature Model Integration*
DSDM	*Dynamic Systems Development Method*
FDD	*Feature Driven Development*
FINEP	Financiadora de Estudos e Projetos
GPR	Gerência de Projetos
GRE	Gerência de Requisitos
IEC	*International Eletronichal Comission*
ISO	*International Standardization Organization*
LSD	*Lean Software Development*
MAs	Metodologias Ágeis
MR.MPS	Modelo de Referência do Mps.Br
MPS.BR	Melhoria de Processos do Software Brasileiro
OOPSLA	Object Oriented Programming, Systems, Languages and Applications
RUP	*Rational Unified Process*
SEI	*Software Engineering Institute*
SOFTEX	Sociedade Brasileira para Promoção da Exportação de *Software*
XP	*Extreme Program*

Sumário

Introdução

A Engenharia de *Software* desde a década de 50 tem evoluído (BOEHM, 2006), criando e discutindo conceitos importantes como divisão de *software*, arquitetura *top-down* e *botton-up*, diagramas, modelagens, entre tantos outros que resultou no estado atual. Com esses novos conceitos, o *software* deixou de ser uma caixa preta, para ser um conjunto de partes encaixáveis a fim de construir um sistema reutilizável, de fácil manutenção (SOMMERVILLE, 1995).

Apesar da sua evolução, dados de 1995 (GROUP, 1995) usando como base 8380 projetos, mostram que apenas 16,2% dos projetos foram entregues aos clientes respeitando os prazos e os custos e com todas as funcionalidades especificadas. Segundo dados desta mesma pesquisa, aproximadamente 31% dos projetos foram cancelados antes de estarem completos e 52,7% foram entregues, porém com prazos maiores, custos maiores ou com menos funcionalidades do que especificado no início do projeto. Dentre os projetos que não foram finalizados de acordo com os prazos e custos especificados, a média de atraso foi de 222%, e a média de custo foi de 189% a mais do que o previsto. Considerando todos os projetos que foram entregues, além do prazo pré-estabelecido e com custo maior do que o calculado inicial, na média, apenas 61% das funcionalidades originais foram incluídas.

Em meio a esta problemática, em 2001, profissionais relacionados à engenharia de *software* se reuniram para discutir formas de melhorar o desempenho de seus projetos (HIGHSMITH, 2001). Embora cada envolvido tivesse suas próprias práticas e teorias sobre como fazer um projeto de *software* ter sucesso, todos concordavam que em suas experiências prévias, um pequeno conjunto de princípios sempre parecia ter sido respeitado quando os projetos davam certo. A partir deste momento surge o *Manifesto Ágil* que possui como base esse conjunto de princípios.

O *Manifesto Ágil* vem se contrapor as metodologias tradicionais, que tipicamente são aplicadas em situações em que os requisitos do sistema são estáveis com evolução previsível. Entretanto, atualmente os projetos caracterizam-se por terem requisitos mutáveis, estando sujeitos a freqüentes mudanças resultantes da dinâmica da própria organização e do mercado. É neste tipo de projeto que as metodologias ágeis surgem como uma alternativa bastante adequada, por se tratarem de processos adaptativos.

É importante notar que o surgimento das metodologias ágeis não significa a rejeição dos princípios das metodologias mais robustas como *Rational Unified Process* (RUP) por exem-

plo, mas apresenta-se como alternativa para o desenvolvimento de *software*. As metodologias ágeis não apresentam conceitos novos, apenas alteram a perspectiva e os valores com que esses conceitos são encarados (HIGHSMITH; COCKBURN, 2001).

Atualmente pode-se notar uma gama de novas metodologias ágeis, dentre elas se destacam: *Scrum*, *Extreme Program* (XP), *Dynamic Systems Development Method* (DSDM), *Feature-Driven Development* (FDD), *Lean Software Development* (LSD), *Crystal*.

Apesar do Manifesto Ágil ter sido criado em 2001, as metodologias ágeis já apresentam resultados positivos (BECK; ANDRES, 2004). Como em qualquer nova idéia apresentam vantagens e fraquezas, ao considerar menos importantes conceitos considerados até então como boas práticas, já foi interpretada como um regresso a velhos métodos desorganizados de desenvolvimento de *software* (TELES, 2004), além de não levar em conta formalmente a análise de riscos e como estes acontecem. Isso pode ser considerado outra fraqueza. Por outro lado, a sua força resulta da capacidade de se adaptar a situações em que os clientes não estão seguros do que desejam e mudam de opinião freqüentemente. Outro ponto forte é a rapidez na entrega de funcionalidades, permitindo que estas sejam utilizadas o mais cedo possível.

Além das metodologias ágeis, outra linha de estudo que visa a melhorar o processo de desenvolvimento de *software*, são os modelos de qualidades. Devido à concorrência e dinamicidade do mercado de *software*, os modelos de qualidades estão cada vez mais sendo almejados pelas empresas. Segundo dados da Associação Brasileira de Empresas de *Software* (ABES), o número de empresas de desenvolvimento de *software* aumentou 15,5% de 2004 à 2008 (SOFTWARE, 2009). Neste contexto, o selo de certificação pode tornar-se um fator diferencial para um mercado competitivo.

Na área de *software* estes modelos surgiram no início dos anos 90, quando o *Software Engineering Institute* (SEI) lançou a versão 1.0 do *Capability Maturity Model* (CMM). Este mesmo modelo evoluiu em conjunto com as novas tecnologias surgidas na computação, e se adequou as necessidades criadas por esta evolução. Assim surgiu o *Capability Mature Model Integration* (CMMI) que se tornou um modelo de qualidade de *software* respeitado mundialmente.

No Brasil foi criado o projeto intitulado: Melhoria de Processo do *Software* Brasileiro (MPS.BR) pela SOFTEX, voltado para a realidade do mercado de micro, pequenas e médias empresas de desenvolvimento de *software*, sendo baseado no CMMI e nas normas ISO/IEC 12207 e ISO/IEC 15504 (SOFTWARE; SINGH, 1989). O MPS.BR tem como objetivo a disseminação da qualidade visando a competitividade da indústria brasileira de *software*, nos mercados interno e externo, através da melhoria e avaliação de processos e produtos de *software*, a um custo acessível às empresas de menor porte (SOFTEX, 2009).

Um dado relevante para este trabalho, é o número de empresas brasileiras que adotam o modelo de qualidade do MPS.BR, ver tabela 1:

Tabela 1: Avaliações MPS.BR publicadas até 05/2009 (SOFTEX, 2009)

Ano	A	B	C	D	E	F	G	Totais por Ano
2005	0	0	0	0	1	3	1	5
2006	2	0	0	1	1	1	7	12
2007	1	0	0	0	1	12	41	55
Total 2005 à 2007	3	0	0	1	3	16	49	72
2008	1	0	0	0	1	9	40	51
2009	0	0	0	0	1	12	15	28
2010	0	0	0	0	0	0	0	0
Total 2008 à 2010	1	0	0	0	2	21	55	79
TOTAIS	4	0	0	1	5	37	104	151

Dados da Associação Brasileira de Empresas de *Software* (ABES), informam através de pesquisas que no Brasil, até o final de 2008, existiam 2079 empresas de desenvolvimento de *software* (SOFTWARE, 2009). Onde dessas 2079 empresas, apenas 151 empresas possuem certificação MPS.BR e que o maior número destas certificações estão concentradas no nível G (nível inicial), ver tabela 1. Através desta tabela [1], pode-se perceber um crescimento anual significativo de certificações, evidenciando a busca de qualidade seus produtos e serviços por parte das empresas de desenvolvimento de *software*.

Em função da importância dos dois temas (modelos de qualidade e metodologias ágeis), pesquisas estão sendo realizadas com o objetivo de verificar o relacionamento entre eles. Há trabalhos mapeando práticas do SCRUM em relação ao CMMI ((MARCAL; SOARES; BELCHIOR, 2007) e (SIQUEIRA, 2007)), onde através destes trabalhos, pôde-se verificar elementos equivalentes entre processo e o modelo de qualidade. Há publicações também relacionando XP (SANTANA; TIMÓTEO; VASCONCELOS, 2006) e FDD ((CAMARGO, 2007) e (LAURINDO, 2007)) com o MPS.BR em seus níveis iniciais.

Mas os resultados expostos em alguns trabalhos que relacionam as MAs com o MPS.BR, possuem problemas e/ou são incompletos, na análise de equivalência entre o modelo de referência do MPS.BR e as MAs. Não há detalhes sobre como as metodologias contemplam ou não um determinado requisito. No caso dos trabalhos encontrados sobre o mapeamento do XP com nível G do MPS.BR, o autor simplesmente informa quais requisitos são contemplados, e apenas os requisitos não contemplados são discutidos a nível de sugestões para a sua adaptação a fim de atingir o resultado esperado pelo requisito do modelo de qualidade. Já os

[1] A descrição dos níveis de A à G são apresentados na figura 4 da seção 1.3

trabalhos relacionando FDD e MPS.BR concluíram que todos os requisitos do nível G são contemplados por esta metodologia. Uma possível interpretação é que os autores realizaram suas análises baseados no título de cada requisito e não nos resultados esperados presente do guia de implementação do MPS.BR.

Desta forma, este trabalho tem como objetivo central mapear as práticas dos processos ágeis XP, SCRUM e FDD, em relação ao nível G do MPS.BR, expondo a forma com as MAs se relacionam com o nível G do MPS.BR. O principal diferencial é que, no mapeamento, trechos pertinentes do guia de implementação sobre os resultados esperados para cada requisito são destacados e os argumentos conclusivos são apresentados. Vale ressaltar que o mapeamento será baseado na teoria das metodologias ágeis. Não é do escopo deste trabalho analisar customizações que ocasionalmente podem ocorrer na utilização das MAs.

1 Fundamentação Teórica

1.1 Manifesto Ágil

Durante a evolução dos processos de Engenharia de *Software*, a indústria se baseou nos métodos tradicionais de desenvolvimento de *software*, que definiram, por muitos anos, os padrões para criação de *software* nos meios acadêmico e empresarial (TELES, 2004). Porém, percebendo que a indústria apresentava um grande número de casos de fracasso, alguns líderes experientes adotaram modos de trabalho que se opunham aos principais conceitos das metodologias tradicionais. Aos poucos, foram percebendo que suas formas de trabalho, apesar de não seguirem os padrões no mercado, eram bastante eficientes. Aplicando-as em vários projetos, elas foram aprimoradas e, em alguns casos, chegaram a se transformar em novas metodologias de desenvolvimento de *software* (HIGHSMITH, 2001). Essas metodologias passaram a ser chamadas de leves por não utilizarem as formalidades que caracterizavam os processos tradicionais e por evitarem a burocracia imposta pela utilização excessiva de documentos. Com o tempo, algumas delas ganharam destaque nos ambientes empresarial e acadêmico, gerando grandes debates, principalmente relacionados à confiabilidade dos processos e à qualidade do *software*.

Depois de muitos anos de experiência, 17 líderes que trabalhavam no contra-fluxo dos padrões da indústria de *software* perceberam que seus modos de trabalho, além de eficazes, eram parecidos uns com os outros. Em 2001, eles se reuniram durante um final de semana em Utah para discutir sobre suas formas de trabalho e para chegar a uma nova metodologia de produção de *software* que pudesse ser usada por todos eles e em outras empresas, substituindo os modelos tradicionais de desenvolvimento (HIGHSMITH, 2001).

Após dois dias de debate, o grupo não chegou a uma metodologia: concluíram que desenvolver *software* é algo complexo demais para ser definido por um único processo, pois ele depende de muitas variáveis e, principalmente, por ser uma tarefa realizada por pessoas em praticamente todas as etapas do processo. No entanto, o grupo chegou ao consenso de que alguns princípios eram determinantes para a obtenção de bons resultados. O resultado deste encontro foi a identificação de 12 princípios e a publicação do Manifesto Ágil (MANIFESTO,

2001) que é representado com quatro premissas:

- Indivíduos e iterações são mais importantes do que processos e ferramentas;
- *Software* funcionando é mais importante do que documentação completa;
- Colaboração com o cliente é mais importante do que negociação de contratos;
- Adaptação a mudanças é mais importante do que seguir o plano inicial.

Os participantes do Manifesto Ágil eram: *Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Roland Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland e Dave Thomas* (MANIFESTO, 2001).

O Manifesto Ágil ressalta o que mais tem valor para as metodologias ágeis. Processos, contratos, documentação e planejamento têm valor para o desenvolvimento de *software*, mas são menos importantes do que saber como lidar com pessoas, do que ter o cliente colaborando para encontrar a melhor solução, do que entregar o *software* com qualidade e do que se adaptar às mudanças (TELES, 2004)

As metodologias, então chamadas de ágeis, propõem a obtenção de resultados práticos em um período menor do que a indústria de *software* estava acostumada, tirando o foco do processo e o colocando no produto. Para isso, foi preciso que os métodos ágeis dispensassem ou modificassem as etapas do processo e a forma como os envolvidos com o desenvolvimento realizavam suas atividades.

Muitas dessas mudanças alteraram características tidas como essenciais pelos métodos tradicionais, por isso as abordagens ágeis tornaram-se polêmicas e não inspiraram confiança nos mais conservadores. Por outro lado, o foco dos modelos ágeis no produto final e nos interesses de negócio tem despertado interesse de outra parcela da comunidade de desenvolvimento de *software*.

Desde sua publicação, até novembro de 2007, o Manifesto Ágil teve 7.378 pessoas que o assinaram publicamente em http://www.agilemanifesto.org. Além disso, o número de adeptos de metodologias ágeis vem crescendo pelo mundo. A *Scrum Alliance* possui 18.225 associados que pagaram para obter certificações em desenvolvimento e treinamento usando Scrum (ALLIANCE, 2009).

1.2 Metodologias Ágeis

1.2.1 *Extreme Programming* (XP)

Kent Beck criou XP depois de vários anos de atuação na indústria de desenvolvimento de *software*. Publicou em 1997 um livro com suas melhores técnicas de programação (BECK, 1997). No mesmo ano, foi convidado para conduzir um projeto crítico na *Chrysler*: um sistema para a folha de pagamento que já havia estourado os custos e os prazos sem alcançar resultados substanciais. Com *Martin Fowler* e *Ron Jeffries* na equipe, *Kent Beck* conseguiu tornar a equipe altamente produtiva e entregar um sistema de excelente qualidade começando do zero e terminando o projeto em menos tempo do que havia sido gasto nas tentativas anteriores.

No projeto da *Chrysler*, *Kent Beck* selecionou um conjunto de práticas que haviam se mostrado eficientes separadamente em outros projetos e as aplicou juntas (ANDERSON *et al.*, 1998). *Beck* percebeu que revisão de código, testes, integração rápida, *feedback* do cliente, *design* simples, entre outras práticas, eram atividades que aumentavam a qualidade do produto. Sua proposta foi intensificar a utilização delas ao extremo, fazendo, por exemplo, revisão constante do código através de programação em pares, intensificando o uso de testes com testes automatizados, antecipando os testes e permitindo um acompanhamento constante do projeto com o cliente presente.

Em 1999 *Kent Beck* publicou "*Extreme Programming Explained: Embrace Change*" (BECK, 1999), o livro que introduziu e tornou a metodologia internacionalmente conhecida. Na conferência OOPSLA do anos seguintes, o XP ganhou grande repercussão por trazer idéias opostas aos paradigmas vigentes de desenvolvimento de *software*. Em 2004 *Kent Beck*, junto com sua esposa *Cynthia Andres*, lançaram a segunda edição do livro (BECK; ANDRES, 2004), tornando XP mais abrangente e flexível através do enfoque nos valores da metodologia.

1.2.1.1 Valores

A primeira edição de seu livro sobre XP, *Kent Beck* abordou quatro valores que, em alinhamento com o Manifesto Ágil, definiram as prioridades da metodologia. Na segunda edição, o valor do "Respeito" foi incluído para ressaltar o lado humano presente no desenvolvimento de *software*. Segundo *Beck* e *Andres* (2004) os cinco valores são:

- **Comunicação** é fundamental para a obtenção de um produto de qualidade que atenda às necessidades do cliente. XP favorece a comunicação entre todos os membros da

equipe através de atividades colaborativas. Quanto mais intensificada for a comunicação, mais facilmente serão aproveitadas as contribuições de cada membro e tarefas individualizadas poderão contar com a colaboração de outros participantes, fazendo com que críticas tornem-se contribuições que melhoram as soluções antes mesmo de serem implementadas;

- **Simplicidade** nas soluções para evitar esforços desnecessários. É mais válido gastar pouco esforço para produzir uma solução simples e depois sofisticá-las do que criar de imediato implementações complexas com funcionalidades extras. Funcionalidades e complexidade desnecessárias aumentam a complicação do código e o tornam mais propenso à introdução de erros, exigindo mais horas de programação e mais refatorações. Tudo isso dificulta o crescimento e a manutenção da arquitetura e do código, desviando o projeto de seus principais objetivos e atrasando a entrega das funcionalidades essenciais. Os princípios do benefício mútuo, melhoria e passos pequenos promovem a simplicidade durante o desenvolvimento;

- **Coragem** é importante para mudar, inovar e aceitar que não se sabe de tudo. À medida que o conhecimento sobre o projeto aumenta, muitas decisões precisam ser revistas e o preço a ser pago é abrir mão de planos e de trabalho feitos no passado para chegar a soluções adequadas à realidade do projeto. Para motivar o surgimento de coragem junto com responsabilidade, os princípios da humanidade e da aceitação da responsabilidade valorizam as pessoas e aumentam as suas auto-confianças;

- ***Feedback*** constante entre todos os envolvidos permite que a equipe e o projeto identifiquem seus problemas e se adaptem a eles (TELES, 2004). Quanto mais cedo impedimentos forem encontrados e removidos, menos tempo eles irão atrapalhar. Quanto mais freqüente forem as avaliações, do produto e do processo de desenvolvimento, mais rapidamente serão identificados os problemas e as soluções;

- **Respeito** entre as pessoas é a peça principal para que os demais valores corroborem (BECK; ANDRES, 2004). Sem respeito, a Comunicação e o *Feedback* serão pouco eficientes e a Coragem de um membro poderá ser nociva aos demais por não estar alinhada com os interesses da equipe. Todos os participantes devem manter o respeito entre si e em relação aos seus trabalhos. Desvalorizar alguém, a função que exerce ou a qualidade de seu trabalho são formas de falta de respeito que minam a sinergia da equipe. Uma forma de respeito de cada um para com a equipe é assumir responsabilidades que serão capazes de cumprir e da equipe para com seus membros é confiar que cada um fará o

melhor trabalho possível. XP considera que os desenvolvedores também são pessoas e preocupa-se com suas necessidades pessoais e profissionais.

1.2.1.2 Papéis

Uma equipe de XP deve reunir todas as habilidades técnicas e de negócios para produzir o *software*. A hierarquia entre os desenvolvedores deve ser rasa e sem necessariamente uma divisão preestabelecida de tarefas. Em princípio, as responsabilidades são atribuídas de acordo com especialidades, ao longo do tempo, espera-se que essas especialidades sejam disseminadas entre os envolvidos para evitar a concentração de conhecimento e estimular o crescimento profissional de todos os participantes.

Kent Beck descreve papéis importantes dentro de uma equipe, porém nem todos precisam coexistir, ou necessariamente precisam ser atribuídos a pessoas diferentes (BECK; ANDRES, 2004). Uma equipe **completa** de XP prevê papéis para **programadores** responsáveis por estimar, implementar e testar; **analistas de negócios** que auxiliam o cliente a definir histórias; **analistas de testes** que identificam cenários de teste; **projetistas de interação** que avaliam a utilização do sistema pelos usuários; **arquitetos** que analisam a estrutura do sistema para identificar melhorias de grande impacto; **gerentes de projeto** que promovem a comunicação e removem os impedimentos que dificultam o trabalho dos programadores; **gerentes de produto** que escrevem e priorizam histórias e definem os objetivos dos ciclos do projeto; **redatores técnicos** que produzem a documentação para os usuários finais; **executivos** que preocupam-se com objetivos de alto nível; **recursos humanos** que resolvem problemas burocráticos; e, **usuários** que contribuem com opiniões, sugestões e críticas para ajustes e novas histórias.

1.2.1.3 Fases

Um projeto com XP começa com a fase de **Exploração** que, dependendo da complexidade e das restrições do projeto, pode consumir desde poucos dias até algumas semanas. Os programadores começam o projeto tendo contato direto com o cliente para conhecer suas necessidades e intenções. À medida que a equipe de desenvolvimento adquire conhecimento sobre o produto, ela pesquisa e discute as tecnologias, arcabouços e ferramentas capazes de prover a solução mais adequada considerando as funcionalidades e os critérios de desempenho, confiabilidade, segurança, portabilidade, escalabilidade, manutenção, custo e tempo indicados pelo cliente. A fase de exploração é marcada por muita pesquisa e leitura. Para auxiliar na tomada das decisões, a equipe de desenvolvimento pode produzir pequenas aplicações, que **Kent Beck** chama de “spikes” (BECK; ANDRES, 2004), para avaliar com mais precisão as tecnologias em

questão e assim comparar seus resultados. Essas aplicações são feitas sem a pretensão de que possam fazer parte do produto final. Elas são descartadas quando a implementação do *software* se inicia, no entanto serviram para familiarizar a equipe com tecnologias novas ou desconhecidas.

Para fazer o **Planejamento**, a equipe de negócios descreve as funcionalidades que precisa nos cartões de estória, acompanhada pela equipe de desenvolvimento, que pode tirar dúvidas e sugerir adaptações ou melhorias com base em seu conhecimento técnico. Com tecnologias e ferramentas escolhidas, os programadores estimam o volume de trabalho de cada funcionalidade e anotam no próprio cartão de história.

Todo o processo de escrita de cartões, priorização e estimativas faz parte do "Jogo do Planejamento" (BECK; FOWLER, 2000). Através desses elementos a equipe fará o planejamento das versões (*releases*) do produto agrupando os cartões de acordo com funcionalidades que fazem sentido serem entregues juntas para os usuários. Em seguida, a equipe faz o planejamento das iterações, dividindo os cartões da primeira versão em grupos menores: as iterações.

O foco do planejamento das versões é orientar as expectativas de negócio em relação à obtenção de versões que entrem em produção. Conforme a estratégia da equipe de negócios, duas abordagens podem ser usadas para defini-las:

- Se existe um conjunto mínimo de funcionalidades estritamente necessário para compor a versão, o escopo da versão está fixado e as estimativas desses cartões podem ser usadas para calcular o tempo de implementação;
- Se fatores ou eventos externos ao projeto definem uma data, o tempo para entrega da versão está prefixado. Neste caso, a partir das prioridades mais altas, as estimativas são usadas para escolher o escopo, isto é, quais funcionalidades serão implementadas no período determinado.

O planejamento das iterações estabelece pontos de controle dentro da versão para programadores e clientes saibam como a execução está em relação ao planejado. No planejamento da iteração, os programadores analisam novamente cada cartão, desta vez para identificar as tarefas de implementação associadas a ele e para se comprometerem individualmente com a implementação. Opcionalmente cada desenvolvedor pode fazer estimativas das tarefas que identificou em cada cartão. A fase de Implementação inicia com os programadores focados em concluir a iteração. Durante a implementação, a equipe realiza as práticas descritas na seção seguinte. No fim da iteração, os clientes aprovam a implementação dos cartões ou não, através

de testes de aceitação. O planejamento da próxima iteração começa considerando o planejamento feito para a versão e verifica se há mudanças consideráveis nas funcionalidades ou se novos fatores impactam na velocidade de desenvolvimento.

Este ciclo pode ser visualizado através da figura 1.

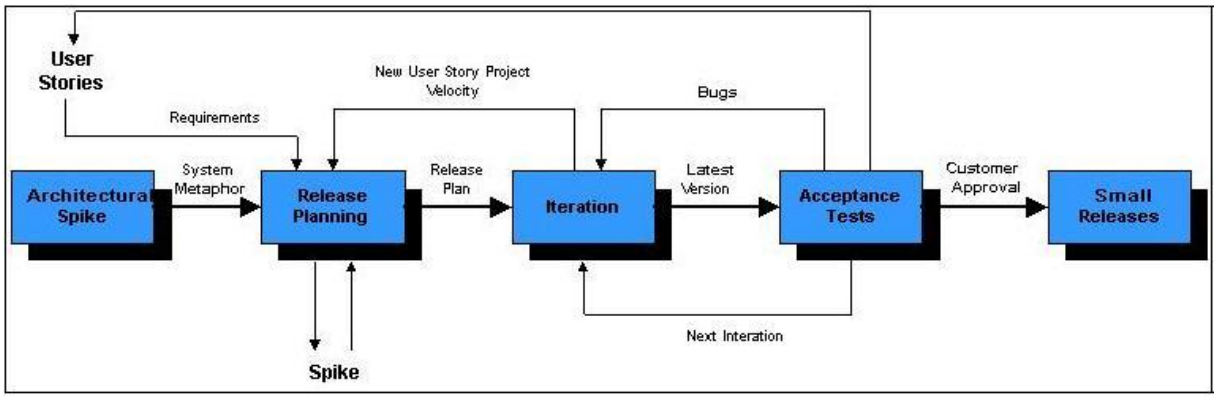

Figura 1: Ciclo de desenvolvimento do XP (WELLS, 1999)

1.2.1.4 Práticas

Na primeira versão do livro de *Kent Beck* (BECK, 1999) foram propostas 12 práticas que traduzem os valores em ações para serem executadas no dia-a-dia do trabalho. São elas:

1. **Versões Pequenas**: A equipe deve implementar e entregar freqüentemente pequenas partes de *software* funcionando. O tamanho da iteração deve ser constante durante o projeto e de, no máximo, três semanas cada uma, para que o cliente forneça *feedback* constante e a equipe consiga ter uma visão precisa do andamento do projeto sem surpresas no final;

2. **Jogo do Planejamento**: No começo do projeto, os clientes escrevem cartões com estórias que descrevem os requisitos do sistema. Programadores e clientes colaboram para selecionar o subconjunto de cartões com maior valor agregado: os programadores fornecem estimativas para cada história enquanto os clientes definem os seus valores de negócio. O planejamento acontece em dois níveis: no começo da implementação de cada *release* e no começo de cada iteração. Uma release é composta de várias iterações. No final de cada iteração um subconjunto das estórias da release devem ter sido implementadas. A decisão feita neste ponto é apenas um plano inicial: mudanças são bem vindas e podem ser incorporadas ao plano durante o projeto.

3. ***Design* Simples**: Simplicidade é o conceito chave que permite que um sistema adapte-se a mudanças. Para minimizar o custo das mudanças, deve ser implementado o *design* mais simples, com o nível de complexidade e flexibilidade necessários para atender às necessidades do momento. Como o desenvolvimento é incremental, os programadores não devem tentar antecipar requisitos, pois serão adicionados mais tarde se forem realmente necessários e o *design* será naturalmente melhorado através de refatorações;

4. **Programação em Pares**: Desenvolvedores trabalham em pares durante as tarefas de implementação para promover o trabalho coletivo e colaborativo. As duplas são trocadas frequentemente para que todos os participantes tenham a possibilidade de interagir, aumentando a comunicação e a união do grupo. Com duas pessoas concentradas na mesma tarefa, a inserção de muitos erros é evitada e quando existem, eles são encontrados mais rapidamente. Enquanto um dos programadores digita, o outro revisa o código e sugere melhorias. Com o tempo, o trabalho em duplas promove a replicação de conhecimentos específicos de cada participante para vários membros da equipe. A seleção dos pares geralmente depende da tarefa, da disponibilidade dos programadores e da experiência de cada um;

5. **Testes**: O *software* é constantemente exercitado por um conjunto de testes automatizados escritos por programadores e clientes. A equipe de programadores escreve testes de unidade para todos os componentes do sistema e os executa várias vezes ao dia para assegurar que funcionalidades adicionadas recentemente não tenham introduzido erros no código antigo. Os clientes escrevem testes de aceitação para assegurar que o sistema faça exatamente o que eles solicitaram. Estes testes são executados sempre que uma nova funcionalidade é implementada para determinar se o cartão de estória realmente está concluído;

6. **Refatorações**: São um conjunto de técnicas sistemáticas para reorganizar o código, alterando sua estrutura interna sem modificar o comportamento externo (FOWLER, 2004). As refatorações podem variar de simples mudanças de nomes de variáveis, métodos ou classes para tornar o código mais inteligível, passando pela remoção de código duplicado, até a simplificação da arquitetura do sistema (KERIEVSKY, 2004);

7. **Integração Contínua**: O código fonte deve ser mantido em um repositório comum a toda a equipe de forma que sempre que alguma tarefa é concluída, o novo código possa ser executado, testado, e, se correto integrado no repositório. Os programadores incluem e recuperam código do repositório várias vezes ao dia, assim todos os membros da equipe mantêm-se sincronizados, trabalhando com a versão mais recente do código;

8. **Propriedade Coletiva do Código**: Não existe o conceito de propriedade exclusiva ou restrita do código. O código do repositório é a última versão estável do sistema e é propriedade de toda a equipe. Se uma dupla de programadores precisar modificar um trecho do código ou identificar oportunidades para refatorá-lo, eles podem, e devem fazê-lo sem pedir permissão a quem o escreveu originalmente. Esta dupla também deve verificar a validade das mudanças escrevendo novos testes e fazendo com que todos os testes passem, inclusive os que já existiam, para depois criar uma nova versão estável do código;

9. **Ritmo Sustentável**: O ritmo de trabalho não deve afetar a saúde ou a vida pessoal dos participantes. Durante o planejamento, a quantidade de horas de trabalho dedicadas ao projeto deve ser definida realisticamente. É aceitável que a equipe trabalhe horas extras em situações raras. Uma extensiva e recorrente sobrecarga de trabalho irá reduzir a qualidade do código e trará perdas no médio e longo prazo;

10. **Cliente Presente**: A equipe deve ser composta por diferentes pessoas com amplo conhecimento e experiência de forma a preencher todas as habilidades necessárias para o projeto. Isto deve incluir pessoal da área de negócios, chamados de clientes, que entendem as necessidades dos usuários e possuem conhecimento sobre as regras de negócio do sistema. O cliente escreve estórias, define prioridades, responde dúvidas dos programadores e acompanha a implantação, se possível, diariamente;

11. **Metáfora**: Os participantes do projeto devem encontrar uma linguagem comum para falar sobre o sistema, relacionando suas abstrações com elementos do mundo real. Esta linguagem deve ser igualmente entendida por pessoas da área técnica e por pessoas da área de negócios para facilitar a comunicação entre eles. Esta provavelmente é a prática mais difícil de introduzir em uma equipe inexperiente porque ela está diretamente relacionada à comunicação e a quão compreensíveis as pessoas serão ao compartilharem seus desejos, idéias e conhecimentos;

12. **Padrões de Código**: Antes de começar a programação, os programadores devem definir um conjunto de padrões para ser usado na escrita do código. Isso torna o código mais fácil de entender, melhora a comunicação, facilita refatorações e contribui para a **Propriedade Coletiva**.

1.2.2 *Scrum*

Em um jogo de *rugby*, *scrum* é uma forma do time recolocar a bola em jogo. Na metodologia criada por *Jeff Sutherland* e formalizada em 1995 por *Ken Schwaber* (SCHWABER, 1995), a

equipe de desenvolvimento trabalha unida com o objetivo de entregar o *software* funcional de alta qualidade. Neste modelo, a equipe se compromete com um objetivo e tem autonomia para definir a tática para chegar até ele.

Em modelos tradicionais, o desenvolvimento de *software* é visto como um processo definido. O que significa que, a partir de entradas, é possível produzir uma determinada saída e que isto pode ser repetido várias vezes sem diferenças expressivas. Porém, *Ken Schwaber* percebeu que a complexidade da produção de sistemas tornava essa abordagem inadequada ao problema em questão e que melhores resultados poderiam ser obtidos tratando o desenvolvimento de *software* como um processo empírico (SCHWABER, 1995). A dificuldade de definir um modelo de produção de *software* capaz de suportar as inúmeras variações dos ambientes de desenvolvimento motivou *Schwaber* a unir-se a *Jeff Sutherland e Mike Beedle* para compilarem suas idéias e lapidar o Scrum como modelo ágil de desenvolvimento de *software* que agrupa um pequeno conjunto de simples regras gerenciais que dão liberdade à evolução do processo de desenvolvimento (SCHWABER; BEEDLE, 2001).

Scrum atua principalmente na gerência do projeto, sem determinar como a equipe executará as tarefas de programação. Esta abordagem favorece a auto-organização da equipe e permite a integração com outras metodologias ágeis que foquem nas práticas de programação, como por exemplo, XP e FDD.

1.2.2.1 Artefatos do *Scrum*

Os elementos que a equipe produz para seguir a práticas de *Scrum* são cartões com as funcionalidades e gráficos de acompanhamento. Os cartões agrupados formam o *Product Backlog* e outros *backlogs*. Os gráficos são atualizados frequentemente e devem refletir o estado do projeto.

Os artefatos utilizados pelo Scrum (SCHWABER; BEEDLE, 2001) são:

- *Product Backlog*: É basicamente uma lista de requisitos, estórias, coisas que o cliente deseja, descritas utilizando a terminologia do cliente, dentre outros artefatos que são pertinentes ao projeto. Todos os requisitos do *Product Backlog*, é discutido por toda a equipe, mas a ordenação do mesmo por prioridade de implementação é responsabilidade do próprio *product owner* (cliente);
- *Selected Backlog*: Um subconjunto de funcionalidades que o cliente escolheu a partir do *Product Backlog* para ser implementado no *sprint* atual e que não pode ser modificado durante o *sprint*;

- *Sprint Backlog*: Lista priorizada, obtida a partir da quebra dos itens do *selected backlog* em tarefas menores;
- *Impediment Backlog*: Lista dos obstáculos identificados pela equipe que não pertencem ao contexto do desenvolvimento;
- *Burndown Chart*: Este gráfico é utilizado para a visualização do progresso do *sprint*. Ele mostra o número de horas estimadas pelo número de dias para um *sprint*. Através de suas informações o *Scrum Master* pode analisar o andamento e tomar algumas medidas caso algo esteja saindo do planejado.

1.2.2.2 Papéis

Poucos papéis caracterizam o *Scrum*. Apenas três estão presentes na metodologia (SCHWABER; BEEDLE, 2001):

***Product Owner* (cliente)**: O cliente deve possuir a visão do produto em vários níveis. A visão de longo prazo é mantida com o gerenciamento do *backlog* e a de curto prazo através da definição do objetivo do *Sprint*. Quando existe vários interessados ou envolvidos com o produto, o dono do produto deve ser representado por uma só pessoa que entenda todas as necessidades e seja capaz de priorizá-las.

Scrum Team : A equipe de desenvolvimento é vista como uma unidade dentro da metodologia. Ela deve ser multi-funcional e auto-suficiente, o que significa que seus membros devem reunir todas as habilidades necessárias para atingir o objetivo, assumindo compromissos e responsabilidades em relação ao projeto. Não existe hierarquia, cada um deve possuir consciência de sua responsabilidade para com a meta e pessoas sem comprometimento devem deixar a equipe.

Scrum Master: O *Scrum Master* deve possuir conhecimento de todo o processo para garantir que ele seja seguido e deve manter uma ampla visão sobre o projeto. Assim, pode auxiliar o cliente a tomar decisões que conduzam o projeto a obter o máximo retorno e aconselhar a equipe com suas escolhas. É responsabilidade do *Scrum Master* eliminar os itens do *impediment backlog*, fazer gráficos de acompanhamento da evolução do projeto e proteger a equipe contra instabilidades e fatores externos que prejudiquem seu rendimento.

1.2.2.3 Fases

Scrum caracteriza-se como um processo empírico e adaptativo. As fases do *Scrum*: planejamento, *Sprint* e avaliação, correspondem, respectivamente, aos três primeiros estados do ciclo

evolutivo do processo. A etapa de ação acontece no *Scrum* com mudanças no próximo planejamento baseadas nas conclusões da última avaliação.

O primeiro planejamento pode durar desde dias até algumas semanas, o tempo necessário para a equipe conhecer satisfatoriamente as características do produto que será desenvolvido e escolher as tecnologias e ferramentas de trabalho. Os planejamentos seguintes têm duração de aproximadamente dois dias, nos quais três reuniões são realizadas: uma para estimar, outra para definir a meta do *sprint* e a terceira para definir as tarefas necessárias para atingir a meta. A cada *sprint*, o cliente se baseia nas prioridades de negócio para escolher uma parte do *backlog* para compor o *selected backlog*. A equipe pode concordar com a seleção ou negociar os itens até que se comprometa a produzi-los durante o *sprint*.

Na reunião de estimativa, a equipe se reúne para estimar a quantidade de trabalho dos itens mais prioritários do *product backlog* com a presença do *Scrum Master* e do cliente para esclarecer dúvidas referentes às regras de negócio. Depois, na primeira reunião de planejamento do sprint (figura 2, Sprint Planning 1), todos se reúnem para definir a meta do *sprint* e criar o *selected backlog* de acordo com as prioridades de negócio estabelecidas pelo dono do produto. Em seguida, na segunda reunião de planejamento do *sprint* (figura 2, *Sprint Planning* 2), a equipe se reúne para quebrar os itens do *selected backlog* em tarefas de implementação de menor granularidade.

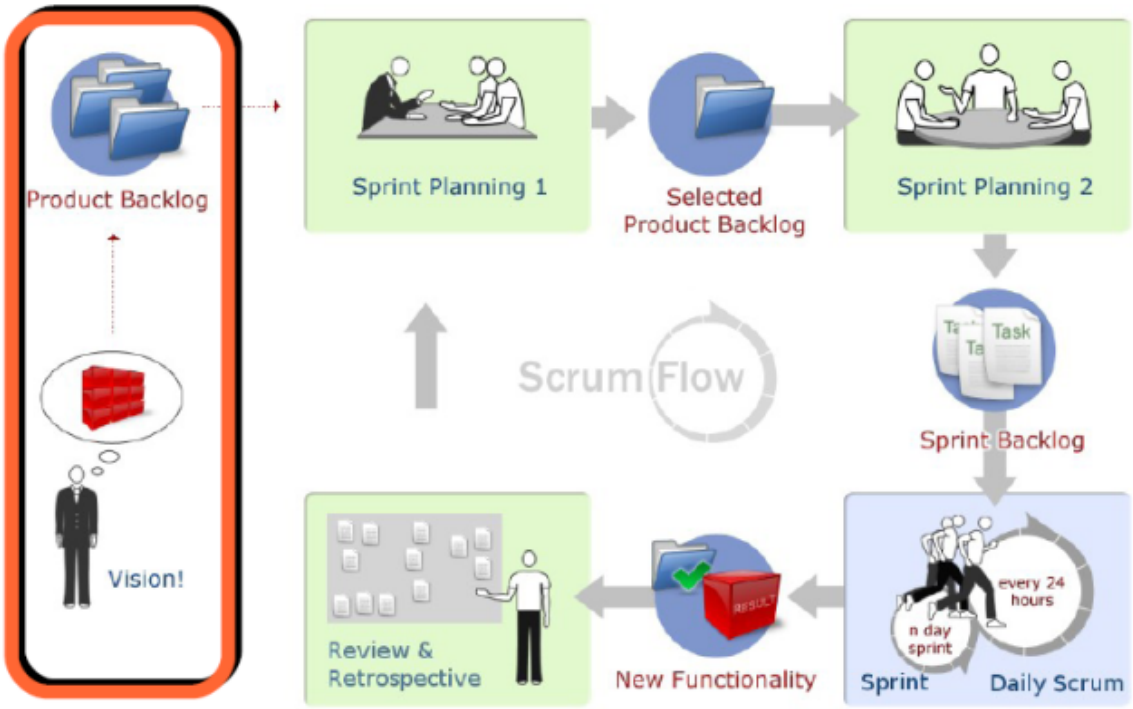

Figura 2: Ciclo de vida do *Scrum* (ALLIANCE, 2009)

O *sprint* (figura 2, *Sprint*) é a fase de implementação em que a equipe irá trabalhar unida para atingir o objetivo de entregar a parte do *software* que está no *selected backlog*. Inicialmente esta fase era recomendada num tamanho padrão de 30 dias corridos; atualmente a duração é definida conforme características do projeto e da equipe. É comum encontrar *sprints* de duas ou três semanas (SCHWABER, 1995).

Durante o *sprint*, o *Scrum Master* assegura que os itens do *selected backlog* não mudarão e elimina os itens do *impediment backlog* garantir que a equipe mantenha-se focada em seu objetivo. O cliente acompanha o desenvolvimento e esclarece eventuais dúvidas, sem poder mudar o escopo porém, caso realmente precise mudar os itens do backlog selecionado, ele tem a opção de cancelar o *sprint*. Neste caso, a iteração volta ao início, para a "*Sprint Planning* 1" da figura 2.

Diariamente durante o *sprint*, os membros da equipe se reúnem no mesmo horário para fazer a *Daily Scrum Meetings*, uma rápida reunião para sincronizar o trabalho. Cada membro da equipe conta o que fez desde o último encontro, o que pretende fazer até o próximo e quais problemas está tendo. Discussões técnicas ou específicas devem ser tratadas individualmente após a reunião, pois ela não deve durar mais do que 15 minutos. O *Scrum Master* atua como facilitador do encontro. O cliente pode participar apenas como ouvinte pois essas discussões dizem respeito ao trabalho da equipe, que deve ter autonomia e liberdade de tomar decisões para alcançar sua meta.

Quando o *sprint* termina, a equipe apresenta o trabalho na *sprint review meeting* (reunião de revisão de um *sprint*) ao cliente através de uma demonstração. O dono do produto faz testes para verificar se cada item atende às suas expectativas e determinar se a meta foi atingida. A meta sempre "é atingida" ou "não é atingida". Não existem opções intermediárias. Desta forma a equipe deve se preocupar com a qualidade de tudo que entrega para não colocar em risco o trabalho do *sprint* inteiro.

Após cada entrega, a equipe realiza uma reunião de retrospectiva (figura 2, *Retrospective*) para avaliar seu trabalho e identificar oportunidades para melhorar seu desempenho nos próximos *sprints*. O *Scrum Master* e os membros da equipe repensam o último *sprint* e refinam o processo de desenvolvimento através de uma reflexão sobre o passado. O resultado da retrospectiva são três listas (SCHWABER, 1995): 1) as ações que trouxeram resultados positivos e devem ser mantidas; 2) as ações sem resultado ou problemas que reduziram o rendimento; e, 3) mudanças e novas práticas para usar no próximo *sprint* e eliminar o que foi apontado na listagem 2. As listas 2 e 3 servem de entrada para o próximo planejamento como pontos a serem considerados.

1.2.3 *Feature Driven Development* (FDD)

FDD foi usada pela primeira vez em 1997 por *Peter Coad* e *Jeff De Luca*. Dois anos depois, publicaram um livro contando essa experiência (COAD; LUCA; LEFEBVRE, 1999). Mais tarde, *Stephen Palmer* e *John Felsing* publicaram uma versão aprimorada e detalhada da metodologia (PALMER; FELSING, 2002). FDD mantém seu foco nas fases de modelagem e implementação, sem a pretensão de cobrir todo o processo de desenvolvimento do *software*. Por isso, ela pode ser combinada com outras técnicas para a produção de sistemas complexos (PALMER; FELSING, 2002).

1.2.3.1 Papéis

FDD divide os papéis em três grupos: papéis chave, papéis de apoio e opcionais. Dentro de cada categoria os papéis podem ser atribuídos a vários participantes para multiplicar uma função ou compartilhar responsabilidades. **Gerente do projeto**, **arquiteto líder**, **gerente de desenvolvimento**, **programador líder**, **dono de classe** e o **especialista de negócios** (cliente) são os responsáveis pelo desenvolvimento da aplicação, por isso são os papéis chave. Os papéis de apoio ao desenvolvimento são o **gerente de versão**, o **guru de linguagem**, o **engenheiro de integração** e o ***toolsmith*** (ferramenteiro), que é o responsável por criar programas que auxiliem os outros desenvolvedores a construir o produto final. Os papéis opcionais são **testadores** e **implantadores**.

1.2.3.2 Processo

Os processos propostos por FDD consiste de cinco etapas que cobrem as fases de modelagem e implementação do *software*. As três primeiras são feitas no início do projeto instituindo um planejamento de nível médio, as duas últimas são repetidas iterativamente para construir o planejado. Segundo *Coad* e *Luca* (1999) as etapas são:

1. **Desenvolver um Modelo Abrangente**: Para criar um modelo de alto nível os especialistas do negócio já devem ter estabelecido um consenso sobre os objetivos, o escopo e as funcionalidades do projeto. Como a metodologia não define como gerenciar os requisitos, as especificações e os casos de uso já devem estar prontos para que a equipe de negócios ofereça explanações de alto nível ao arquiteto líder e os desenvolvedores para que eles criem um modelo panorâmico do projeto. Em seguida, este modelo é subdividido para que pequenos grupos formados por especialistas de negócio e desenvolvedores

abordem cada uma das partes para chegar à modelagem de objetos. Por fim, os objetos são reunidos, revisados e integrados para chegar à versão inicial do modelo de objetos do sistema;

2. **Construir a Lista de Funcionalidades**: Baseado nas explicações dos especialistas de negócio, nos artefatos da especificação e no modelo de objetos criado na fase anterior, é identificada uma lista de funcionalidades com valor de negócio que compreende todo o escopo do projeto. No fim, clientes e usuários revisam e validam a listagem;

3. **Planejar por Funcionalidade**: Objetiva criar um planejamento de alto nível para todo o projeto. Os gerentes e programadores líderes definem a ordem em que as funcionalidades serão implementas baseados no valor de negócios, dependências e volumes de trabalho. As funcionalidades podem ser agrupadas para formar versões potencialmente entregáveis e as classes do modelo atribuídas aos programadores, que passam a ser chamados de donos dessas classes;

4. **Detalhar por Funcionalidade**: Os programadores líderes selecionam conjuntos de funcionalidades para formar pacotes de trabalho com duração de dois dias a duas semanas. Para cada pacote de funcionalidades, os programadores donos das classes envolvidas são convocados para integrar a equipe. Dessa forma, pequenas equipes trabalham em paralelo sob o comando de um programador líder. Se preciso, o modelo de objetos é refinado para comportar as funcionalidades de cada pacote;

5. **Construir por Funcionalidade**: Para cada funcionalidade do pacote, a equipe implementa, testa e inspeciona o código. Quando o programador líder autoriza, as funcionalidades são integradas ao repositório principal e novos pacote e equipe são formados pelo programador líder para a próxima iteração.

A figura 3 representa o graficamente os processos descritos acima.

1.2.3.3 Práticas

FDD utiliza oito técnicas conhecidas na indústria como ”boas práticas”para o desenvolvimento de sistemas (PALMER; FELSING, 2002). Além das vantagens de cada uma, a força da metodologia está no uso combinado das práticas desse conjunto. As práticas citadas são:

1. **Modelagem de domínio de objetos**: Diagramas de classe e de seqüência com os objetos do sistema, mesmo que simplificados, permitem validar possíveis soluções e melhorar o

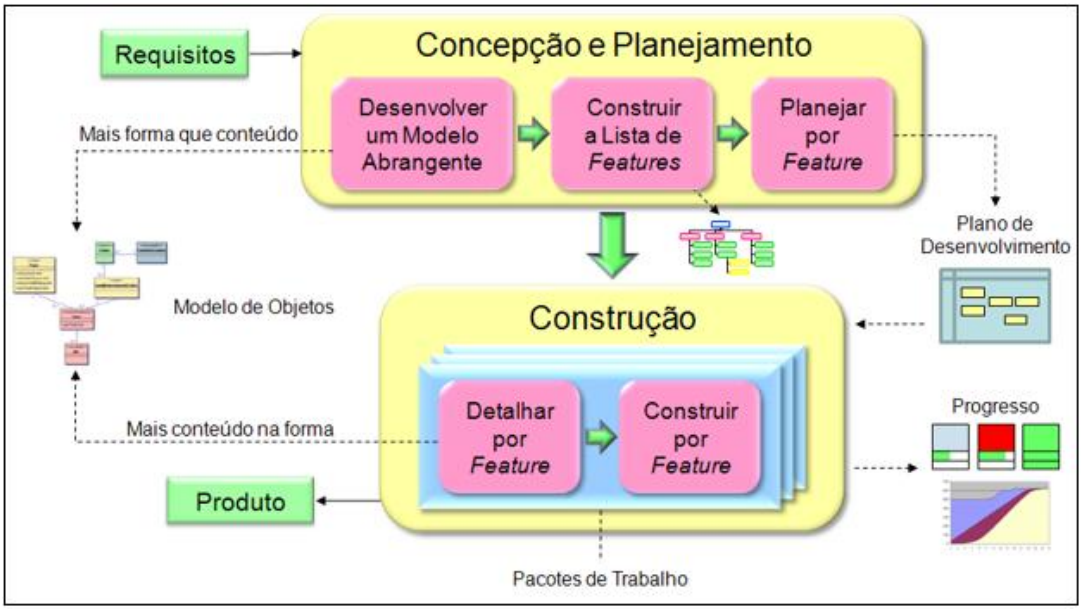

Figura 3: Ciclo de vida do FDD (HEPTAGON, 1997).

entendimento das funcionalidades. A participação da equipe na criação da modelagem de objetos elimina divergências de modelagem e de implementação;

2. **Desenvolvendo por Funcionalidade**: Desenvolver e entregar por funcionalidade garante que o projeto produza o que os clientes desejam, pois o desenvolvimento é guiado pelos requisitos funcionais com valor de negócio e em tamanhos pequenos;

3. **Propriedade de Classes**: Cada classe do sistema possui um programador responsável e cada programador é dono de um grupo de classes. Esta estratégia aumenta o senso de responsabilidade sobre o código e evita problemas com múltiplos desenvolvedores modificando o mesmo código.

4. **Equipes por Funcionalidade**: As equipes são montadas a partir das funcionalidades escolhidas para cada iteração e os participantes são os donos das classes envolvidas. Como várias equipes podem estar trabalhando em paralelo, a melhor forma de evitar problemas com a alocação dos donos das classes é permitir que a classe mude de dono durante o projeto;

5. **Inspeções**: Alguns estudos concluíram que inspeções de código por outros desenvolvedores aumentam a qualidade do código (SHULL *et al.*, 2002). Além disso, inspeções promovem transferência de conhecimento, estimulam a padronização e inibem a inserção de código de baixa qualidade, pois os desenvolvedores sabem que alguém irá verificá-lo mais tarde;

6. **Versões com Regularidade**: A integração frequente do código e a construção frequente de versões estáveis do *software* permite identificar e resolver rapidamente problemas de integração e fornecer informações seguras quanto à evolução do projeto.

7. **Gerenciamento de Configurações**: Idealmente, o sistema de gerenciamento de configurações deve ser simples. No entanto, conforme as características do projeto, é interessante manter um controle de versão para todos os artefatos do projeto;

8. **Relatórios de Progresso**: É importante acompanhar a evolução do projeto e prover informações em níveis diferentes para cada tipo de envolvido com o projeto através de análises e gráficos.

1.3 Melhoria de Processos do *Software* Brasileiro

O MPS.BR é um programa para a melhoria de processo do *software* brasileiro coordenado pela Associação para a Promoção da Excelência do *Software* Brasileiro (SOFTEX), que conta com o apoio do Ministério da Ciência e Tecnologia, da financiadora de Estudos e Projetos (FINEP) e do Banco de Interamericano de Desenvolvimento.

O programa MPS.BR está em desenvolvimento desde 2003. Várias empresas têm se motivado a modificar seus processos de trabalho de forma a aumentar a eficiência e eficácia de seus processos.

O MPS.BR é divido em sete níveis de maturidade, que vão de G a A, com isso o custo de certificação para cada nível fica menor, em relação aos cinco níveis de maturidade do CMMI, facilitando a certificação de muitas empresas, que não poderiam se certificar em um modelo como o CMMI, de custo mais elevado.

Na figura 4 é mostrado cada passo da evolução da maturidade segundo o modelo MPS.BR e uma breve descrição dos requisitos de cada fase. Como citado anteriormente, os níveis evoluem do nível G ao A, identificados na primeira coluna da tabela. É importante lembrar que para atingir um nível superior proposto no modelo a empresa deve estar em conformidade com todos os níveis anteriores.

Na segunda coluna, denominada **Processos**, demonstra-se quais os processos devem ser atendidos para estar em conformidade com um nível específico. Os processos são expressos através de seus propósitos e resultados esperados. O propósito define o objetivo a ser alcançado, considerando o processo executado, já os resultados esperados descrevem os objetivos parciais que devem ser alcançados para atender ao propósito do processo.

Na terceira coluna, denominada **Atributos de Processo**, representa um conjunto de atributos de processo (AP), que, por sua vez, são expressos através de resultados para cada atributo. A Capacidade representa o nível de refinamento do processo na empresa. Conforme a empresa vai aumentando seu nível de maturidade, um maior nível desse refinamento, capacidade, passa a ser exigido.

Como o foco deste trabalho é o mapeamento das práticas dos processos ágeis com o nível G, este será abordado com um pouco mais de detalhes a partir de agora.

1.3.1 Nível G

O nível de maturidade G é composto pelos processos Gerência de Projetos e Gerência de Requisitos.

1.3.1.1 Processo: Gerência de Projetos - GPR

O propósito do processo Gerência de Projetos é estabelecer e manter planos que definem as atividades, recursos e responsabilidades do projeto, bem como prover informações sobre o andamento do projeto que permitam a realização de correções quando houver desvios significativos no desempenho do projeto. O propósito deste processo evolui à medida que a organização cresce em maturidade (níveis).

Os resultados esperados, segundo o guia de implementação (SOFTEX MR-MPS, 2009), são:

- GPR1 - O escopo do trabalho para o projeto é definido;
- GPR2 - As tarefas e os produtos de trabalho do projeto são dimensionados utilizando métodos apropriados;
- GPR3 - O modelo e as fases do ciclo de vida do projeto são definidos;
- GPR4 - O esforço e o custo para a execução das tarefas e dos produtos de trabalho são estimados com base em dados históricos ou referências técnicas;
- GPR5 - O orçamento e o cronograma do projeto, incluindo a definição de marcos e pontos de controle, são estabelecidos e mantidos;
- GPR6 - Os riscos do projeto são identificados e o seu impacto, probabilidade de ocorrência e prioridade de tratamento são determinados e documentados;

- GPR7 - Os recursos humanos para o projeto são planejados considerando o perfil e o conhecimento necessários para executá-lo;
- GPR8 - Os recursos e o ambiente de trabalho necessários para executar o projeto são planejados;
- GPR9 - Os dados relevantes do projeto são identificados e planejados quanto à forma de coleta, armazenamento e distribuição. Um mecanismo é estabelecido para acessá-los, incluindo, se pertinente, questões de privacidade e segurança;
- GPR10 - Um plano geral para a execução do projeto é estabelecido com a integração de planos específicos;
- GPR11 - A viabilidade de atingir as metas do projeto, considerando as restrições e os recursos disponíveis, é avaliada. Se necessário, ajustes são realizados;
- GPR12 - O Plano do Projeto é revisado com todos os interessados e o compromisso com ele é obtido;
- GPR13 - O projeto é gerenciado utilizando-se o Plano do Projeto e outros planos que afetam o projeto e os resultados são documentados;
- GPR14 - O envolvimento das partes interessadas no projeto é gerenciado;
- GPR15 - Revisões são realizadas em marcos do projeto e conforme estabelecido no planejamento;
- GPR16 - Registros de problemas identificados e o resultado da análise de questões pertinentes, incluindo dependências críticas, são estabelecidos e tratados com as partes interessadas;
- GPR17 - Ações para corrigir desvios em relação ao planejado e para prevenir a repetição dos problemas identificados são estabelecidas, implementadas e acompanhadas até a sua conclusão.

1.3.1.2 Processo: Gerência de Requisitos - GRE

O propósito do processo Gerência de Requisitos é gerenciar os requisitos do produto e dos componentes do produto do projeto e identificar inconsistências entre os requisitos, os planos do projeto e os produtos de trabalho do projeto.

Os resultados esperados, segundo o guia de implementação (SOFTEX MR-MPS, 2009), são:

- GRE1 - Os requisitos são entendidos, avaliados e aceitos junto aos fornecedores de requisitos, utilizando critérios objetivos;
- GRE2 - O comprometimento da equipe técnica com os requisitos aprovados é obtido;
- GRE3 - A rastreabilidade bidirecional entre os requisitos e os produtos de trabalho é estabelecida e mantida;
- GRE4 - Revisões em planos e produtos de trabalho do projeto são realizadas visando a identificar e corrigir inconsistências em relação aos requisitos;
- GRE5 - Mudanças nos requisitos são gerenciadas ao longo do projeto.

Nível	Processos	Atributos de Processo
A		AP 1.1, AP 2.1, AP 2.2, AP 3.1, AP 3.2, AP 4.1, AP 4.2 , AP 5.1 e AP 5.2
B	Gerência de Projetos – GPR (evolução)	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2, AP 4.1 e AP 4.2
C	Gerência de Riscos – GRI Desenvolvimento para Reutilização – DRU Gerência de Decisões – GDE	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
D	Verificação – VER Validação – VAL Projeto e Construção do Produto – PCP Integração do Produto – ITP Desenvolvimento de Requisitos – DRE	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
E	Gerência de Projetos – GPR (evolução) Gerência de Reutilização – GRU Gerência de Recursos Humanos – GRH Definição do Processo Organizacional – DFP Avaliação e Melhoria do Processo Organizacional – AMP	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
F	Medição – MED Garantia da Qualidade – GQA Gerência de Portfólio de Projetos – GPP Gerência de Configuração – GCO Aquisição – AQU	AP 1.1, AP 2.1 e AP 2.2
G	Gerência de Requisitos – GRE Gerência de Projetos – GPR	AP 1.1 e AP 2.1

Figura 4: Níveis de Maturidade do MPS.BR (SOFTEX, 2009)

2 Metodologia

Para alcançar os objetivos deste trabalho, foram definidas algumas etapas. A primeira delas foi a realização do levantamento bibliográfico. Foi estudado o modelo de qualidade de *software* MPS.BR, referente ao escopo deste trabalho. As MAs também foram abordadas neste levantamento bibliográfico, a fim de fornecer os elementos necessários para a realização deste trabalho. Dentre elas XP, FDD e SCRUM. Além dos modelos de qualidades e as MAs, trabalhos que propõem o relacionamento entre estes dois temas, também forma analisados. Estes trabalhos serviram como base para analisar o formato utilizado no relacionamento entre MAs e os modelos de qualidade de *software*.

Para a realização do mapeamento foi detectado a necessidade de definir critérios para classificar a equivalência entre as MAs e o nível G do MPS.BR. Através da observação dos artigos e monografias relacionadas a este trabalho, foi identificado um conjunto de critérios que atendem o escopo deste trabalho. Foram selecionados 3 critérios para a classificação das metodologias ágeis em relação aos requisitos do MPS.BR. Os critérios selecionados tiveram como base um artigo que realiza o mapeamento entre o Scrum e o modelo de qualidade CMMI (MARCAL; SOARES; BELCHIOR, 2007). Estes critérios estão presentes na tabela 2.

Tabela 2: Critérios para classificação das MAs X MPS.BR

Classificação	**Critério**
(S) Satisfeito	A prática está totalmente em conformidade
(PS) Parcialmente Satisfeito	Há evidências, embora não esteja plenamente em conformidade.
(NS) Não Satisfeito	Não há evidências

Os critérios mostrados na tabela 2 foram selecionados com o objetivo de satisfazer três possíveis situações:

1. Satisfeito - A metodologia ágil atende totalmente os resultados esperados pelo requisito do MPS.BR;

2. Parcialmente Satisfeito - A metodologia ágil atende parcialmente os resultados esperados pelo requisito do MPS.BR;

3. Não Satisfeito - A metodologia ágil não possui práticas relacionadas às atividades solicitadas pelo requisito do MPS.BR;

Para tentar evitar interpretações dúbias foi definido um formato para o mapeamento dos requisito do MPS.BR. Este formato é composto de trechos pertinentes da descrição dos resultado esperados (do guia de implementação) e em seguida a análise sobre como as práticas das MAs estão relacionadas com os resultados esperados. Foi definido este formato, pois só o título do requisito não expressa completamente o resultado esperado pelo requisito.

A realização do mapeamento representando a etapa central deste trabalho, foi realizada após a definição dos critérios para a classificação e formato do mapeamento. Cada uma das metodologias foi analisada de acordo com a descrição dos requisitos proposto pelo modelo de referência (MR.MPS), utilizando os critérios selecionados para a classificação evidenciando o **como** as práticas das MAs atingem (ou não) os resultados esperados do nível G do MPS.BR.

A etapa seguinte foi a de análise dos resultados do mapeamento. Para melhorar a visualização destes resultados foram criadas tabelas contendo as informações sintetizadas do mapeamento. Estas tabelas permitem uma forma de representação visual para o resultado obtido. As tabelas criadas se encontram no apêndice A e B.

3 Mapeamento MAs X Nível G (MPS.BR)

Conforme definido na sessão 1.3 do capítulo 1, o MPS.BR visa a maturidade do processo. Para empresas que estão iniciando sua jornada em busca do amadurecimento de seus processos o XP, Scrum e FDD pode trazer vantagens em relação à certificação se comparada com outras empresas que tentam criar uma metodologia baseada apenas na experiência coletiva de sua equipe. Neste contexto, as sessões a seguir têm como objetivo mapear as MAs em questão com o nível G (composto pelos processos Gerência de Projetos (GRP) e Gerência de Requisitos (GRE)) do MPS.BR.

3.1 Gerência de Projetos (GPR) X MAs

O propósito do processo Gerência de Projetos é estabelecer e manter planos que definem as atividades, recursos e responsabilidades do projeto, bem como prover informações sobre o andamento do projeto que permitam a realização de correções quando houver desvios significativos no desempenho do projeto.

Nesta seção é apresentado o mapeamento das MAs em relação ao processo Gerência de Projetos, descrito no guia de implementação do MPS.BR.

- GPR1 - O escopo do trabalho para o projeto é definido.

> [...] O escopo do projeto define todo o trabalho necessário, e somente ele, para entregar um produto que satisfaça as necessidades, características e funções especificadas para o projeto, de forma a concluí-lo com sucesso. [...] Este resultado também pode ser implementado por meio de um Documento de Visão ou outro documento que defina, claramente, o escopo do trabalho (SOFTEX MR-MPS, 2009, p.9).

  - **XP**: O XP não define o escopo de todo o projeto no inicio. O escopo do projeto é dividido através das *releases* (BECK, 1999), que são construídas no "Jogo do Planejamento", contendo funcionalidades prioritárias para o cliente. Desta forma **o XP**

atende parcialmente este requisito, pois para atender totalmente é necessário que as *releases* sejam todas definidas no inicio do projeto.

- ***Scrum***: Para atender este requisito o *Scrum* utiliza o *product backlog*, onde este possui uma lista com todos os requisitos que o sistema deve apresentar, representando o escopo do projeto. **Desta forma este requisito é atendido pelo Scrum**.
- **FDD**: A definição do escopo no FDD ocorre através de duas etapas "Desenvolver o modelo abrangente", onde posteriormente este modelo é decomposto em funcionalidades, através da prática: "Construir a lista de funcionalidades". Desta forma **o FDD contempla este requisito**.

- GPR2 - As tarefas e os produtos de trabalho do projeto são dimensionados utilizando métodos apropriados.

  Segundo o guia de implementação do Nível G do MR-MPS (Modelo de Referência do MPS.BR):

  > Uma estrutura de decomposição do trabalho apropriada deve ser estabelecida. [...]No nível G, a estimativa de escopo, produtos e tarefas pode ser feita baseada na complexidade, no número de requisitos ou no uso da EAP juntamente com dados históricos e a experiência em projetos anteriores. (SOFTEX MR-MPS, 2009, p.10)

  - **XP**: Uma das etapas da prática "Jogo do Planejamento" é a decomposição de estórias, em tarefas, onde uma história pode originar muitas tarefas. Esta decomposição ocorre baseada em experiências anteriores. O XP não define claramente o método de decomposição, ficando a critério da equipe, e também não mantém dados históricos sobre a decomposição de requisitos. Devido a falta de formalismo e base histórica para o dimensionamento das tarefas **o XP atende parcialmente este requisito**.
  - ***Scrum***: No Scrum, após a criação do *selected backlog* a equipe "quebra" os itens selecionados em tarefas menores para a criação do *sprint backlog*. No entanto não há orientação explícita no *Scrum* para estabelecer, por exemplo, o tamanho e / ou a complexidade dos elementos do *Product Backlog* e *Sprint Backlog*. Nenhum método é explicitamente mencionado por esta metodologia para orientar no dimensionamento dos produtos do trabalho como também não mantém dados históricos sobre a decomposição de requisitos. Devido a falta de formalismo e base histórica para o dimensionamento das tarefas **o Scrum atende parcialmente este requisito**.

  - **FDD**: Na segunda etapa do processo, "Construir a Lista de Funcionalidades", ocorre a decomposição funcional das áreas do negócio definidas na primeira etapa. Apesar desta decomposição ocorrer, o FDD não evidencia métodos para esta e também não mantém dados históricos sobre a decomposição de requisitos. Devido a falta de formalismo e base histórica para o dimensionamento das tarefas **o FDD atende parcialmente este requisito**.

- GPR3 - O modelo e as fases do ciclo de vida do projeto são definidas.

  Segundo o guia de implementação do Nível G do MR-MPS (Modelo de Referência do MPS.BR):

  > [...] O ciclo de vida de projeto define um conjunto de fases, que por sua vez geram produtos de trabalho necessários para o desenvolvimento de fases posteriores. Essa organização em fases permite planejar o projeto, incluindo marcos importantes para o controle e revisões (SOFTEX MR-MPS, 2009, p.10).

  - **XP**: O modelo do ciclo de vida do XP é definido de acordo com a figura 1 na página 23. Neste ciclo de vida são evidenciadas as etapas e a comunicação entre elas, além de definir marcos para o controle do projeto. Sendo assim **o XP atende este requisito**.
  - ***Scrum***: O ciclo de vida do *Scrum* é realizado por etapas bem definidas (ALLIANCE, 2009), ver figura 2 na página 28.

    Cada etapa do ciclo de vida do *Scrum* fornece, à etapa seguinte, informações e/ou artefatos necessários para a continuidade do ciclo, além de prover pontos de controle sobre o projeto. Desta forma **o *Scrum* contempla este requisito**.
  - **FDD**: O Ciclo do FDD é evidenciado pela figura 3 na página 32.

    Cada etapa do FDD possuem critérios de entradas e saídas (COAD; LUCA; LEFEBVRE, 1999), ou seja, cada etapa do processo produz dados necessários para a etapa posterior. O ciclo de vida do FDD define pontos de controles para o projeto, sendo estes ao final da etapa "Construir por Funcionalidade", gerando as funcionalidades do projeto. Desta forma o FDD contempla este requisito.

- GPR4 - O esforço e o custo para a execução das tarefas e dos produtos de trabalho são estimados com base em dados históricos ou referências técnicas.

  Segundo o guia de implementação do Nível G do MR-MPS:

> As estimativas de esforço e custo tipicamente consideram: o escopo, produtos de trabalho e as tarefas estimadas para o projeto; os riscos; as mudanças já previstas; o ciclo de vida escolhido para o projeto; viagens previstas; nível de competência da equipe do projeto, dentre outros. [...] Os parâmetros de produtividade podem ter valores diversos, conforme fatores como tecnologia adotada, experiência do profissional, grau de ineditismo do serviço para a organização ou para os profissionais alocados. [...]Empresas implementando o nível G do MR-MPS geralmente não possuem bases de dados históricas. (SOFTEX MR-MPS, 2009, p.11)

  - **XP**: Baseado na descrição do requisito do MR-MPS, **o XP contempla este requisito**, devido a não obrigatoriedade de manter dados históricos de estimativas e custo das tarefas e produtos de trabalho para o nível G. O XP estima o custo e esforços das tarefas na prática "Jogo do Planejamento", onde cada tarefa é avaliada e estimada de acordo com os parâmetros de produtividade comentados na citação acima. É importante salientar que no XP não há uma separação entre esforço e custo, ou seja, a equipe estima o tempo para a tarefa já considerando os parâmetros de esforço e custo considerados pelo guia de implementação, onde este tempo representa um custo para o cliente.
  - ***Scrum***: As estimativas são divididas em duas etapas pelo *Scrum*, onde a primeira é realizada diretamente no *Product Backlog*, de forma mais alto nível, proporcionando visibilidade do tamanho de cada requisito. A segunda etapa é realizada no *Sprint Backlog*, onde as estimativas são mais precisas do que as primeiras, a equipe estima tendo como base o desempenho em *sprints* anteriores, capacidade para o próximo *sprint* e complexidade relativa das tarefas. O *Scrum* não menciona formas para armazenar informações de estimativas de esforço, e também o custo não é mencionado explicitamente no processo. Contudo o nível G do MPS.BR não exige o registro em bases históricas, desta forma **o *Scrum* atende parcialmente este requisito**, pois trata da estimativa de esforço mas não do custo para os produtos de trabalho.
  - **FDD**: As estimativas de esforço no FDD, acontece na etapa "Planejar por Funcionalidade", estas estimativas são realizadas através da complexidade e experiências anteriores, como resultado desta etapa é gerado o Plano de Desenvolvimento, onde é constituído, dentre outras coisas, com a lista de atividades de negócio com datas de término (mês e ano) (HEPTAGON, 1997). Por outro lado, não há evidências da estimativa de custo no FDD, desta forma, **este requisito é atendido parcialmente**.

- GPR5 - O orçamento e o cronograma do projeto, incluindo marcos e/ou pontos de controle, são estabelecidos e mantidos.

> As dependências entre tarefas são estabelecidas e potenciais gargalos são identificados utilizando métodos apropriados (por exemplo, análise de caminho crítico). Os gargalos são resolvidos quando possível e o cronograma das atividades com início, duração e término é estabelecido. Recursos requeridos são refletidos nos custos estimados. [..] O orçamento do projeto é estabelecido com base no cronograma e na estimativa de custos (SOFTEX MR-MPS, 2009, p.12).

– **XP**: O cronograma do projeto no XP é definido em etapas através de *releases*, onde estas são planejadas no "Jogo do Planejamento", em que as estórias são priorizadas pelo cliente e suas dependências analisadas pela equipe. O XP não define todo o cronograma no início do projeto, este é construído de forma incremental.

  Da mesma forma que o cronograma, o orçamento do projeto é definido de forma incremental, tendo como parâmetro a velocidade da equipe (TELES, 2005), ou seja, o orçamento é definido por *release*. Desta forma **o XP atende parcialmente este requisito**.

– ***Scrum***: A partir do *Product Backlog*, é obtido o cronograma, onde este é priorizado de acordo com as dependências entre os *Itens Backlog* e subdividido em *Sprints* considerando a alocação da equipe de acordo com sua capacidade de produção. O cronograma é então definido através do número de *Sprints*, onde cada um tem tempo médio de 30 dias (MARCAL; SOARES; BELCHIOR, 2007).Os pontos de controle são as datas finais de cada *Sprint*, onde ocorre a *Sprint Review Meeting*.

  Em relação ao orçamento do projeto, o *Scrum* não contempla, pois para isto, seria necessário a realização de estimativa de custo (GPR4) *Itens Backlog*, o que não ocorre. Sendo assim não há como obter um orçamento aproximado para o projeto através somente do cronograma. Desta forma **o *Scrum* atende parcialmente este requisito**.

– **FDD**: Ao final da etapa "Planejar por Funcionalidade" é gerado o Plano de Desenvolvimento, onde neste se encontra as dependências entre funcionalidades, a sequencia de desenvolvimento e todas as atividades de negócio com suas respectivas datas de términos e pontos de controle (HEPTAGON, 1997), representando o cronograma do projeto.

  Em relação ao orçamento do projeto, o FDD não contempla, pois para isto, seria necessário a realização de estimativa de custo (GPR4) sobre as atividades, o que não ocorre. Sendo assim não há como obter um orçamento aproximado para o projeto. Desta forma **o FDD atende parcialmente este requisito**.

- GPR6 - Os riscos do projeto são identificados e o seu impacto, probabilidade de ocorrência e prioridade de tratamento são determinados e documentados.

  Segundo o guia de implementação do Nível G do MR-MPS:

  > Os riscos identificados devem ser registrados, bem como o acompanhamento dos seus estados e ações tomadas. Uma planilha de riscos, contendo dados como identificador, descrição, probabilidade, impacto e prioridades no seu tratamento, pode ser utilizada para identificação dos riscos, monitoração dos riscos identificados e atualização da lista de riscos do projeto à medida que novos riscos forem sendo identificados. É importante demonstrar que esta planilha está sendo monitorada e atualizada. [...]Este resultado não significa o Gerenciamento de Riscos, ou seja, a identificação, o gerenciamento e a redução contínua dos riscos nos níveis organizacionais e de projeto, que são tratados pelo processo Gerência de Riscos (GRI). No nível G, os riscos são acompanhados para verificar como afetam o projeto e para se tomar ações, mesmo que ainda sem um gerenciamento completo. (SOFTEX MR-MPS, 2009, p.12).

  - **XP**: A identificação e tratamentos de riscos pelo XP, ocorre diretamente e indiretamente, através de alguns valores e práticas, tais como: Entregas frequentes, *feedback* rápido, Jogo do Planejamento, Cliente Presente, TDD (*Test-driven development*), Refatoração e Programação em Pares. Entretanto o XP não possui práticas que evidencie o registro de identificação e acompanhamento destes riscos. Logo **o XP atende parcialmente este requisito**.
  - ***Scrum***: A identificação dos riscos é realizada de forma iterativa, durante as reuniões diárias da equipe, sendo documentados em *whiteboards*, *flipcharts* e na lista de impedimentos (*impediment backlog*), estes riscos são acompanhados pelo Scrum *Master*. **O *Scrum* atende este requisito**.
  - **FDD**: O FDD possui o papel do Gerente de Desenvolvimento, que é responsável pela resolução de conflitos que possa vir a ocorrer dentro da equipe. Na etapa "Planejar por funcionalidade" ocorre o adiantamento de atividades de negócio de alto risco e/ou complexidade (HEPTAGON, 1997), no entanto não há outras práticas sobre a identificação e acompanhamento de riscos. Desta forma **o FDD atende parcialmente este requisito**.

- GPR7 - Os recursos humanos para o projeto são planejados considerando o perfil e o conhecimento necessários para executá-lo.

  Segundo o guia de implementação do Nível G do MR-MPS:

> O planejamento de recursos humanos determina funções, responsabilidades e relações hierárquicas do projeto. [...]O planejamento de recursos humanos inclui informações de como e quando o recurso será envolvido no projeto, critérios para sua liberação, competência necessária para a execução das atividades, mapa de competências da equipe e identificação de necessidades de treinamento. A existência de registros das necessidades e disponibilidade evita a alocação com base em critérios subjetivos (SOFTEX MR-MPS, 2009, p.13).

  - **XP**: Não há no XP, evidencias de tais registros sobre alocação de recursos humanos. Na fase inicial de um projeto em XP (Fase de Exploração) é considerado que já existe uma equipe.

    O XP se adequa a equipes com conhecimentos heterogêneos (TELES, 2004), não formalizando os conhecimentos necessários ao projeto. Por outro lado o XP define um conjunto de papéis, nestes são descritos as responsabilidades e relações hierárquicas entre os mesmos (BECK, 1999). Desta forma **o XP contempla parcialmente este requisito**.
  - ***Scrum***: A alocação da equipe é realizadas no início do projeto, sob a responsabilidade do *Scrum Master*, durante a fase de *Staging* (MARCAL; SOARES; BELCHIOR, 2007). O *Scrum* não evidencia a prática de registrar os critérios para a seleção de membros, ficando a critério do *Scrum Master*. Devido a falta de registro, o ***Scrum*** **atende parcialmente este requisito**.
  - **FDD**: O FDD possui papéis definidos, onde em cada um é descrito as funções responsabilidades, juntamente com seu nível hierárquico (COAD; LUCA; LEFEBVRE, 1999). O planejamento dos recursos humanos é realizado na etapa "Planejar por Funcionalidade", onde as pessoas da equipe são alocadas de acordo com a ordem de execução das funcionalidades. Nas definições de responsabilidades do FDD, o Gerente de Projetos possui a responsabilidade de obter os recursos humanos, mas o processo não evidencia critérios para tal, nem tão pouco o registro desses critérios. Devido a falta de registros destas atividades **este requisito é atendido parcialmente pelo FDD**.

- GPR8 - As tarefas, os recursos e o ambiente de trabalho necessários para executar o projeto são planejados.

  Segundo o guia de implementação do Nível G do MR-MPS:

  > Todos os recursos precisam ser explicitamente planejados, mesmo os já considerados como existentes e disponíveis ou que serão compartilhados com outros projetos, uma vez que se trata da sua alocação para uso.

> Estes itens podem, por exemplo, estar registrados no plano do projeto. Caso não haja necessidade de nenhum recurso a ser adquirido para o projeto deve-se registrar o fato de que a questão foi examinada (SOFTEX MR-MPS, 2009, p.13).

  - **XP**: O planejamento formal de recurso não está no escopo do XP. O que ocorre neste sentido, acontece no "Jogo do Planejamento" onde são abordadas as tarefas a serem implementadas na *release* e os recursos necessários para implementá-las, caso seja necessário. É de responsabilidade do "coach" organizar o ambiente de trabalho na fase de exploração (BECK; FOWLER, 2000), mas não há registro deste planejamento. Desta forma o **XP atende parcialmente este requisito**.
  - ***Scrum***: A preparação da infra-estrutura do projeto é realizada no início do projeto durante a fase de *Staging* (MARCAL; SOARES; BELCHIOR, 2007). Ao *Product Backlog* são adicionados os recursos necessários ao desenvolvimento, tais como: máquinas, ferramentas e demais investimentos necessários para a configuração do ambiente de desenvolvimento. Além disso, o *Scrum Master* é o responsável por prover os recursos necessários ao longo do projeto de acordo com necessidades e impedimentos que são reportados nas reuniões diárias. Sendo assim o ***Scrum*** **contempla este requisito**.
  - **FDD**: O planejamento de recursos e ambiente de trabalho no FDD, são responsabilidades do Gerente de Projeto (ABRAHAMSSON *et al.*, 2002, p.53), fornecendo condições de trabalho adequadas para a equipe. Mas não há evidências sobre o registro deste planejamento. Portanto o **FDD atende parcialmente este requisito**.

- GPR9 - Os dados relevantes do projeto são identificados e planejados quanto à forma de coleta, armazenamento e distribuição. Um mecanismo é estabelecido para acessá-los, incluindo, se pertinente, questões de privacidade e segurança.

  Segundo o guia de implementação do Nível G do MR-MPS:

> [...]Os dados do projeto são as várias formas de documentação exigidas para sua execução, por exemplo: relatórios; dados informais; estudos e análises; atas de reuniões; documentação; lições aprendidas; artefatos gerados; itens de ação; e indicadores. Os dados podem estar em qualquer formato e existir em qualquer meio, como: impressos ou desenhados em diversos materiais; fotografias; meio eletrônico; e multimídia (SOFTEX MR-MPS, 2009, p.14).

- **XP**: Os dados relevantes para um projeto em XP são as estórias de usuários e o código. Todos os envolvidos no projeto possuem acesso aos cartões de estórias e a equipe tem acesso ao código através de ferramentas de versionamento, pois no XP o código é propriedade coletiva (BECK, 1999). O XP coleta as estórias de usuários de forma planejada, para a definição das *releases*. **Este requisito é atendido pelo XP**.
- ***Scrum***: Os dados relevantes para *Scrum*, são:
  - O *Product Backlog*;
  - O *Sprint Backlog*;
  - O *Impediment Backlog*;
  - O gráfico *Burndown*.

  Estes itens acima são coletados e/ou gerados em etapas específicas do projeto, de forma planejada. O acesso a estes artefatos é livre entre os membros do projeto, de forma a manter todos os participantes no mesmo nível de conhecimento sobre o projeto e seu andamento. **O *Scrum* atende este requisito.**
- **FDD**: Num projeto que utiliza o FDD como metodologia, os principais artefatos gerados são:
  - Modelo Abrangente de Dados (diagrama de classes);
  - A lista de funcionalidades;
  - O plano de desenvolvimento, com data de início e término para cada funcionalidade;
  - Diagramas de sequencia e diagramas de classes atualizado.

  A geração dos artefatos mencionados acima, ocorre de forma planejada e em etapas bem definidas. Seu armazenamento é realizado de forma que a equipe tenha fácil acesso. Ao contrário do XP, cada classe possui um "dono", onde qualquer alteração a uma de suas classes tem que passar por sua inspeção e autorização (COAD; LUCA; LEFEBVRE, 1999). Desta forma **o FDD contempla este requisito.**

- GPR10 - Planos para a execução do projeto são estabelecidos e reunidos no Plano do Projeto.

  Segundo o guia de implementação do Nível G do MR-MPS:

  > [...] A realização do planejamento do projeto é garantida pelos resultados esperados no escopo do nível G do MR-MPS do processo Gerência de Projetos (GPR), que prevê, dentre outros, a criação do cronograma de

> atividades, o planejamento de recursos humanos, custos, riscos, dados etc. A reunião destes documentos é entendida como sendo o Plano de Projeto. As tarefas do processo definido para o projeto podem, também, fazer parte deste Plano do Projeto (SOFTEX MR-MPS, 2009, p.14).

OBS.: A análise deste requisito é realizada considerando o ato de reunir e não o de criar os documentos solicitados. Por exemplo: se uma metodologia confecciona todos os documentos solicitados mas não evidencia práticas que os reuni, para a geração do Plano de Projeto, esta metodologia será classificada como "Não Satisfaz". No capítulo 4 é discutido os resultado do mapeamento deste requisito.

- **XP**: O XP não produz os documentos solicitados neste requisito para a criação do Plano de Projeto exceto pelo cronograma das atividades de uma *release*. Desta forma **o XP não contempla este requisito**.
- ***Scrum***: O *product backlog* é constituído, dentre outras coisas, pelos requisitos do *software* e recursos necessários para a infra-estrutura (ver análise do GPR8). Os riscos identificados são registrados no *impediment backlog*, mas este não é integrado ao *product backlog*. O Scrum também gera um cronograma das atividades presentes no *product backlog*. Os outros documentos resultados de planejamentos, não são produzidos pelo *Scrum*. Devido a falta destes outros documentos e a falta de integração da documentação produzida em um único documento (Plano de Projeto) **o *Scrum* não contempla este requisito**.
- **FDD**: No FDD é gerado um plano de desenvolvimento contendo:
  - Atividades de negócio com datas de término (mês e ano);
  - Programadores líderes atribuídos a atividades de negócio;
  - Áreas com datas de término (mês e ano), derivadas da data do último término de suas respectivas atividades de negócio;
  - Lista das classes e seus respectivos desenvolvedores proprietários (a lista de proprietários de classes).

  Os outros documentos solicitados por este requisito, não são produzidos pelo FDD e também não há práticas que evidenciam a reunião dos documentos gerados em um único documento, para a geração do Plano de Projeto. Desta forma **o FDD não contempla este requisito**.

- GPR11 - A viabilidade de atingir as metas do projeto, considerando as restrições e os recursos disponíveis, é avaliada. Se necessário, ajustes são realizados.

  Segundo o guia de implementação do Nível G do MR-MPS:

> [...] No início do projeto, uma avaliação preliminar pode ser conduzida, a partir da visão geral dos objetivos e características dos resultados pretendidos, dos recursos financeiros, técnicos e humanos, bem como de restrições impostas pelo cliente, ambiente externo e interno, além de condições para o desenvolvimento. À medida que o projeto evolui, a viabilidade de sucesso pode ser reavaliada com mais precisão. As mudanças de requisitos são eventos que podem levar à necessidade de reavaliar a viabilidade do projeto. Em marcos do projeto e mesmo durante as atividades de acompanhamento, pode ser necessária a confirmação da viabilidade de continuidade do projeto (SOFTEX MR-MPS, 2009, p.15).

– **XP**: No XP o acompanhamento do projeto ocorre diariamente através das "reuniões diárias". Outros marcos são os finais das iterações, onde são avaliados os progressos, atrasos, problemas ocorridos, etc. Há também esta análise ao final de uma *release*. Desta forma o XP avalia junto ao cliente, a viabilidade de atingir as metas e adaptações destas às mudanças de requisitos. Sendo assim o **XP contempla este requisito**.

– ***Scrum***: No *Scrum* há avaliações diárias sobre o andamento do projeto. Estas são realizadas nas *Daily Scrum Meetings* (reuniões diárias). Esta reunião é utilizada para que a equipe relate o que foi feito desde a última reunião, o que será feito até a próxima reunião e para informar a causa para determinados atrasos. Com essas informações o *Scrum Master* reavalia o projeto e toma medidas para contornar os problemas relatados. Há também as reuniões de planejamentos de *sprints*, reuniões de revisão de *sprint* e reuniões de retrospectiva de *sprints*, ou seja, em intervalos pequenos (dias, semanas e meses) a viabilidade de atingir as metas do projeto é reavaliada, considerando a mudança de requisitos, com a presença do cliente nestas reuniões. Desta forma o ***Scrum*** **atende este requisito.**

– **FDD**: A etapa "Planejar por Funcionalidade" gera um plano de desenvolvimento (COAD; LUCA; LEFEBVRE, 1999), contendo dentre outras coisas:

  * Atividades de negócio com datas de término (mês e ano);
  * Áreas com datas de término (mês e ano), derivadas da data do último término de suas respectivas.

  Este plano de desenvolvimento é a base para o acompanhamento do projeto e revisto com a presença do cliente. Este plano geralmente é alterado e revisto devido as mudanças de requisitos pelo cliente. Há também as práticas: "Entregas frequentes", onde a cada entrega o projeto é reavaliado e "Relatórios de Progresso" para que

a evolução do projeto seja acompanhada por todos. Logo o **FDD contempla este requisito**.

- GPR12 - O Plano do Projeto é revisado com todos os interessados e o compromisso com ele é obtido.

  Segundo o guia de implementação do Nível G do MR-MPS:

  > [...] Obter o compromisso pode envolver a interação entre todos os interessados relevantes internos e externos ao projeto. Os indivíduos ou grupos que se comprometem deverão ter a confiança de que o trabalho pode ser executado dentro das restrições de custo, cronograma e desempenho. Algumas organizações costumam realizar uma reunião de início do projeto (*kick off*) que pode ser utilizada para resolver os conflitos e obter o comprometimento (SOFTEX MR-MPS, 2009, p.15).

  – **XP**: No "Jogo do Planejamento" o projeto é revisado na presença da equipe e do cliente (TELES, 2005). Esta prática é utilizada, dentre outras coisas, para que o cliente possa selecionar as estórias que comporão a próxima *release*, reafirmando o compromisso com todos os participantes do projeto.

    Em projetos que utilizam XP, a interação entre o cliente e a equipe é intensa, isso fortalece o comprometimento de ambas as partes (BECK, 1999).

    Entretanto o XP não produz o Plano do Projeto solicitado no GPR10. Devido a falta deste plano **o XP contempla parcialmente este requisito**.

  – ***Scrum***: A cada reunião de planejamento de um *sprint* o planejamento do projeto é revisado junto ao cliente, através dos requisitos restantes no *product backlog* (ALLIANCE, 2009). Nesta reunião o cliente seleciona os itens para o próximo *sprint* e o compromisso com o novo *sprint* é assumido por todos.

    Assim como no XP, o *Scrum* promove uma interação intensa entre o *product owner* (cliente) e a equipe, através das *daily scrum meeting*, *sprint planning* e *sprint review meeting*.

    Entretanto o *Scrum* não produz o Plano do Projeto solicitado no GPR10. Devido a falta deste plano **o *Scrum* contempla parcialmente este requisito**.

  – **FDD**: A etapa inicial "Desenvolver o Modelo Abrangente" é realizada por membros de domínio do negócio (cliente) por desenvolvedores e pelo arquiteto líder. Esta etapa serve de base pra construir a lista de funcionalidades, criando através desta o plano de desenvolvimento (COAD; LUCA; LEFEBVRE, 1999). Este é revisado pelo

cliente e a equipe, obtendo o comprometimento de ambos. Ao final de cada funcionalidade o plano de desenvolvimento é revisado e atualizado.

Entretanto o *Scrum* não produz o Plano do Projeto solicitado no GPR10. Devido a falta deste plano **o FDD contempla parcialmente este requisito**.

- GPR13 - O progresso do projeto é monitorado com relação ao estabelecido no Plano do Projeto e os resultados são documentados.

  Segundo o guia de implementação do Nível G do MR-MPS:

  > [...]Os resultados e os critérios de conclusão de cada tarefa são analisados, as entregas são avaliadas em relação às suas características (por meio de revisões e auditorias, por exemplo), a aderência ao cronograma e o dispêndio de esforços são examinados, bem como o uso dos recursos.
  > [...]O acompanhamento pode ser realizado utilizando-se ferramentas de planejamento, em que se pode examinar o previsto contra o realizado, usando-se indicadores de progresso e cumprimento de marcos, entre outros. O acompanhamento também pode ser feito por meio de reuniões e comunicação pessoal. Contudo, é importante ressaltar que devem existir registros desses acompanhamentos (SOFTEX MR-MPS, 2009, p.16).

  – **XP**: No XP o acompanhamento do projeto é realizado a todo momento através da presença do cliente no ambiente de desenvolvimento, das reuniões diárias e reuniões de planejamento das iterações (BECK, 1999). Esta última estabelece pontos de controle dentro da *release* para que programadores e clientes saibam como a execução das tarefas estão em relação ao planejado. Após cada iteração ocorre os testes de aceitação, que são realizados pelo cliente para validar os requisitos. Apesar de realizar o acompanhamento do projeto, o XP não evidencia o registro destes acompanhamentos e também não produz o Plano de Projeto como solicitado no GPR10. Portanto **o XP atende parcialmente este requisito**.

  – ***Scrum***: Os marcos no *Scrum* são representados pelos *sprints*, que são criados através das *Sprint Planning*. Assim como no XP, o *Scrum* monitora o projeto diariamente com as reuniões diárias. Há também o monitoramento ao final de cada *sprint*, onde a equipe apresenta o trabalho na reunião de revisão do *sprint* ao *product owner* através de uma demonstração. O *product owner* faz testes para verificar se cada item atende às suas expectativas e determinar se a meta foi atingida. O *Scrum* utiliza o gráfico *Burndown* para visualizar e registrar o acompanhamento do projeto, permitindo identificar os desvios entre o previsto e o realizado. Entretanto o *Scrum*

não produz o Plano de Projeto como solicitado no GPR10. Devido a falta deste Plano de Projeto **O *Scrum* atende parcialmente este requisito**.

- **FDD**: Os marcos para o projeto são definidos ao final da etapa "Planejar por Funcionalidade" onde é gerado o plano de desenvolvimento, contendo as datas para as entregas das funcionalidades. Ao final da etapa "Construir por Funcionalidade", são confrontadas as datas previstas e a realizada para cada funcionalidade. Não há no FDD sinalizações de registro deste acompanhamento e também o FDD não produz o Plano de Projeto como solicitado no GPR10. Desta forma **este requisito é atendido parcialmente pelo FDD.**

- GPR14 - O envolvimento das partes interessadas no projeto é gerenciado.

  Segundo o guia de implementação do Nível G do MR-MPS:

  > [...] Devem ser identificados os interessados relevantes no projeto, em que fases eles são importantes e como eles serão envolvidos (comunicações, revisões em marcos de projeto, comprometimentos, entre outros). Uma vez identificado e planejado o envolvimento, este deverá ser seguido. Os interessados no projeto podem incluir o cliente e o usuário (ou seus representantes), a direção da organização e os membros da equipe do projeto. Em projetos pequenos, estas atividades podem ser simplificadas devido ao pequeno número de interessados e à pouca comunicação necessária em função do curto prazo (SOFTEX MR-MPS, 2009, p.16).

  - **XP**: O XP possui como uma de suas práticas, manter o cliente sempre presente no ambiente de desenvolvimento. A interação com o cliente deve ser intensa (BECK, 1999). Na fase inicial e ao longo do projeto o cliente escreve as estórias de usuários e nas reuniões de planejamento de *releases* ele negocia com a equipe de desenvolvimento, priorizando as estórias a serem desenvolvidas. No XP o cliente é convidado a participar das reuniões diárias (TELES, 2004), acompanhando o progresso e fortalecendo o envolvimento com o projeto, ou seja, em projetos XP o envolvimento dos interessados no projeto é uma de suas principais características. Desta forma **o XP atende este requisito**.
  - ***Scrum***: Assim como no XP o *Scrum* também utiliza o cliente presente no dia-a-dia do projeto. O planejamento do envolvimento das pessoas no projeto é realizado, definindo as suas participações em etapas específicas do projeto. Na fase inicial do projeto o cliente (*product owner*) participa da criação do *product backlog* e posteriormente participa de reuniões de planejamento de *sprints* juntamente com a equipe do projeto. Ao final de cada sprint o cliente é convidado a participar de reuniões

de revisão de sprints com a equipe (ALLIANCE, 2009). No *Scrum* as etapas para o envolvimento dos interessados no projeto são bem definidas, deixando claro o momento em que estas pessoas são requeridas ao processo. Desta forma **o *Scrum* atende este requisito.**

- **FDD**: Nas cinco etapas de um projeto com FDD, o envolvimento das partes interessadas é planejado. Na etapa "Desenvolver o Modelo Abrangente" participam os membros do domínio do negócio (clientes) e desenvolvedores, sob a orientação de um modelador de objetos experiente. Na etapa "Construir a Lista de Funcionalidades" participam programadores líderes da equipe de modelagem da etapa anterior. Na etapa seguinte "Planejar por Funcionalidade" gerente de desenvolvimento e programadores líderes. Nas etapas "Detalhar por Funcionalidade" e "Construir por Funcionalidade" toda a equipe de desenvolvimento participa. Sendo assim **o FDD contempla este requisito.**

- GPR15 - Revisões são realizadas em marcos do projeto e conforme estabelecido no planejamento.

  Segundo o guia de implementação do Nível G do MR-MPS:

  > [...] Revisões em marcos do projeto não devem ser confundidas com o acompanhamento descrito em GPR13, que é o acompanhamento do dia-a-dia do projeto. Os marcos do projeto precisam, portanto, ser previamente definidos ao se realizar o planejamento do projeto (SOFTEX MR-MPS, 2009, p.17).

  - **XP**: O XP utiliza as iterações como os pontos de controle dentro das *releases* para que programadores e clientes saibam como a execução está em relação ao planejado. Uma iteração possui duração entre uma e três semanas (BECK, 1999). Desta forma **o XP contempla este requisito**.
  - ***Scrum***: Ao final de cada *sprint* são realizadas reuniões que objetivam revisar e avaliar os progressos ou atrasos no projeto em relação ao definido na *sprint planning*. Um *sprint* possui duração de duas a quatro semanas, ficando a critério do *Scrum Master* avaliar e definir a duração de um *sprint*, representando os pontos para as revisões do projeto. **O *Scrum* atende este requisito**.
  - **FDD**: Ao final da etapa "Planejar por Funcionalidade" é gerado o plano de desenvolvimento contendo as atividades de negócio com datas de término (mês e ano)

(COAD; LUCA; LEFEBVRE, 1999). Nestas datas são realizadas avaliações do progresso de cada atividade e do projeto em relação ao planejado. Desta forma **o FDD contempla estes requisito**.

- GPR16 - Registros de problemas identificados e o resultado da análise de questões pertinentes, incluindo dependências críticas, são estabelecidos e tratados com as partes interessadas.

  Segundo o guia de implementação do Nível G do MR-MPS:

  > [...] As atividades de revisão em marcos (GPR15) e de monitoramento (GPR13) do projeto possibilitam a identificação de problemas que estejam ocorrendo nos projetos. Estes problemas devem ser analisados e registrados, por exemplo, por meio de ferramentas específicas, planilhas ou outros tipos de mecanismos de gerenciamento de problemas (SOFTEX MR-MPS, 2009, p.17).

  – **XP**: No "Jogo do Planejamento" são identificadas e analisadas, com a participação do cliente e a equipe, as dependências e problemas relacionados às estórias selecionadas para a próxima *release*. Nas reuniões diárias são levantados, dentre outras coisas, os problemas encontrados no dia-a-dia do desenvolvimento do projeto. Uma das responsabilidades do *coach* é a remoção ou amortização de tais empecilhos.

    No entanto, não há práticas que evidenciam o registro de problemas identificados e análises realizadas sobre estes problemas. Desta forma **o XP atende parcialmente este requisito.**

  – ***Scrum***: A identificação e análise de problemas junto as partes interessadas acontece a todo momento no *Scrum*, por exemplo: nas reuniões de planejamento de *sprints*, reuniões diárias, reuniões de revisão de *sprints*, reuniões de retrospectiva de cada *sprint*. Em todas estas reuniões há a presença do *product owner* (cliente) e equipe.

    O *Scrum* registra todos os problemas encontrados num artefato chamado de *impediment backlog*, onde a manutenção e ações dirigidas a estes problemas é de responsabilidade do *Scrum Master* (ALLIANCE, 2009). **O *Scrum* contempla este requisito**.

  – **FDD**: Os dependências no FDD são detectadas na etapa "Planejar por Funcionalidade", gerando o plano de desenvolvimento afirmado pela equipe e o cliente contendo, dentre outras coisas, as dependências para cada funcionalidade (COAD; LUCA; LEFEBVRE, 1999).

Em relação a identificação e análise de problemas o FDD atua apenas na identificação e antecipação de funcionalidades consideradas críticas. Não há práticas que evidenciam o registro e análise de problemas junto ao cliente durante o projeto. **O FDD atende parcialmente este requisito**.

- GPR17 - Ações para corrigir desvios em relação ao planejado e para prevenir a repetição dos problemas identificados são estabelecidas, implementadas e acompanhadas até a sua conclusão.

  Segundo o guia de implementação do Nível G do MR-MPS:

  > [...] Como resultado do acompanhamento do projeto (GPR13) e das revisões em marcos (GPR16), problemas são identificados, analisados e registrados (GPR16). Ações corretivas devem ser estabelecidas para resolver problemas que possam impedir o projeto de atingir seus objetivos se não forem resolvidos de forma adequada. As ações corretivas definidas devem ser gerenciadas até serem concluídas. O controle dos problemas levantados, as ações tomadas, os responsáveis pelas ações e os resultados devem ser registrados (SOFTEX MR-MPS, 2009, p.17).

  – **XP**: No XP as reuniões diárias são utilizadas, dentre outras coisas, para identificar possíveis problemas (dependências, atrasos, riscos, etc.) o mais breve possível (BECK, 1999), permitindo ao *coach* aplicar medidas corretivas. No entanto não há práticas que evidenciam o registro desses problemas, nem das ações corretivas aplicadas. Desta forma **o XP atende parcialmente este requisito**.

  – ***Scrum***: As reuniões diárias buscam, dentre outras coisas, levantar impedimentos (problemas, dependências, riscos, etc.). Logo há uma identificação, tomada de ações e monitoramento dos impedimentos pelo *Scrum Master*. Entretanto não há registro de planos de ações corretivas, nem de documentação relacionada a isso. Portanto **o *Scrum* atende parcialmente este requisito.**

  – **FDD**: Ao final da etapa "Planejar por Funcionalidade" é gerado o plano de desenvolvimento contendo as atividades de negócio com datas de término (mês e ano). Nestas datas são realizadas avaliações do progresso de cada atividade e do projeto em relação ao planejado, permitindo identificar possíveis problema (atrasos, dependências, riscos, etc.). É de responsabilidade do gerente de projeto aplicar medidas que sanem esses problemas (COAD; LUCA; LEFEBVRE, 1999). Entretanto não há registro das ações corretivas aplicadas. Sendo assim **o FDD atende parcialmente este requisito**.

3.2 Gerência de Requisitos (GRE) X MAs

O propósito do processo Gerência de Requisitos é gerenciar os requisitos do produto e dos componentes do produto do projeto e identificar inconsistências entre os requisitos, o plano do projeto e os produtos de trabalho do projeto.

Nesta seção é realizado o mapeamento das MAs em relação ao processo Gerência de Requisitos, descrito no guia de implementação do MPS.BR.

- GRE1 - O entendimento dos requisitos é obtido junto aos fornecedores de requisitos.

  Segundo o guia de implementação do Nível G do MR-MPS:

  > [...] O objetivo deste resultado é garantir que os requisitos estejam claramente definidos a partir do entendimento dos requisitos realizado junto aos fornecedores de requisitos. Informações sobre esses fornecedores podem ser identificadas no plano do projeto, bem como informações sobre como será a comunicação com eles. Essas comunicações devem ser registradas formalmente em atas, e-mails, ferramentas de comunicação ou outros meios. Como comprovação do entendimento dos requisitos, deve-se ter um documento de requisitos, que pode ter diferentes formas de acordo com as necessidades da organização, por exemplo, o entendimento dos requisitos pode ser registrado na forma de uma lista de requisitos, especificações de casos de uso ou detalhados conforme uma metodologia própria da organização, entre outras formas.
  > Após a identificação dos requisitos do produto e dos componentes do produto do projeto, é necessário garantir que os requisitos propostos atendem às necessidades e expectativas do cliente e dos usuários. Para tanto, os requisitos identificados devem ser avaliados com base em um conjunto de critérios objetivos, previamente estabelecidos. Alguns exemplos de critérios são: possuir identificação única; estar claro e apropriadamente declarado; não ser ambíguo; ser relevante; ser completo; estar consistente com os demais requisitos; ser implementável, testável e rastreável.
  > [...] Após a avaliação dos requisitos, um registro de aceite dos requisitos deve ser obtido pelos fornecedores de requisitos (SOFTEX MR-MPS, 2009, p.20).

  – **XP**: Os requisitos no XP são documentados em cartões de estórias, onde estas estórias são escritas pelo próprio cliente. O entendimento dos requisitos ocorre nas reuniões de planejamento das iterações. Nesta reunião o cliente seleciona as estórias que serão desenvolvidas na iteração em questão e esclarece dúvidas restantes sobre as estórias selecionadas aos membros da equipe, que posteriormente estimam o tempo de desenvolvimento para cada estória.

O XP não possui critérios objetivos para realizar avaliação da identificação de cada requisito. Em contrapartida a prática "cliente presente" fornece à equipe respostas rápidas sobre questões pertinentes às estórias.

O XP não possui práticas que evidenciam o registro das reuniões realizadas com o cliente, exceto o registro de alguma informação pertinente à estórias, onde estas informações são registradas nos próprios cartões de estórias.

Devido a falta de registro completo nas reuniões para o levantamento de requisitos e de não possuir critérios objetivos para a avaliação dos requisitos, **o XP atende parcialmente este requisito**.

– ***Scrum***: No *Scrum* os requisitos são obtidos diretamente com o cliente. Os requisitos são discutidos pelos participantes do projeto e adicionados ao *Product Backlog*. Após discutir e entender os requisitos junto ao cliente, a equipe atribui a estes requisitos: descrições, tempo e a definição de quem irá desenvolver. A cada *sprint* um conjunto de tarefas são selecionadas para serem desenvolvidas. Este conjunto de requisitos é representado pelo *sprint backlog*.

  O *Scrum* não estabelece critérios objetivos para a avaliação dos requisitos. No entanto o *Scrum* utiliza-se da prática "Cliente Presente" para avaliar o desenvolvimento do requisito a todo momento.

  Nas reuniões com o cliente para o levantamento de requisitos, não há práticas que evidenciam o registro de toda a reunião, como por exemplo: o uso de atas, exceto pelo registro dos próprios requisitos informados pelo cliente, sendo estes registrados no modelo de dados.

  Devido a falta de registro completo nas reuniões para o levantamento de requisitos e de não possuir critérios objetivos para a avaliação dos requisitos. **O Scrum atende parcialmente este requisito**.

– **FDD**: Na etapa "Desenvolver o Modelo Abrangente" realiza-se, junto ao cliente, um estudo dirigido sobre o escopo do sistema e seu contexto. Em seguida são realizados estudos mais detalhados sobre o domínio do negócio, também com o cliente, para cada área a ser modelada. Após cada estudo dirigido sobre o domínio são criados modelos para cada um dos domínios identificados. Na etapa seguinte é construída a lista de funcionalidades, representando o documento de requisitos, que utiliza como base o modelo abrangente definido na etapa anterior. Ao final desta etapa é realizada uma auto-avaliação ou uma avaliação interna através da participação ativa dos membros da equipe de modelagem e especialistas no domínio do negócio (cliente), com o objetivo de confirmar e/ou esclarecer as questões que afetam a lista de fun-

cionalidades.

O FDD não evidencia práticas que estabeleçam critérios para a avaliação dos requisitos, apesar dessa avaliação ocorrer.

Devido a falta de registro completo nas reuniões para o levantamento de requisitos e de não possuir critérios objetivos para a avaliação dos requisitos. **O FDD atende parcialmente este requisito**.

- GRE2 - Os requisitos de software são aprovados utilizando critérios objetivos.

  Segundo o guia de implementação do Nível G do MR-MPS:

  > [...] Após a aprovação dos requisitos, um comprometimento formal da equipe técnica com os requisitos aprovados deve ser obtido e registrado, por exemplo, na forma de ata de reunião, e-mail ou algum outro mecanismo. Mudanças de requisitos aprovados pelos fornecedores de requisitos podem afetar compromissos já estabelecidos pela equipe técnica. Nestes casos, um novo comprometimento da equipe técnica com os requisitos modificados deve ser obtido e registrado após os requisitos modificados terem sido novamente aprovados a partir de critérios estabelecidos conforme GRE 1 (SOFTEX MR-MPS, 2009, p.21).

  – **XP**: No XP a aprovação dos requisitos (estórias) é realizada durante o "Jogo do Planejamento", onde a equipe obtém as informações complementares sobre os requisitos. Ao final desta prática todas as estórias abordadas são estimadas e aprovadas pela a equipe, ficando ao critério do cliente selecionar as estórias que serão desenvolvidas na próxima iteração.

    No entanto não há no XP registro nem critérios objetivos sobre a aprovação dos requisitos pela equipe. Desta forma **o XP atende parcialmente este requisito.**

  – ***Scrum***: No *Scrum* após o levantamento inicial dos requisitos, estes são adicionados ao *Product Backlog*. Os itens do *Product Backlog* são analisados pelo cliente e por todos os membros da equipe. Na reunião de planejamento de um *sprint* o cliente seleciona os requisitos que serão implementados no próximo *sprint*. A partir desta seleção a equipe esclarece algumas dúvidas restantes em relação aos itens selecionados estimando horas aos mesmos, estes itens dão origem ao *Sprint Backlog*. A aprovação dos requisitos ocorre de maneira informal durante a reunião de planejamento, ou seja, os itens presentes no *Sprint Backlog* passaram pela aprovação tanto da equipe quanto do próprio cliente.

    No entanto não há no *Scrum* registro nem critérios objetivos sobre a aprovação dos requisitos pela equipe. Desta forma **o *Scrum* atende parcialmente este requisito.**

  - **FDD**: No FDD a aprovação das funcionalidades pela equipe técnica ocorre na etapa "Planejar por Funcionalidade", gerando o Plano de Desenvolvimento revisado pelo cliente e pelos membros da equipe contendo todas as funcionalidades com datas de início e término.

    O FDD não possui critérios objetivos para a aprovação das funcionalidades. Desta forma **o FDD atende parcialmente este requisito**.

- GRE3 - A rastreabilidade bidirecional entre os requisitos e os produtos de trabalho é estabelecida e mantida.

  Segundo o guia de implementação do Nível G do MR-MPS:

  > [...] Este resultado indica a necessidade de se estabelecer um mecanismo que permita rastrear a dependência entre os requisitos e os produtos de trabalho.
  > [...] É importante ressaltar que este resultado estabelece a criação de um sistema de rastreamento e que não necessariamente envolve a criação de uma matriz de rastreabilidade específica para atendimento ao resultado esperado (SOFTEX MR-MPS, 2009, p.21).

  - **XP**: O XP não possui práticas que estabeleçam a rastreabilidade de requisitos e não implementa um sistema de acompanhamento de requisitos, em relação a rastreabilidade. Sendo assim **o XP não contempla este requisito**.
  - ***Scrum***: O *Scrum* não menciona a relação dos requisitos entre si ou os requisitos aos modelos resultantes do projeto. O *Scrum* não possui práticas que estabeleçam a rastreabilidade de requisitos e não implementa um sistema de acompanhamento de requisitos. Sendo assim **o *Scrum* não contempla este requisito**.
  - **FDD**: O FDD não evidencia práticas que estabeleçam a rastreabilidade de requisitos e não implementa um sistema de acompanhamento de requisitos. Sendo assim **o FDD não contempla este requisito**.

- GRE4 - Revisões em planos e produtos de trabalho do projeto são realizadas visando identificar e corrigir inconsistências em relação aos requisitos.

  Segundo o guia de implementação do Nível G do MR-MPS:

  > [...] A consistência entre os requisitos e os produtos de trabalho do projeto deve ser avaliada e os problemas identificados devem ser corrigidos. Este resultado sugere, portanto, a realização de revisões ou de algum mecanismo equivalente para identificar inconsistências entre os requisitos e os demais elementos do projeto como, por exemplo, planos,

> atividades e produtos de trabalho. As inconsistências identificadas devem ser registradas e ações corretivas executadas a fim de resolvê-las (SOFTEX MR-MPS, 2009, p.21).

- **XP**: Na metodologia XP as revisões entre o que foi solicitado e o que está sendo desenvolvido acontecem a todo momento, devido a presença do cliente no ambiente de desenvolvimento. Nestas revisões o cliente verifica se a estória desenvolvida está inconsistente com sua descrição. Neste caso o cliente esclarece o funcionamento desejado de tal estória, caso seja necessário o cartão desta estória é atualizado com os dados atualizados. A partir deste momento o desenvolvedor reavalia, caso seja necessário, e corrige o requisito.

  Outro ponto para avaliação das estórias desenvolvidas ocorrem a todo momento através dos testes unitários e ao final de cada iteração, onde é solicitado ao cliente a realização do teste de aceitação.

  No entanto o XP não possui práticas que evidenciam o registro sobre inconsistências detectadas. Portanto **o XP atende parcialmente este requisito**.

- ***Scrum***: O *Scrum*, assim como o XP, baseia-se em princípios onde os requisitos são poucos estáveis, as prováveis inconsistências são verificadas pela equipe durante o *sprint*, através das reuniões diárias e do *Sprint Review*. As ações corretivas são realizadas no *Sprint Review* em consonância com o *Scrum Master* e o Cliente.

  O Scrum não determina explicitamente a geração de documentação das inconsistências do projeto. As ações corretivas são executadas e seus resultados (produtos de trabalho) são incrementados ao resultado final do *sprint*.

  Devido a falta de registro **o *Scrum* atende parcialmente este requisito**.

- **FDD**: A etapa "Construir por Funcionalidade" possui atividades como: inspeção de código e testes de unidade, possibilitando identificar inconsistências. Esta mesma etapa possui como critério de saída uma ou mais funcionalidades de valor para o cliente (COAD; LUCA; LEFEBVRE, 1999), sendo estas funcionalidades incorporadas a versão atual, permitindo ao cliente realizar testes e detectar inconsistências em relação ao solicitado para as funcionalidades.

  No entanto o FDD não possui práticas que evidenciam o registro sobre inconsistências detectadas. Sendo assim **o FDD atende parcialmente este requisito**.

- GRE5 - Mudanças nos requisitos são gerenciadas ao longo do projeto.

  Segundo o guia de implementação do Nível G do MR-MPS:

> [...] As necessidades de mudanças devem ser registradas e um histórico das decisões acerca dos requisitos deve estar disponível. Estas decisões são tomadas por meio da realização de análises de impacto da mudança no projeto e podem incluir aspectos como: influência em outros requisitos, expectativa dos interessados, esforço, cronograma, riscos e custo (SOFTEX MR-MPS, 2009, p.22).

– **XP**: O XP utiliza do ambiente colaborativo entre a equipe e o cliente para que, dentre outras coisas, o cliente aprenda cada vez mais sobre o sistema que deseja (TELES, 2004). Esse aprendizado resulta em mudanças nas estórias, objetivando certificar a necessidade do cliente.

> [...] A maioria dos métodos ágeis expressa os requisitos em termos de histórias informais ajustáveis. Métodos ágeis contam com seus ciclos rápidos de iterações para determinar as mudanças necessárias nas funcionalidades desejadas e para corrigi-las na iteração seguinte. (BOEHM; TURNER, 2003, p.37).

As mudanças de requisitos são tratadas com muita naturalidade em XP. Sendo estas alterações informadas nas reuniões diárias ou no "Jogo do Planejamento", possibilitando a equipe realizar um novo planejamento para a iteração.

No entanto o XP não evidencia o registro das mudanças de requisitos, como também não mantém o histórico das decisões sobre alterações de requisitos.

Portanto devido a falta destes registros, **o XP atende parcialmente este requisito**.

– ***Scrum***: O ciclo de vida do *Scrum*, assim como o do XP, favorece um processo de aprendizagem em relação ao sistema, ou seja, a colaboração entre cliente e a equipe *Scrum*.

Todo processo de gerenciamento de requisitos no *Scrum* é controlado durante o *sprint*. A cada novo *sprint*, o cliente tem a chance de alterar a prioridade, adicionar novas tarefas ou alterar tarefas existentes para o *sprint*. Além disso o cliente acompanha o desenvolvimento do *sprint* e esclarece eventuais dúvidas, sem poder mudar o escopo do *sprint* porém, caso realmente precise mudar os itens do *backlog* selecionado, ele tem a opção de cancelar o *sprint*. Neste caso, a iteração volta ao início realizando uma nova reunião de planejamento de *sprint*.

O *Scrum* não evidencia o registro das mudanças de requisitos, como também não mantém o histórico das decisões sobre alterações de requisitos.

Portanto devido a falta destes registros, **o *Scrum* atende parcialmente este requisito**.

– **FDD**: As alterações de requisitos no FDD podem ocorrer em qualquer etapa do projeto. Neste caso o Plano de Desenvolvimento é alterado, contendo as novas datas referentes as mudanças realizadas nos requisitos.

  O FDD não evidencia o registro das mudanças de requisitos, como também não mantém o histórico das decisões sobre alterações de requisitos.

  Portanto devido a falta destes registros, **o FDD atende parcialmente este requisito**.

4 Análise dos Resultados

Este capítulo apresenta os resultados relacionados ao mapeamento realizado no capítulo 3. Estes resultados foram sintetizados em duas tabelas, uma referente ao mapeamento do GPR e a outra do GRE. Estas tabelas estão nos apêndices A e B nas páginas 76 e 78 respectivamente. A partir destas tabelas foram gerados gráficos evidenciando percentualmente a conformidade entre os processos Gerência de Projetos (GPR) e Gerência de Requisitos (GRE) do nível G do MPS.BR e as metodologias ágeis XP, Scrum e FDD.

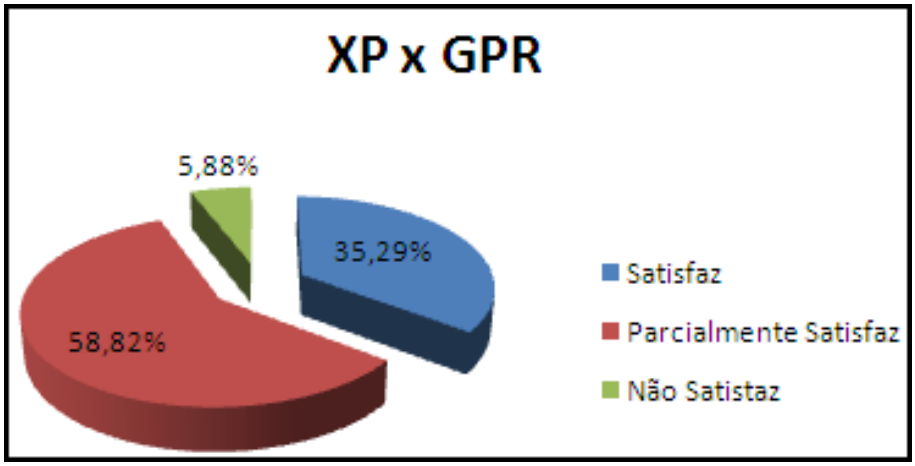

Figura 5: XP X GPR

Os gráficos referentes aos resultados do mapeamento do GPR, em relação as metodologias, são representados pelas figuras 5, 6 e 7. É possível notar que as informações presentes nestes gráficos refletem as características de cada metodologia. Segundo o mapeamento realizado, o Scrum satisfaz 52,94%, o XP e FDD 35,29% dos requisitos solicitados pelo processo GRP. Este resultado pode ser associado ao foco de cada metodologia. No caso do Scrum que é um processo de desenvolvimento voltado para a gerência do projeto (ALLIANCE, 2009), proporcionando um maior controle sobre o mesmo, é justificável uma maior adequação. O resultado visualizado nas figuras 5 e 7, para o XP e FDD, pode ser uma evidência de que estas duas são metodologias concentram seus esforços no dia a dia do desenvolvimento, justificando assim uma menor adequação em relação ao GPR.

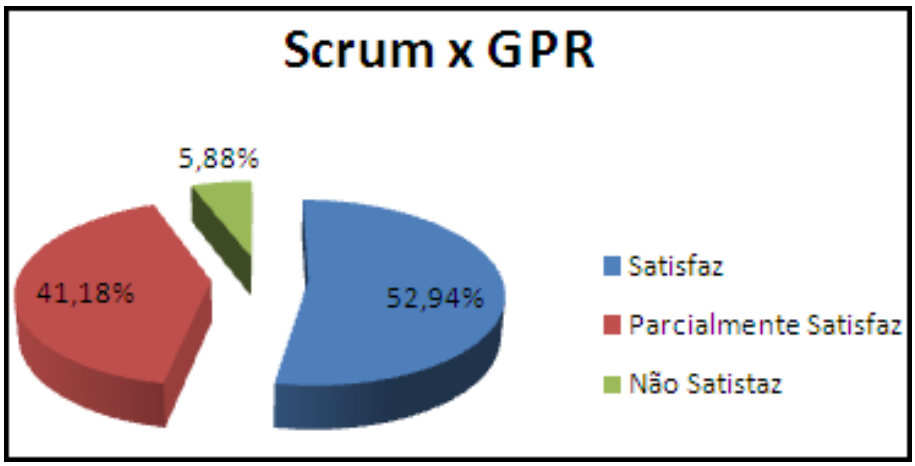

Figura 6: Scrum X GPR

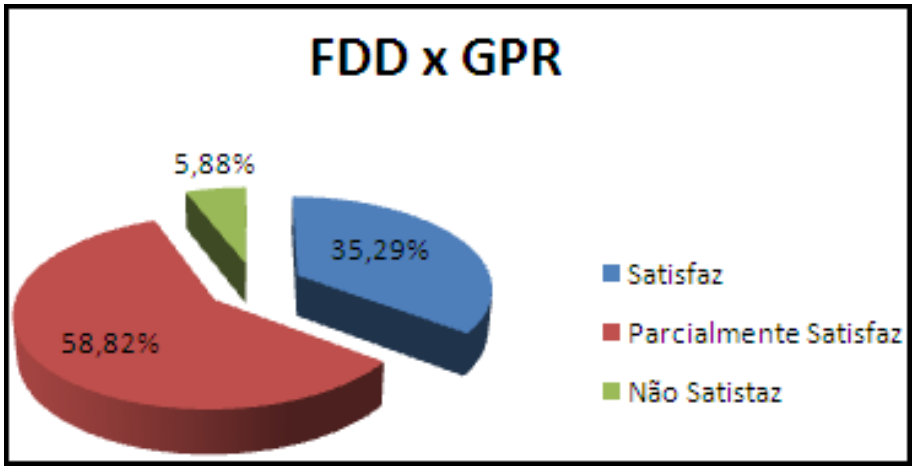

Figura 7: FDD X GPR

Outra observação pertinente está relacionada aos requisitos classificados como "Parcialmente Satisfeito". Foram analisados os motivos pelos quais estes foram assim classificados. O resultado desta análise encontra-se na tabela no apêndice C, está tabela é representada através dos gráficos das figuras 8, 9 e 10.

Pelos gráficos representados pelas figuras 8, 9 e 10, pode-se observar que a classificação da maioria dos requisitos classificados como "Parcialmente Satisfeito" ocorre devido ao fato de não haver registro sobre alguns elementos das práticas realizadas pelas metodologias. Por exemplo: no mapeamento do GPR6[1], o XP atende parcialmente pois não evidencia o registro (FR) de identificação e acompanhamento de riscos, apesar de identificar e acompanhar os riscos do projeto.

[1]Os riscos do projeto são identificados e o seu impacto, probabilidade de ocorrência e prioridade de tratamento são determinados e documentados

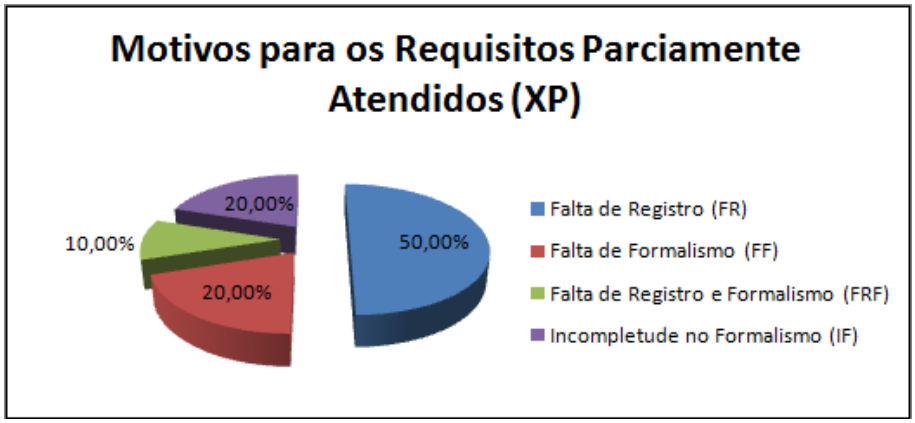

Figura 8: XP: Motivos para a classificação do XP em relação do GPR

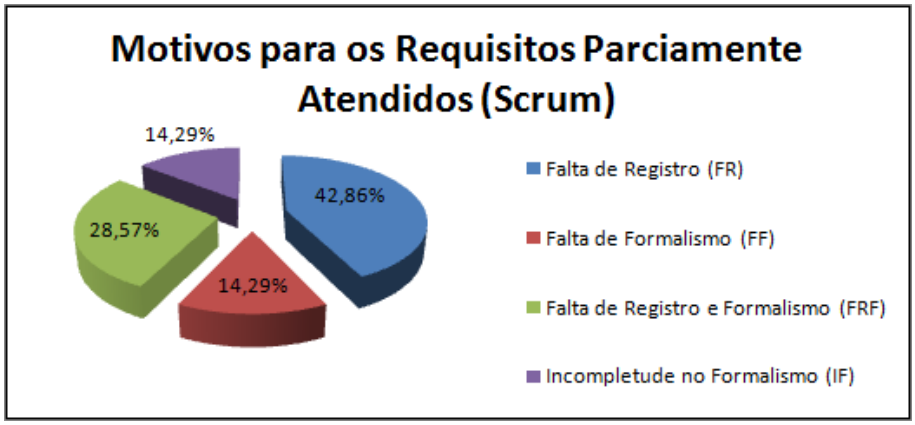

Figura 9: Scrum: Motivos para a classificação do Scrum em relação do GPR

Outro fator que proporcionou a classificação como "Parcialmente Satisfeito" foi a falta de formalismo (FF), ou seja, em alguns casos é solicitado que o processo de desenvolvimento possua critérios objetivos para desempenhar determinada tarefa, o que não ocorre nas MAs, sendo estes critérios em alguns casos ausentes ou subjetivos. Por exemplo: no GPR7[2] o Scrum não evidencia práticas que estabeleçam critérios para a seleção dos membros da equipe (exigido pelo requisito), ficando na responsabilidade do *Scrum Master* realizar a seleção. Outro exemplo ocorre em XP no GPR2[3], onde é necessário a utilização de um método objetivo para a decomposição das estórias, sendo que esta decomposição ocorre informalmente, através da experiência da equipe em projetos anteriores. O XP, no GPR1[4], é classificado como "Parcialmente Satisfeito" pois há uma incompletude no formalismo (IF) empregado, uma vez que a metodologia define na fase inicial o escopo da primeira release mas não de todo o projeto.

[2]Os recursos humanos para o projeto são planejados considerando o perfil e o conhecimento necessários para executá-lo

[3]As tarefas e os produtos de trabalho do projeto são dimensionados utilizando métodos apropriados

[4]O escopo do trabalho para o projeto é definido

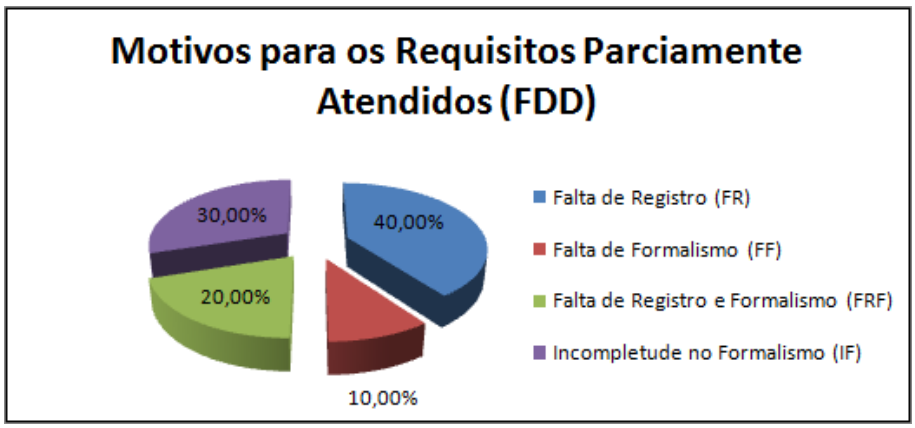

Figura 10: FDD:Motivos para a classificação do FDD em relação do GPR

Dos 17 requisitos do processo GPR do nível G do MPS.BR, o único foi classificado como "Não Satisfeito" foi o GPR10[5]. A não contemplação deste requisito por parte das MAs analisadas se deve ao fato destas não possuírem práticas que reúnam os documentos, referentes a planejamentos confeccionados, em um único documento. Além do fato de que as 3 MAs não confeccionam todos os documentos solicitados. Por outro lado, se não for considerado a reunião destes documentos e sim a criação deles, as 3 MAs analisadas seriam classificadas como "Parcialmente Satisfeito", pois todas elas criam pelo menos um artefato que representa o planejamento do projeto, por exemplo, o XP elabora o planejamento da *release* tendo como base para o planejamento a seleção das estórias que farão parte da *release*.

Segundo o mapeamento realizado o Scrum vai mais além em relação ao XP. No *Product Backlog* contém recursos necessários para a infra-estrutura, as atividades com estimativas primárias (alto nível) e planejamento para todo o projeto. O Scrum também confecciona uma lista contendo todos os impedimentos identificados. O FDD na etapa "Planejar por Funcionalidade" possui como critério de saída o plano de desenvolvimento contendo as datas de entregas para cada funcionalidade. Contudo foi considerado a reunião e não criação de tais documentos, devido a esse critério adotado as 3 MAs analisadas foram classificadas como "Não Satisfeito".

XP e FDD possuem a mesma porcentagem de adequação ao GPR, o que não implica que a classificação seja igual para todos os requisitos. Segundo o mapeamento realizado o XP e o FDD possuem classificações diferentes em 2 requisitos: o GPR1 (definição de escopo) e GPR4 (estimativas de esforço e custo). No GPR1 o XP foi classificado com "Parcialmente Satisfeito", pois, como foi informado, o XP gerencia escopo baseado das *releases* através de estórias e não do projeto inteiro, como solicitado neste requisito. Já o FDD cria a lista de funcionalidades de todo o projeto, satisfazendo o GPR1. O XP no GPR4 foi classificado como

[5]Planos para a execução do projeto são estabelecidos e reunidos no Plano do Projeto

"Satisfeito" pois realiza as estimativas e custos para as estórias. Em contrapartida o FDD realiza as estimativas das funcionalidades, mas não evidencia o custo destas, sendo assim classificado como "Parcialmente Satisfeito".

O mapeamento realizado para o processo GRE evidenciou uma igualdade entre os resultados de cada metodologia. O gráfico gerado pela tabela contendo a síntese do resultado do mapeamento do processo GRE (apêndice B), é representado pela figura 11. Segundo o mapeamento, nenhum dos requisitos foi classificado como "Satisfeito", sendo que 4 dos 5 requisitos foram classificados como "Parcialmente Satisfeito" e o restante como "Não Satisfeito". A classificação das MAs em relação ao GRE foram idênticas.

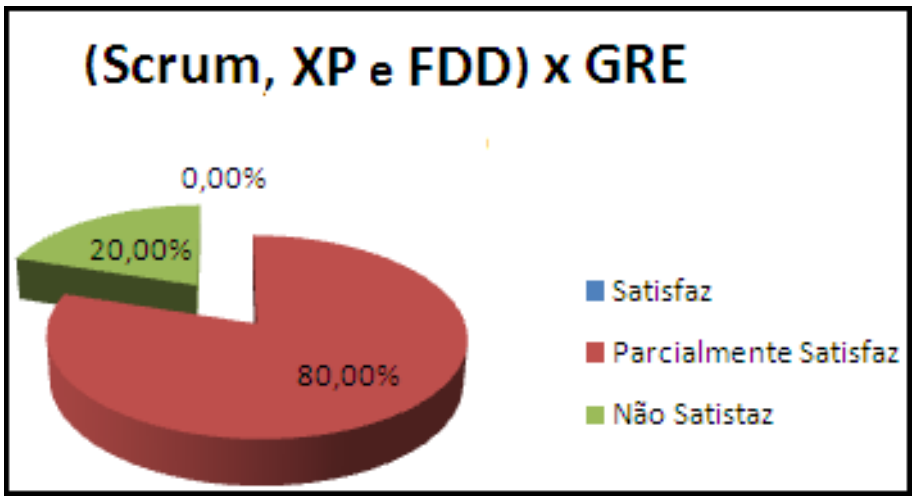

Figura 11: (Scrum, XP e FDD) X GRE

Assim como foi analisado o GPR, analisou-se também os motivos que resultaram na classificação realizada para o mapeamento dos requisitos do GRE (ver apêndice B) em relação as MAs. Este resultado encontra-se na tabela do apêndice D que é representada através do gráfico da figura 12.

Através da análise dos resultados do mapeamento referentes aos requisitos classificados como "Parcialmente Satisfeito", pôde-se perceber que o principal motivo para esta classificação se deve a falta de registro (FR) sobre os resultados esperados nos requisitos GRE1, GRE2, GRE4 e GRE5. Por exemplo, no GRE5[6] é solicitado a manutenção de um histórico para as decisões referentes as alterações de requisitos. As 3 metodologias ágeis analisados favorecem a mudança de requisitos ao longo do projeto, pois se tratam de processos adaptativos. Como foi demonstrado no mapeamento do GRE5, estas metodologias não praticam o registro sobre as modificações e históricos de decisões. Desta forma o motivo da classificação do GRE5 como

[6]Mudanças nos requisitos são gerenciadas ao longo do projeto

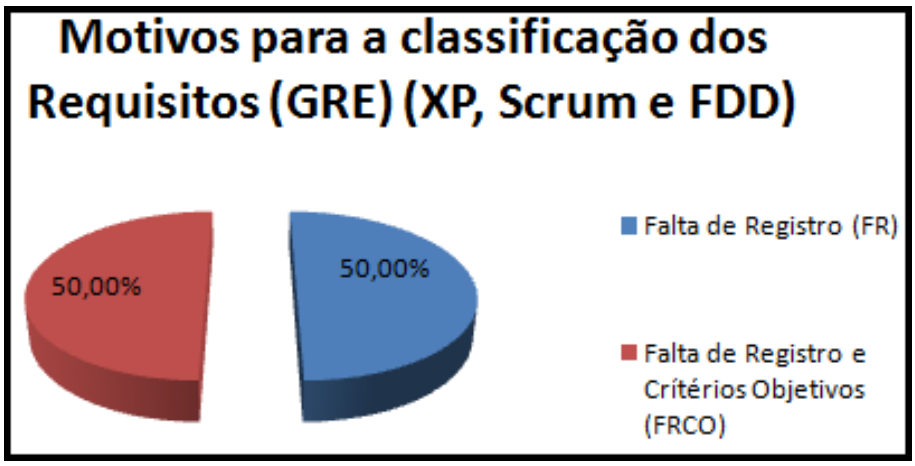

Figura 12: Motivos para a classificação do XP, Scrum e FDD em relação do GRE

"Parcialmente Satisfeito", deve-se a falta dos registros solicitados.

Além da falta de registro, os requisitos GRE1 e GRE2 foram classificados como "Parcialmente Satisfeito" pois também não utilizam critérios objetivos (FRCO) sobre alguns elementos solicitados no resultados esperado. No caso GRE2[7], os motivos para a sua classificação como "Parcialmente Satisfeito" foram a falta de registro e a falta de critérios objetivos, requeridos para este requisito. No primeiro caso é solicitado um registro formal sobre o comprometimento da equipe técnica em relação aos requisitos aprovados. Como foi discutido no mapeamento do requisito em questão, as 3 metodologias não realizam o registro formal deste comprometimento. No segundo caso (falta de critérios), é solicitado a utilização da aprovação dos requisitos através de critérios objetivos estabelecidos no GPR1. Segundo o mapeamento deste requisito, nenhuma das 3 metodologias utilizam critérios objetivos para a aprovação dos requisitos. Esta aprovação ocorre de forma dinâmica, juntamente com o cliente.

Dos 5 requisitos solicitados pelo processo GRE, o único requisito classificado como "Não Satisfeito", foi o GRE3 (Rastreabilidade bidirecional entre os requisitos e os produtos de trabalho). Nenhuma das 3 MAs analisadas possuem práticas que realizam esta rastreabilidade, mostrado nos gráficos acima como "Não Satisfaz".

Outro ponto a ser observado é a possibilidade de combinação dos processos ágeis. Como já foi citado, o Scrum é um processo com o foco no controle do projeto. O XP e FDD focados no dia a dia do desenvolvimento. Desta forma pode ser bastante viável a combinação do Scrum com XP ou Scrum com FDD. Com essa combinação o processo de desenvolvimento resultante estará voltado tanto para o controle quanto para os detalhes do dia a dia do desenvolvimento. As gráficos representados pelas figuras 13 e 14 mostram o grau de adequação a partir

[7]Os requisitos de software são aprovados utilizando critérios objetivos

das combinações comentadas acima. Vale ressaltar que estas combinações foram realizadas desconsiderando possíveis interferências que práticas de uma metodologia poderá ocasionar nas práticas de outras metodologias, ou seja, é realizada apenas uma análise fria sobre o resultado das combinações das metodologias.

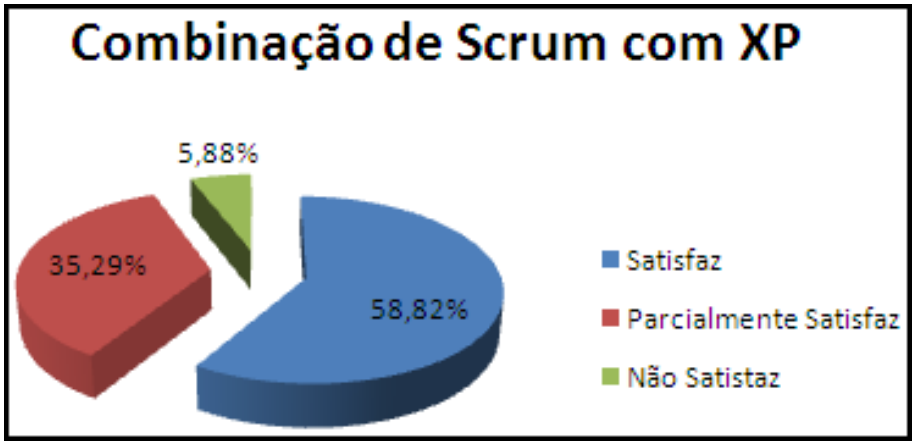

Figura 13: Grau de adequação ao processo GPR resultante da combinação entre Scrum e XP

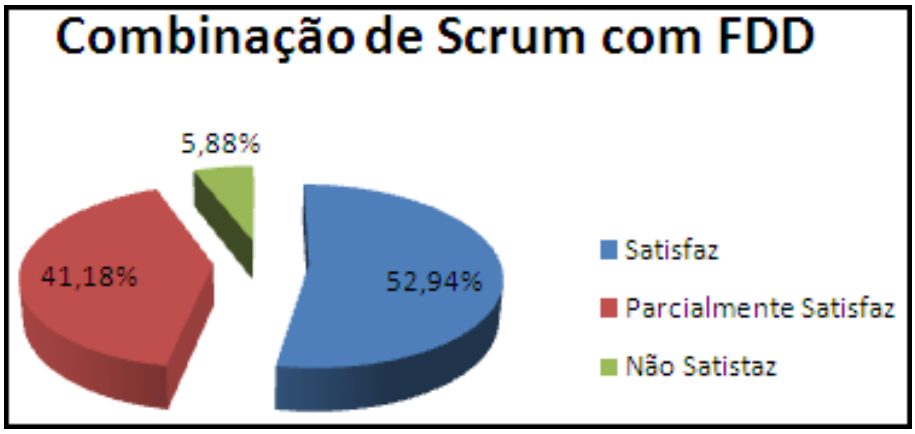

Figura 14: Grau de adequação ao processo GPR resultante da combinação entre Scrum e FDD

Com a comparação dos resultados das MAs após a combinação, pôde-se perceber que, apenas a combinação do Scrum com o XP obteve uma melhora, passando de 52,94% para 58,82%. Esse aumento é representado pela alteração da classificação do requisito GPR4 [8] para "Satisfeito", que segundo o mapeamento realizado, o XP satisfaz e o Scrum satisfaz parcialmente. Os resultados obtidos após a combinação do Scrum com o FDD continuaram os mesmos.

Vale ressaltar que apesar da pouca alteração ocorrida no resultado do mapeamento não implica dizer que a combinação dessas metodologias não seja interessante, pois estas podem se relacionar contemplando ambientes distintos num projeto.

[8] O esforço e o custo para a execução das tarefas e dos produtos de trabalho são estimados com base em dados históricos ou referências técnicas

Conclusão e Trabalhos Futuros

Neste trabalho foi realizado o mapeamento entre as MAs (XP, Scrum e FDD) com o nível G do MPS.BR, discutindo o **como** estas metodologias atendem ou não os resultados esperados pelo guia de implementação do MPS.BR. Foi apresentada de forma visual a adequação das MAs em relação ao nível G do MPS.BR. Para isto foram utilizadas tabelas e gráficos, contendo os resultados condensados e os percentuais de conformidades entre as MAs e o modelo de qualidade em questão, respectivamente.

É importante registrar que o mapeamento realizado é passível de interpretações. Para deixar claro os motivos para a classificação dos requisitos foi seguida uma metodologia, exposta no capítulo 2. Mas como os elementos das metodologias ágeis podem sofrer alterações/customizações e possuir interpretações, o mesmo pode ocorrer com o resultado do mapeamento.

Este trabalho pode servir como referência para profissionais que trabalham com desenvolvimento de *software*. Este trabalho procurou ampliar a visibilidade da relação entre as metodologias ágeis e o modelo MPS.BR. Para que empresas que utilizam métodos ágeis e almejam um selo de qualidade de *software*, detectar os pontos fortes e fracos das metodologias que utilizam (ou pretendem utilizar) em relação ao nível G do MPS.BR.

É importante ressaltar que as metodologias ágeis não são utilizadas em sua totalidade (BECK; ANDRES, 2004), geralmente são utilizadas práticas que se adequam as necessidades e limitações de uma determinada empresa. Desta forma uma possibilidade para um trabalho futuro, é a pesquisa das práticas e sua customizações mais utilizadas, a fim de realizar o mapeamento destas em relação aos modelos de qualidade.

O mapeamento realizado pode dar subsídio a realização de outros trabalhos como por exemplo: ampliar a discussão para os outros níveis do MPS.BR, aumentando a visibilidade das relações entre as metodologias ágeis e o modelo de qualidade de *software* do Brasil. Realizar o mapeamento para as áreas de processos do CMMI em seus cinco níveis, obtendo o conhecimento da distância entre o CMMI e o mundo ágil. Realizar o mapeamento das metodologias ágeis em relação MPS.BR, levando em consideração o processo e o Método de Avaliação MA-MPS que é responsável por verificar a maturidade da unidade organizacional na execução de seus processos de software. Desenvolver uma ferramenta que auxilie as empresas que utilizam métodos ágeis a obter um selo de qualidade de *software*, tanto do MPS.BR como do CMMI.

Esta ferramenta irá ajudar em situações que requisitos solicitem o registro do planejamento e armazenamento de informações pertinentes ao projeto e fornecer funcionalidades de acordo com as características das metodologias ágeis.

Uma pesquisa sobre a combinação das metodologias ágeis pode ser realizada, de modo a verificar as possíveis interferências que podem ocorrer com sobreposição de suas práticas. A partir dessa combinação um conjunto de práticas poderão ser selecionadas afim de contemplar mais áreas do projeto, como por exemplo a combinação do SCRUM com XP, abrangendo a gerência de projetos e a fase de implementação.

Referências

ABRAHAMSSON, P. *et al. Agile software development methods - Review and analysis*. [S.l.], 2002.

ALLIANCE, S. *Scrum Alliance*. 2009. Disponível em: <http://www.scrumalliance.org>. Acesso em: 05 Jul. 2009.

ANDERSON, A. *et al.* Chrysler goes to extremes. In: . [S.l.]: Distributed Computing, 1998. p. 24–28.

BECK, K. *Smalltalk Best Practice Patterns*. [S.l.]: Prentice Hall, 1997.

BECK, K. *Extreme Programming Explained: Embrace Change*. 1st. ed. [S.l.]: Addison-Wesley Professional, 1999. 224 p. ISBN 0201616416.

BECK, K.; ANDRES, C. *Extreme Programming Explained: Embrace Change (2nd Edition)*. Boston: Addison-Wesley, 2004. ISBN 0321278658.

BECK, K.; FOWLER, M. *Planning Extreme Programming*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000. ISBN 0201710919.

BOEHM, B. A view of 20th and 21st century software engineering. In: *ICSE '06: Proceedings of the 28th international conference on Software engineering*. New York, NY, USA: ACM, 2006. p. 12–29. ISBN 1-59593-375-1.

BOEHM, B.; TURNER, R. *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN 0321186125.

CAMARGO, D. T. *Feature Driven Development: Uma alternativa metodológica de desenvolvimento ágil às microempresas para o alcance dos níveis de maturidade iniciais conforme o modelo de referência para melhoria do processo de software brasileiro*. Monografia — Universidade Estadual de Londrina, Paraná, 2007.

COAD, P.; LUCA, J. d.; LEFEBVRE, E. *Java Modeling Color with Uml: Enterprise Components and Process with Cdrom*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999. ISBN 013011510X.

FOWLER, M. *Refatoração: Aperfeiçoando o Projeto de Código Existente*. Boston, MA, USA: Bookman, 2004.

GROUP, T. S. *Report Chaos*. [S.l.], 1995.

HEPTAGON. *Heptagon*. 1997. Disponível em: <http://www.heptagon.com.br/>. Acesso em: 15 Jul. 2009.

HIGHSMITH, J. *History: The Agile Manifesto*. 2001. Disponível em: <http://agilemanifesto.org/history.html>. Acesso em: 01 Ago. 2009.

HIGHSMITH, J.; COCKBURN, A. Agile software development: The business of innovation. *Computer*, IEEE Computer Society, Los Alamitos, CA, USA, v. 34, n. 9, p. 120–122, 2001. ISSN 0018-9162.

KERIEVSKY, J. *Refactoring to Patterns*. [S.l.]: Addison-Wesley, 2004.

LAURINDO, D. *Feature Driven Development (FDD): Uma metodologia ágil utilizada para apoiar a implantação dos níveis G e F do modelo de melhoria de processo de software - Mps.Br*. Monografia — Universidade Estadual de Londrina, Paraná, 2007.

MANIFESTO, A. *Agile Manifesto*. 2001. Disponível em: <http://agilemanifesto.org>. Acesso em: 01 Ago. 2009.

MARCAL, A. S. C.; SOARES, F. S. F.; BELCHIOR, A. D. Mapping cmmi project management process areas to scrum practices. *Software Engineering Workshop, Annual IEEE/NASA Goddard*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 13–22, 2007. ISSN 1550-6215.

PALMER, S. R.; FELSING, J. M. *A Practical Guide to the Feature Driven Development*. [S.l.]: Prentice Hall, 2002.

SANTANA, C. A.; TIMÓTEO, A. L.; VASCONCELOS, A. M. L. Mapeamento do modelo de melhoria do processo de software brasileiro (mps.br) para empresas que utilizam extreme programming (xp) como metodologia de desenvolvimento. *V Simpósio Brasileiro de Qualidade de Software [SBQS]*, 05 2006.

SCHWABER, K. The scrum development process. *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications*, Austin, Texas, USA, 1995.

SCHWABER, K.; BEEDLE, M. *Agile Software Development with Scrum*. [S.l.]: Prentice Hall, 2001.

SHULL, F. *et al.* What we have learned about fighting defects. In: *METRICS '02: Proceedings of the 8th International Symposium on Software Metrics*. Washington, DC, USA: [s.n.], 2002. p. 249. ISBN 0-7695-1339-5.

SIQUEIRA, H. B. A. *Mapeamento das práticas de Scrum nas áreas de processo do CMMI e uma proposta para sua aderência*. Monografia — Universidade Federal de Pernambuco, Pernambuco, 2007.

SOFTEX. *SOFTEX*. 2009. Disponível em: <http://www.softex.br>. Acesso em: 22 Jul. 2009.

SOFTEX MR-MPS. *Guia de Implementação Parte 1: Fundamentação para Implementação do Nível G do MR-MPS*. [S.l.], 05 2009.

SOFTWARE, A. B. das Empresas de. *Associação Brasileira das Empresas de Software*. 2009. Disponível em: <http://www.abes.org.br>. Acesso em: 02 Abr. 2009.

SOFTWARE, I. I.; SINGH, R. International standard. In: *Software Lifecycle Process Standards*. [S.l.: s.n.], 1989.

SOMMERVILLE, I. *Software engineering (5th ed.)*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1995. ISBN 0-201-42765-6.

TELES, V. M. *Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade*. [S.l.]: Novatec, 2004. ISBN 85-7522-047-0.

TELES, V. M. *Um estudo de caso da adoção das práticas e valores do Extreme Programming*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, 2005.

WELLS, J. D. *Extreme Programming*. 1999. Disponível em: <www.extremeprogramming.org/>. Acesso em: 13 Jul. 2009.

APÊNDICE A – Tabela GPR X MAs

Tabela 3: Mapeamento GPR X MAs

	XP	***Scrum***	**FDD**
GPR1 - O escopo do trabalho para o projeto é definido.	PS	S	S
GPR2 - As tarefas e os produtos de trabalho do projeto são dimensionados utilizando métodos apropriados.	PS	PS	PS
GPR3 - O modelo e as fases do ciclo de vida do projeto são definidas.	S	S	S
GPR4 - O esforço e o custo para a execução das tarefas e dos produtos de trabalho são estimados com base em dados históricos ou referências técnicas.	S	PS	PS
GPR5 - O orçamento e o cronograma do projeto, incluindo marcos e/ou pontos de controle, são estabelecidos e mantidos.	PS	PS	PS
GPR6 - Os riscos do projeto são identificados e o seu impacto, probabilidade de ocorrência e prioridade de tratamento são determinados e documentados.	PS	S	PS
GPR7 - Os recursos humanos para o projeto são planejados considerando o perfil e o conhecimento necessários para executá-lo.	PS	PS	PS
GPR8 - As tarefas, os recursos e o ambiente de trabalho necessários para executar o projeto são planejados.	PS	S	PS

GPR9 - Os dados relevantes do projeto são identificados e planejados quanto à forma de coleta, armazenamento e distribuição. Um mecanismo é estabelecido para acessá-los, incluindo, se pertinente, questões de privacidade e segurança.	S	S	S
GPR10 - Planos para a execução do projeto são estabelecidos e reunidos no Plano do Projeto.	NS	NS	NS
GPR11 - A viabilidade de atingir as metas do projeto, considerando as restrições e os recursos disponíveis, é avaliada. Se necessário, ajustes são realizados.	S	S	S
GPR12 - O Plano do Projeto é revisado com todos os interessados e o compromisso com ele é obtido.	PS	PS	PS
GPR13 - O progresso do projeto é monitorado com relação ao estabelecido no Plano do Projeto e os resultados são documentados.	PS	PS	PS
GPR14 - O envolvimento das partes interessadas no projeto é gerenciado.	S	S	S
GPR15 - Revisões são realizadas em marcos do projeto e conforme estabelecido no planejamento.	S	S	S
GPR16 - Registros de problemas identificados e o resultado da análise de questões pertinentes, incluindo dependências críticas, são estabelecidos e tratados com as partes interessadas.	PS	S	PS
GPR17 - Ações para corrigir desvios em relação ao planejado e para prevenir a repetição dos problemas identificados são estabelecidas, implementadas e acompanhadas até a sua conclusão.	PS	PS	PS

APÊNDICE B – Tabela GRE X MAs

Tabela 4: Mapeamento GRE X MAs

	XP	***Scrum***	**FDD**
GRE1 - O entendimento dos requisitos é obtido junto aos fornecedores de requisitos.	PS	PS	PS
GRE2 - Os requisitos de software são aprovados utilizando critérios objetivos.	PS	PS	PS
GRE3 - A rastreabilidade bidirecional entre os requisitos e os produtos de trabalho é estabelecida e mantida.	NS	NS	NS
GRE4 - Revisões em planos e produtos de trabalho do projeto são realizadas visando identificar e corrigir inconsistências em relação aos requisitos.	PS	PS	PS
GRE5 - Mudanças nos requisitos são gerenciadas ao longo do projeto.	PS	PS	PS

APÊNDICE C – Motivos para os requisitos classificados como Parcialmente Satisfeito (GPR)

Motivos para a classificação dos requisitos Parcialmente Satisfeitos

- Falta de Registro (FR)
- Falta de Formalismo (FF)
- Incompletude no formalismo (IF)
- Falta de Registro e Formalismo (FRF)

Tabela 5: Tabela contendo os motivos para os requisitos do GPR classificados como Parcialmente Satisfeito (PS)

	XP	***Scrum***	**FDD**
GPR1 - O escopo do trabalho para o projeto é definido.	IF	–	–
GPR2 - As tarefas e os produtos de trabalho do projeto são dimensionados utilizando métodos apropriados.	FRF	FRF	FRF
GPR3 - O modelo e as fases do ciclo de vida do projeto são definidas.	–	–	–
GPR4 - O esforço e o custo para a execução das tarefas e dos produtos de trabalho são estimados com base em dados históricos ou referências técnicas.	–	FR	IF
GPR5 - O orçamento e o cronograma do projeto, incluindo marcos e/ou pontos de controle, são estabelecidos e mantidos.	IF	IF	IF
GPR6 - Os riscos do projeto são identificados e o seu impacto, probabilidade de ocorrência e prioridade de tratamento são determinados e documentados.	FR	–	IF

GPR7 - Os recursos humanos para o projeto são planejados considerando o perfil e o conhecimento necessários para executá-lo.	FF	FRF	FRF
GPR8 - As tarefas, os recursos e o ambiente de trabalho necessários para executar o projeto são planejados.	FR	–	FR
GPR9 - Os dados relevantes do projeto são identificados e planejados quanto à forma de coleta, armazenamento e distribuição. Um mecanismo é estabelecido para acessá-los, incluindo, se pertinente, questões de privacidade e segurança.	–	–	–
GPR10 - Planos para a execução do projeto são estabelecidos e reunidos no Plano do Projeto.	–	–	–
GPR11 - A viabilidade de atingir as metas do projeto, considerando as restrições e os recursos disponíveis, é avaliada. Se necessário, ajustes são realizados.	–	–	–
GPR12 - O Plano do Projeto é revisado com todos os interessados e o compromisso com ele é obtido.	FF	FF	FF
GPR13 - O progresso do projeto é monitorado com relação ao estabelecido no Plano do Projeto e os resultados são documentados.	FR	FR	FR
GPR14 - O envolvimento das partes interessadas no projeto é gerenciado.	–	–	–
GPR15 - Revisões são realizadas em marcos do projeto e conforme estabelecido no planejamento.	–	–	–
GPR16 - Registros de problemas identificados e o resultado da análise de questões pertinentes, incluindo dependências críticas, são estabelecidos e tratados com as partes interessadas.	FR	–	FR
GPR17 - Ações para corrigir desvios em relação ao planejado e para prevenir a repetição dos problemas identificados são estabelecidas, implementadas e acompanhadas até a sua conclusão.	FR	FR	FR

APÊNDICE D – Motivos para os requisitos classificados como Parcialmente Satisfeito (GRE)

Motivos para a classificação dos requisitos Parcialmente Satisfeitos

- Falta de Registro (FR)
- Falta de Registro e Critérios Objetivos (FRCO)

Tabela 6: Tabela contendo os motivos para os requisitos do GRE classificados como Parcialmente Satisfeito (PS)

	XP	***Scrum***	**FDD**
GRE1 - O entendimento dos requisitos é obtido junto aos fornecedores de requisitos.	FRCO	FRCO	FRCO
GRE2 - Os requisitos de software são aprovados utilizando critérios objetivos.	FRCO	FRCO	FRCO
GRE3 - A rastreabilidade bidirecional entre os requisitos e os produtos de trabalho é estabelecida e mantida.	–	–	–
GRE4 - Revisões em planos e produtos de trabalho do projeto são realizadas visando identificar e corrigir inconsistências em relação aos requisitos.	FR	FR	FR
GRE5 - Mudanças nos requisitos são gerenciadas ao longo do projeto.	FR	FR	FR