UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

RENATA CRISTINA FURLAN

AGILE UNIFIED PROCESS E A ADERÊNCIA AO MODELO DE PROCESSO DE SOFTWARE MPS.BR NÍVEL G

MONOGRAFIA DE ESPECIALIZAÇÃO

MEDIANEIRA
2012

RENATA CRISTINA FURLAN

AGILE UNIFIED PROCESS E A ADERÊNCIA AO MODELO DE PROCESSO DE SOFTWARE MPS.BR NÍVEL G

Monografia apresentada como requisito parcial à obtenção do título de Especialista na Pós Graduação em Engenharia de Software, da Universidade Tecnológica Federal do Paraná – UTFPR – Campus Medianeira.

Orientador: Prof Dr. Vilson Dalle Mole.

MEDIANEIRA
2012

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Especialização em Engenharia de Software

TERMO DE APROVAÇÃO

AGILE UNIFIED PROCESS E A ADERÊNCIA AO MODELO DE PROCESSO DE SOFTWARE MPS.BR NÍVEL G

Por

Renata Cristina Furlan

Esta monografia foi apresentada às 11h10min do dia 17 de março de 2012 como requisito parcial para a obtenção do título de Especialista no curso de Especialização em Engenharia de Software, da Universidade Tecnológica Federal do Paraná, *Campus* Medianeira. A acadêmica foi arguida pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado com louvor e mérito.

_____
Prof. Dr Vilson Dalle Mole
UTFPR – Campus Toledo
(orientador)

_____
Prof. M.Sc. Alan Gavioli
UTFPR – Campus Medianeira

_____
Prof. M.Sc. Juliano Rodrigo Lamb
UTFPR – Campus Medianeira

À minha família.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me possibilitar todos os momentos vividos.

Agradeço a meus pais, Romeu José Furlan e Clair Terezinha Furlan e a minha tia Cristiane Canan pelo incentivo, apoio e força constante em minha vida. Agradeço também a minha irmã Camila Regina Furlan pela parceria em todos os momentos.

Agradeço a meu orientador Professor Vilson Dalle Mole pelas orientações durante todo o processo de construção do presente trabalho, corrigindo além de erros de português e normas ortográficas, mas também o rumo da monografia.

A todos os professores, pelos conhecimentos passados durante todo o curso de Especialização em Engenharia de Software.

A todos que direta ou indiretamente contribuíram para a conclusão desse trabalho.

“Deus não joga dados com o universo”.

(ALBERT EINSTEIN)

RESUMO

FURLAN, Renata C. Agile Unified Process e a Aderência ao Modelo de Processo de Software MPS.BR. 2012. 78 p. Monografia (Especialização em Engenharia de Software). Universidade Tecnológica Federal do Paraná, Medianeira, 2012.

O desenvolvimento de software possui papel crítico na sociedade, e a preocupação com sua qualidade se torna fator estratégico para que empresas de software conquistem espaço e reconhecimento no mercado. Estudos abordados no presente trabalho revelam que a qualidade do processo seguido durante o desenvolvimento do produto de software está diretamente ligada com sua qualidade, fazendo com que as empresas procurem melhorar seus processos através da adoção de modelos de referencias de processo, como o Modelo MPS.BR, que avalia os processos da empresa, conforme a aderência com suas práticas e processos definidos, atribuindo ao final um nível de maturidade. Em contrapartida, com a popularização de métodos ágeis de desenvolvimento de software, conhecidos por serem leves, o presente trabalho analisou a aderência do método ágil AUP com o Modelo MPS.BR nível G de maturidade, e constatou que a falta de documentação e atividades que formalizassem a execução de itens exigidos pelo Modelo MPS.BR, tornaram o AUP não aderente em sua totalidade com o nível G de maturidade.

Palavras-chave: AUP. Desenvolvimento Ágil. Modelo de Processo de Software.

ABSTRACT

FURLAN, Renata C. Agile Unified Process e a Aderência ao Modelo de Processo de Software MPS.BR. 2012. 78 p. Monografia (Especialização em Engenharia de Software). Universidade Tecnológica Federal do Paraná, Medianeira, 2012.

Software development has a critical role in society, and concern about its quality becomes a strategic factor for software companies to conquer space and market recognition. Studies discussed in this paper shows that the quality of the process followed during the development of the software product is directly linked to their quality, so, companies have been seeking to improve their processes through the adoption of process reference models, such as the MPS.BR model, which assesses the company's processes, through their adherence to practices and defined processes, giving at the end, a level of maturity. However, with the popularization of agile software development, known for being lightweight, this study examined the adherence of the AUP agile method with the MPS.BR model, level of maturity G, and found that the lack of documentation and activities to formalize the implementation of items required by the MPS.BR model, the AUP become noncompliant in its entirety with the G level of maturity.

Keywords: AUP. Agile Development. Software Process Model.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O software conquistou um papel essencial e crítico na sociedade durante os últimos anos. Cada vez mais há dependência dos recursos e serviços oferecidos através de sistemas informatizados. Infelizmente, as aplicações de software são produtos complexos e difíceis de desenvolver e testar. Muitas vezes, o software apresenta comportamentos inesperados e indesejados que podem causar problemas e danos graves. Por estas razões, pesquisadores e profissionais buscam cada vez mais melhorar a qualidade do software a ser desenvolvido, através de uma série de abordagens e técnicas. Uma das principais direções seguidas por eles está centrada no estudo e aprimoramento do processo através do qual o software é desenvolvido. A hipótese principal é que existe uma correlação direta entre a qualidade do processo e a qualidade do software desenvolvido. A área de pesquisa que lida com estas questões é chamada Processo de Software (FUGGETTA, 2000).

Como uma disciplina autônoma, a área de Processo de Software foi iniciada na década de 80, através de uma série de *workshops* e eventos (em particular, o Workshop Internacional de Processo de Software). Importantes instituições foram criadas nos EUA e na Europa para estudar os processos de software: o *Software Engineering Institute* (SEI, Pittsburgh, EUA) e o *European Software Institute* (ESI, Bilbao, Espanha). Até as organizações de padronização começaram importantes esforços centrados em processos de software, como a ISO, que criou duas normas importantes: a ISO 12207 (Atividades do Ciclo de Vida do Software) e a ISO 15504 (Determinação da Capacidade do Processo de Software) (FUGGETTA, 2000).

Mais tarde, em dezembro de 2003, coordenado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), foi criado o programa MPS.BR, cujo objetivo é a Melhoria de Processo do Software Brasileiro baseando-se nos conceitos de maturidade e capacidade de processo para a avaliação e melhoria da qualidade e produtividade de produtos de software e serviços correlatos estando em consonância com as principais abordagens internacionais para definição, avaliação e melhoria de processos de software. O Modelo MPS.BR busca ser adequado ao perfil de empresas com diferentes tamanhos e características, públicas

e privadas, embora com especial atenção às micro, pequenas e médias empresas (SOFTEX, 2011).

Em contrapartida às iniciativas de pesquisa na área de processo de software, muitos anos antes, cerca de setenta e cinco anos atrás, o principal fundamento dos Métodos Ágeis de Desenvolvimento de Software já era adotado por engenheiros do Departamento de Defesa dos Estados Unidos (DoD – *Departament of Defence*): o Projeto e Desenvolvimento Iterativo e Incremental (IIDD – *Iterative and Incremental Design and Development*) (SOFTWARE ENGINEERING INSTITUTE, 2011).

Ao longo dos anos, o IIDD ganhou ampla aceitação na comunidade de software tomando várias formas, incluindo o *Rational Unified Process* (RUP) [1]. Com essa proliferação, veio a necessidade de coordenar e comparar essas variações do método em benefício de seu crescimento e sustentação. O resultado foi um "encontro de mentes" entre os principais responsáveis pela teoria e a aplicação de cada variação do método (SOFTWARE ENGINEERING INSTITUTE, 2011).

Em fevereiro de 2001, dezessete desenvolvedores de software se reuniram para discutir métodos de desenvolvimento leves (*lightweight*). Eles publicaram o Manifesto para Desenvolvimento Ágil de Software (*The Agile Manifesto*) para definir a abordagem hoje conhecida como Desenvolvimento Ágil de Software (BECK et al., 2001).

Em setembro de 2005, Scott Ambler desenvolveu o *Agile Unified Process* (AUP), uma versão simplificada do RUP que descreve uma abordagem para desenvolvimento de software utilizando técnicas e conceitos ágeis que ainda assim permanece fiel ao RUP. A abordagem aplica técnicas ágeis como Desenvolvimento Dirigido por Testes (TDD - *Test Driven Development*), Desenvolvimento Dirigido por Modelagem Ágil (AMDD - *Agile Model Driven Development*), Gerenciamento de Mudanças Ágeis e *Refactoring* de Banco de Dados para melhorar sua produtividade (AMBLER, 2009).

[1] Na década de 1990, James Rumbaugh, Grady Booch e Ivar Jacobson desenvolveram o Processo Unificado (UP - *Unified Process*), um arcabouço para a engenharia de software orientada a objetos usando a UML (*Unified Modeling Language*) (PRESSMAN, 2006; KRUCHTEN, 2003). O UP e, especialmente, a sua implementação pela *Rational Software Corporation*, o *Rational Unified Process* (RUP), é um processo de desenvolvimento de software concentrado em assegurar a produção de sistemas (KRUCHTEN, 2003).

Assumindo a importância do processo de software na qualidade do produto desenvolvido e a disseminação dos métodos ágeis de desenvolvimento de software, o presente trabalho apresentará um estudo sobre o processo de desenvolvimento AUP e sua aderência ao Modelo de Melhoria de Processo MPS.BR nível G (Parcialmente Gerenciado) de Maturidade.

1.1 OBJETIVO GERAL

Analisar a aderência do *Agile Unified Process* (AUP) aos processos definidos pelo Modelo de Melhoria de Processo MPS.BR Nível G (Parcialmente Gerenciado) de Maturidade.

1.2 OBJETIVOS ESPECÍFICOS

- De acordo com os processos definidos no nível de maturidade G do Modelo MPS.BR, desenvolver uma análise de compatibilidade do AUP com esses processos.

1.3 JUSTIFICATIVA

O Manifesto para Desenvolvimento Ágil de Software diz o seguinte (BECK et al., 2001):

> "Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:
> - Indivíduos e interações mais que processos e ferramentas;
> - Software em funcionamento mais que documentação abrangente;
> - Colaboração com o cliente mais que negociação de contratos;
> - Responder a mudanças mais que seguir um plano.
>
> Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda."

O Manifesto Ágil é frequentemente citado por defensores dos métodos ágeis como justificativa por não terem processos, por não documentarem seu trabalho, e por não terem planos. Esta interpretação dá justificativa aos opositores dos métodos ágeis para acusá-los de serem indisciplinados. No entanto, estas interpretações são equivocadas (SOFTWARE ENGINEERING INSTITUTE, 2006), como mostra a pesquisa realizada anualmente pela VersionOne a respeito do estado do desenvolvimento ágil entre as empresas (VERSIONONE, 2011). Em 2011, quase dois terços dos entrevistados disseram que até metade dos projetos em sua empresa foram executados utilizando métodos ágeis, e que elas têm adotado práticas ágeis em três ou mais equipes. Além disso, 75% dos entrevistados disseram que a produtividade em sua empresa aumentou e 84% que a capacidade de gerenciar as mudanças de prioridades dos clientes melhorou (VERSIONONE, 2011).

A ampla adoção de métodos ágeis por muitos profissionais de engenharia de software é resultado de fatores como a publicação do Manifesto Ágil e da Declaração de Independência do Gerenciamento de Projetos [2] (DoI - *Project Management Declaration of Interdependence*); a publicação de livros; a formação de organizações sem fins lucrativos para promover a abordagem ágil; o uso difundido da Internet para a pesquisa por profissionais de software, entre outros (SOFTWARE ENGINEERING INSTITUTE, 2006).

Da mesma forma que os métodos ágeis têm possibilitado resultados positivos entre as empresas que os utilizam, o Modelo MPS.BR, assim como outros modelos de maturidade e padronização, tem permitido principalmente às pequenas e médias empresas aprimorar seus processos de software, melhorar a capacidade de produção e permitir que a qualidade do software, em geral, possa ser garantida e obtida com embasamento em conhecimento atual e de fácil acesso em engenharia de software (TRAVASSOS; KALINOWSKI, 2008).

Diante desses fatores, considerando o cenário no qual se discute a adoção de abordagens ágeis para desenvolvimento de software e a utilização do Modelo MPS.BR para obter um nível de maturidade do processo produtivo do software, o presente trabalho busca realizar uma análise da aderência do processo

[2] DoI (*Project Management Declaration of Interdependence)* foi criado por autores do Manifesto Ágil para estabelecer os Seis Princípios do Gerenciamento Ágil.

de desenvolvimento ágil AUP com os processos definidos pelo nível de maturidade G (Parcialmente Gerenciado) do modelo MPS.BR.

1.4 ESTRUTURA DO TRABALHO

O presente trabalho é constituído de quatro capítulos. O primeiro apresenta uma introdução, justificativas e objetivos do presente trabalho, bem como uma breve apresentação dos temas abordados pelos demais capítulos..

O capítulo dois apresenta a fundamentação teórica do presente trabalho, através de um estudo sobre Qualidade de Software e Modelos de Referências de Processo, detalhando especificamente o Modelo de Processo MPS.BR. Em seguida, é apresentado um estudo sobre o Desenvolvimento Ágil de Software, e mais detalhadamente, o processo AUP (*Agile Unified Process*), foco principal do presente trabalho.

No capitulo três, são apresentadas as considerações da análise de conformidade do AUP com os processos definidos pelo Modelo MPS.BR nível G.

O capítulo quatro ressalta as considerações finais sobre o desenvolvimento do presente trabalho, e relaciona os possíveis trabalhos que podem ser derivados.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 QUALIDADE DE SOFTWARE E MODELOS DE REFERÊNCIA DE PROCESSO

De acordo com PRESSMAN (2006), Qualidade de Software é a satisfação de requisitos funcionais e de desempenho explicitamente declarados, normas de desenvolvimento explicitamente documentadas e características implícitas que são esperadas em todo o software desenvolvido profissionalmente. São enfatizados três pontos importantes:

- Requisitos de software são a fundação a partir da qual a qualidade é medida (PRESSMAN, 2006). Segundo o SWEBOK (*Software Engineering Boby of Knowlegde*), os requisitos de software definem as características de qualidade exigidos do software e influenciam os métodos de medição e critérios de aceitação para avaliar essas características (IEEE COMPUTER SOCIETY, 2004);
- Normas especificadas definem um conjunto de critérios de desenvolvimento que guiam o modo pela qual o software é construído. Se os critérios não são seguidos, quase sempre vai resultar em falta de qualidade;
- Há um conjunto de requisitos implícitos que frequentemente não são mencionados (por exemplo, o desejo de facilidade de uso). Se o software satisfaz seus requisitos explícitos, mas deixa de satisfazer os requisitos implícitos, a qualidade do software é suspeita.

Qualidade de software é uma mistura complexa de fatores que variam de aplicação para aplicação de acordo com o que os clientes (*stakeholders* [3]) encomendam. Todavia, não deve ser algo para se preocupar depois que o sistema esteja pronto (PRESSMAN, 2006).

A garantia de qualidade é uma atividade essencial para qualquer negócio que faz produtos para serem usados por outros. Antes do século XX, a garantia da qualidade era de responsabilidade tão somente do artesão que construía um

[3] São as partes interessadas do sistema. É qualquer pessoa ou organização que tenha interesse ou seja afetado pelo projeto.

produto. A primeira função formal de garantia e controle de qualidade foi introduzida nos laboratórios Bell, em 1916, e disseminou-se rapidamente por todo o mundo industrializado. Durante os anos 40, abordagens mais formais para o controle de qualidade foram sugeridas. Elas se concentravam na medição e aperfeiçoamento contínuo do processo como elementos-chave da gerência de qualidade (PRESSMAN, 2006). FUGGETTA (2000), IEEE COMPUTER SOCIETY (2004) e SOFTWARE ENGINEERING INSTITUTE (2006), também afirmam que existe uma correlação direta entre a qualidade do processo seguido e a qualidade do software desenvolvido.

Segundo PRESSMAN (2006), modelos de processo que enfatizam a definição, identificação e aplicação detalhada de atividades e tarefas de processo são aplicados com o objetivo de melhorar a qualidade do sistema para tornar os projetos mais gerenciáveis, as datas de entrega e os custos mais previsíveis e para guiar as equipes de engenheiros de software à medida que eles realizam o trabalho necessário para construir um sistema.

Diferentes abordagens têm sido desenvolvidas para melhorar e avaliar a capacidade de desenvolvimento de software. Tendo em vista a diversidade de tais abordagens, a ISO/IEC iniciou um esforço para desenvolver um *framework* internacionalmente reconhecido para a avaliação de processos de software que culminou com a publicação da norma internacional ISO/IEC 15504. Esta norma provê uma abordagem genérica para a avaliação da capacidade de processos baseada em modelos. A avaliação da capacidade dos processos é realizada utilizando um *framework* de medição que utiliza descrições de atributos de processo, aplicáveis a qualquer processo. Estes atributos representam características mensuráveis necessárias para gerenciar um processo e melhorar sua capacidade de operar dentro de uma escala de níveis de capacidade (KALINOWSKI, 2010).

Visando apoiar a definição, avaliação e melhoria de processos, a ISO/IEC iniciou ainda um esforço para desenvolver um modelo de referência de processos para o domínio de engenharia de software. A norma base para esta iniciativa foi a ISO/IEC 12207, publicada em 1995. Esta norma provê um conjunto de processos, atividades e tarefas para ciclos de vida de produtos e serviços de software (KALINOWSKI, 2010).

Outro modelo relevante e internacionalmente reconhecido para melhoria de processos de software é o CMMI® (*Capability Maturity Model Integration* - Modelo Integrado de Maturidade e de Capacidade). O CMMI® é um modelo de maturidade de processos desenvolvido pelo SEI (S*oftware Engineering Institute*) para o desenvolvimento de produtos e serviços de software (PRESSMAN, 2006) (SOFTWARE ENGINEERING INSTITUTE, 2006). O modelo CMMI® é consistente com a norma internacional ISO/IEC 15504:2003 e provê duas representações para melhoria de processos de software. A representação continua permite que organizações melhorem seus processos individualmente em escalas de níveis de capacidade variando de 0 a 5. A representação em estágios permite a melhoria de um conjunto relacionado de processos visando melhorar processos organizacionais em escalas de níveis de maturidade variando de 1 a 5 (SOFTWARE ENGINEERING INSTITUTE, 2006; KALINOWSKI, 2010).

Semelhante ao CMMI®, foi criado o programa MPS.BR, cujo objetivo é a Melhoria do Processo de Software Brasileiro baseando-se nos conceitos de maturidade e capacidade de processo para a avaliação e melhoria da qualidade e produtividade de produtos de software e serviços correlatos estando em consonância com as principais abordagens internacionais para definição, avaliação e melhoria de processos de software (SOFTEX, 2011).

Tanto o CMMI® como o MPS.BR, são modelos de referência de processo cujo objetivo é promover a melhoria do processo de desenvolvimento de software das empresas, e consequentemente, melhorar a qualidade dos produtos por elas desenvolvidos. Esses modelos definem requisitos que os processos das empresas devem atender para estar em conformidade com suas definições de capacidade e maturidade de processo. Após avaliação por empresas certificadoras desses modelos, as empresas avaliadas podem (no caso de avaliação positiva) atestar que seus processos possuem conformidade com esses modelos de referência e receber um determinado nível de capacidade e/ou maturidade de seus processos.

2.2 MODELO DE MELHORIA DE PROCESSO DE SOFTWARE BRASILEIRO (MPS.BR)

A melhoria contínua da capacidade de desenvolvimento é fundamental para que organizações de software prosperem em mercados competitivos. Ao longo dos anos modelos de referência têm surgido para guiar a melhoria da capacidade de processos de engenharia de software (TRAVASSOS, 2008). Entretanto, a melhoria baseada neste tipo de modelo costuma ser de longo prazo e requerer grandes investimentos (GOLDENSON, 2003). Estes obstáculos podem se tornar impeditivos para que organizações melhorem seus processos, especialmente para Pequenas e Médias Empresas (PMEs) que operam com rígidas restrições financeiras (KALINOWSKI, 2010).

No Brasil, onde aproximadamente 73% da indústria de software é constituída por PMEs, poucas organizações têm adotado modelos de referência (KALINOWSKI, 2010; TRAVASSOS, 2008). Constatou-se que normalmente as organizações só implementam as boas práticas da engenharia de software quando estas são exigidas em avaliações de processos (NOGUEIRA, 2006). Acredita-se que o uso destas boas práticas possa melhorar o desempenho das organizações com respeito a custo, prazo, produtividade, qualidade, satisfação do cliente e retorno do investimento (ROI) e, consequentemente, aumentar sua vantagem competitiva (KALINOWSKI, 2010; TRAVASSOS, 2008).

Em 2003 foi criado o Modelo MPS.BR, sob a coordenação da Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), com apoio do governo (MCT e FINEP), SEBRAE/PROIME e BID (Banco Interamericano de Desenvolvimento), da indústria de software brasileira e de instituições de pesquisa. O principal objetivo desta iniciativa é desenvolver e disseminar um modelo de melhoria de processos visando estabelecer um caminho economicamente viável para que organizações, incluindo as PMEs, alcancem os benefícios da melhoria de processos e da utilização de boas práticas da engenharia de software em um intervalo de tempo razoável (SOFTEX, 2011; KALINOWSKI, 2010; TRAVASSOS, 2008; SANTOS, 2008).

O Modelo MPS estabelece um modelo de processos de software e um método de avaliação de processos. Esta estrutura fornece sustentação e garante que o modelo esteja sendo empregado de forma coerente com as suas definições. Além disso, estabelece também um modelo de negócio para apoiar a sua adoção pelas empresas brasileiras desenvolvedoras de software (SOFTEX, 2011).

As normas ISO/IEC 12207 e ISO/IEC 15504 foram usadas como base técnica para definir os componentes do Modelo MPS. Adicionalmente, tendo em vista a importância do modelo CMMI® para organizações brasileiras que atuam em mercados internacionais, a equipe técnica do Modelo MPS considerou o CMMI® como um complemento técnico para a definição de seus processos (SOFTEX, 2011; TRAVASSOS, 2008).

O Modelo MPS possui três componentes principais: o Modelo de Referência MPS (MR-MPS) (constituído pelo Guia Geral, Guia de Implementação e Guia de Aquisição); o Método de Avaliação MPS (MA-MPS) (constituído pelo Guia de Avaliação) e o Modelo de Negócios MPS (MN-MPS), conforme ilustrado na Figura 1.

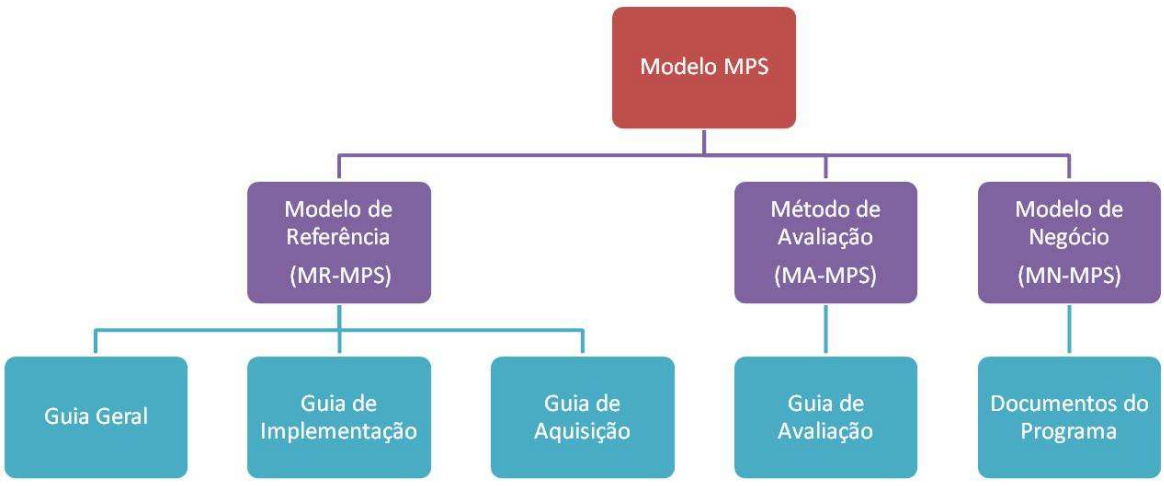

FIGURA 1– Componentes do Modelo MPS.BR.
Fonte: Adaptado de SOFTEX (2011).

O Modelo de Referência MR-MPS contém os requisitos que os processos das unidades organizacionais[4] devem atender para estar em conformidade com o

[4] Unidade Organizacional: Parte de uma organização que será avaliada. Uma unidade organizacional é tipicamente parte de uma grande organização, embora, em uma pequena organização, a unidade organizacional possa ser toda a organização (SOFTEX, 2011).

MR-MPS. Ele contém as definições dos níveis de maturidade, processos e atributos do processo, descrito no Guia Geral. O MR-MPS está em conformidade com os requisitos de modelos de referência de processo da Norma Internacional ISO/IEC 15504-2 (SOFTEX, 2011).

O Guia de Aquisição é um documento complementar destinado a organizações que pretendam adquirir software e serviços correlatos. O Guia de Aquisição não contém requisitos do MR-MPS, mas boas práticas para a aquisição de software e serviços correlatos (SOFTEX, 2011).

O Guia de Implementação sugere formas de implementar cada um dos níveis do MR-MPS. São explicações que não constituem requisitos do modelo e devem ser consideradas apenas em caráter informativo (SOFTEX, 2011).

O Guia de Avaliação contém o processo e o método de avaliação MA-MPS, os requisitos para os avaliadores líderes, avaliadores adjuntos e Instituições Avaliadoras (IA). O processo e o método de avaliação MA-MPS estão em conformidade com a Norma Internacional ISO/IEC 15504-2 (SOFTEX, 2011).

De acordo com SOFTEX (2011), o Modelo de Negócio MN-MPS descreve regras de negócio para implementação do MR-MPS pelas Instituições Implementadoras (II), avaliação seguindo o MA-MPS pelas Instituições Avaliadoras (IA), organização de grupos de empresas pelas Instituições Organizadoras de Grupos de Empresas (IOGE) para implementação do MR-MPS e avaliação MA-MPS, certificação de Consultores de Aquisição (CA) e programas anuais de treinamento do MPS.BR por meio de cursos, provas e workshops.

2.2.1 Níveis de Capacidade e Maturidade

O Modelo de Referência MR-MPS define níveis de maturidade que são uma combinação entre processos e sua capacidade (SOFTEX, 2011).

A definição dos processos segue os requisitos para um modelo de referência de processo apresentados na ISO/IEC 15504-2, declarando o propósito e os resultados esperados de sua execução. Isso permite avaliar e atribuir graus de efetividade na execução dos processos. As atividades e tarefas necessárias para

atender ao propósito e aos resultados esperados não são definidas no Guia Geral, devendo ficar a cargo dos usuários do MR-MPS (SOFTEX, 2011).

De acordo com SOFTEX (2011), a capacidade do processo é a caracterização da habilidade do processo para alcançar os objetivos de negócio, atuais e futuros; estando relacionada com o atendimento aos Atributos de Processo (AP) e os Resultados Esperados (RAPs) associados aos processos de cada nível de maturidade. No MR-MPS, à medida que a unidade organizacional evolui nos níveis de maturidade, um maior nível de capacidade para desempenhar o processo deve ser atingido. Os níveis são acumulativos, ou seja, na passagem para um nível de maturidade superior, os processos anteriormente implementados devem passar a ser executados no nível de capacidade exigido neste nível superior. São nove diferentes níveis APs definidas:

- AP 1.1 – O processo é executado: evidencia o quanto o processo atinge o seu propósito, ou seja, está diretamente relacionado ao atendimento do propósito do processo (SOFTEX, 2011b);
- AP 2.1 – O processo é gerenciado: evidencia o quanto a execução do processo é gerenciada, ou seja, está relacionado à gerência dos processos. A implementação deste atributo de processo implica no planejamento da execução do processo, atribuindo responsabilidade e autoridade para sua execução, bem como fornecendo recursos adequados. Envolve também o monitoramento e controle da execução dos processos, tomando ações corretivas, quando necessárias (SOFTEX, 2011b);
- AP 2.2 – Os produtos de trabalho do processo são gerenciados: evidencia o quanto os produtos de trabalho produzidos pelo processo são gerenciados apropriadamente, ou seja, não somente o processo será gerenciado, mas também os produtos de trabalho (SOFTEX, 2011c);
- AP 3.1 – O processo é definido: evidencia o quanto um processo padrão é mantido para apoiar a implementação do processo definido. Este atributo de processo implica na definição dos processos padrão da organização e em sua adaptação para o processo definido. Isto significa que a organização deve ter processos padrão que atendam a todos os processos do MR-MPS no nível considerado, de forma a atender às necessidades dos projetos e da organização (SOFTEX, 2011d);

- AP 3.2 – O processo está implementado: evidencia o quanto o processo padrão é efetivamente implementado como um processo definido para atingir seus resultados. A implementação deste atributo de processo implica na definição do processo definido, sua efetiva implementação por meio da qual se obtém dados para entendimento de seu desempenho e contínua melhoria. As medidas obtidas devem ser armazenadas no repositório de medidas da organização (SOFTEX, 2011d);
- AP 4.1 – O processo é medido: evidencia o quanto os resultados de medição são usados para assegurar que a execução do processo atinge os seus objetivos de desempenho e apoia o alcance dos objetivos de negócio definidos. O objetivo deste atributo de processo é, portanto, garantir a existência de um sistema efetivo para coleta de medidas relevantes para o desempenho do processo e a qualidade dos produtos de trabalho (SOFTEX, 2011e);
- AP 4.2 – O processo é controlado: evidencia o quanto o processo é controlado estatisticamente para produzir um processo estável, capaz e previsível dentro de limites estabelecidos. O processo estável pode ser definido como um processo previsível, cujo desempenho e variabilidade são conhecidos, permitindo elaborar estimativas que utilizam como base seu desempenho passado. Para verificar se determinado processo é estável, torna-se necessário executá-lo, coletar uma quantidade significativa de medidas de desempenho e avaliar se o processo está sob controle estatístico (SOFTEX, 2011e);
- AP 5.1 – O processo é objeto de melhorias incrementais e inovações: evidencia o quanto as mudanças no processo são identificadas a partir da análise de defeitos, problemas, causas comuns de variação do desempenho e da investigação de enfoques inovadores para a definição e implementação do processo. Este atributo está relacionado à existência de um foco proativo na melhoria contínua do atendimento a objetivos de negócio projetados e relevantes da organização (SOFTEX, 2011f);
- AP 5.2 – O processo é otimizado continuamente: evidencia o quanto as mudanças na definição, gerência e desempenho do processo têm impacto efetivo para o alcance dos objetivos relevantes de melhoria do processo. Este atributo está relacionado à implementação de uma abordagem ordenada e proativa para identificação de mudanças de processo apropriadas e introdução dessas mudanças

de forma a minimizar interrupções indesejáveis na operação do processo (SOFTEX, 2011f).

Os níveis de maturidade estabelecem patamares de evolução de processos, caracterizando estágios de melhoria da implementação de processos na organização. O nível de maturidade em que se encontra uma organização permite prever o seu desempenho futuro ao executar um ou mais processos (SOFTEX, 2011). O MR-MPS define sete níveis de maturidade, conforme ilustrado na Figura 2.

A divisão em sete estágios tem o objetivo de possibilitar uma implementação e avaliação adequada às micros, pequenas e médias empresas. A possibilidade de se realizar avaliações considerando mais níveis também permite uma visibilidade dos resultados de melhoria de processos em prazos mais curtos (SOFTEX, 2011).

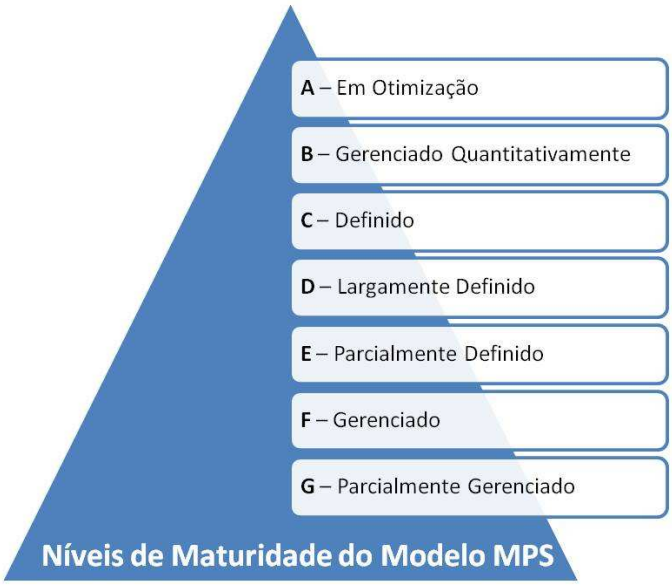

FIGURA 2 – Níveis de Maturidade definidos pelo Modelo de Referência MPS.
Fonte: Adaptado de SOFTEX (2011).

A escala de maturidade se inicia no nível G e progride até o nível A. Para cada um destes sete níveis de maturidade é atribuído um perfil de processos que indicam onde a organização deve colocar o esforço de melhoria. O progresso e o alcance de um determinado nível de maturidade do MR-MPS se obtêm quando são atendidos os propósitos e todos os resultados esperados dos respectivos processos e os resultados esperados dos APs estabelecidos para aquele nível (SOFTEX, 2011). Na Tabela 1 são listados os processos e APs exigidos para cada nível de maturidade.

TABELA 1 – Níveis de Maturidade do Modelo MPS com seus respectivos Processos e Atributos de Processo correspondentes.

Nível	Processos	Atributos de Processo (AP)
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; AP 5.2.
B	Gerência de Projetos – Evolução (GPR)	AP 1.1; AP 2.1; AP 2.2; AP 3.1; AP 3.2; AP 4.1; 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; AP 3.2.
D	Verificação (VER) Validação (VAR) 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; AP 3.2.
E	Gerência de Projetos – Evolução (GPR) Gerência de Reutilização (GRU) Gerência de Recursos Humanos (GRH) Definição de Processo Organizacional (DFP) Avaliação e Melhoria do Processo Organizacional (AMP)	AP 1.1; AP 2.1; AP 2.2; AP 3.1; AP 3.2.
F	Medição (MED) Garantia de Qualidade (GQA) Gerência de Portfólio de Projetos (GPP) Gerência de Configuração (GCO) Aquisição (AQU)	AP 1.1; AP 2.1; AP 2.2.
G	Gerência de Requistos (GRE) Gerência de Projetos (GPR)	AP 1.1; AP 2.1.

FONTE: SOFTEX (2011)

2.2.2 Nível de Maturidade G – Parcialmente Gerenciado

O nível G é o primeiro nível de maturidade do MR-MPS. Sua implementação deve ser executada com cautela por estabelecer o início dos trabalhos em implantação de melhoria dos processos de software na organização. Ao final da implantação deste nível a organização deve ser capaz de gerenciar parcialmente seus projetos de desenvolvimento de software (SOFTEX, 2011b).

O nível de maturidade G é composto pelos processos Gerência de Projetos (GPR) e Gerência de Requisitos (GRE). Neste nível, a implementação dos processos deve satisfazer os Atributos de Processo AP 1.1 e AP 2.1 (SOFTEX, 2011).

2.2.2.1 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. Assim, a partir do nível E, alguns resultados evoluem e outros são incorporados, de forma que a gerência de projetos passe a ser realizada com base no processo definido para o projeto e nos planos integrados. No nível B, a gerência de projetos passa a ter um enfoque quantitativo, refletindo a alta maturidade que se espera da organização. Novamente, alguns resultados evoluem e outros são incorporados (SOFTEX, 2011; SOFTEX, 2011b). Os resultados esperados são:

- GPR 1: o escopo do trabalho para o projeto é definido;
- GPR 2: as tarefas e os produtos de trabalho do projeto são dimensionados utilizando métodos apropriados;
- GPR 3: o modelo e as fases do ciclo de vida do projeto são definidos;
- GPR 4: (até o nível F) 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;
- GPR 4: (a partir do nível E) o planejamento e as estimativas das tarefas do projeto são feitos baseados no repositório de estimativas e no conjunto de ativos de processo organizacional;
- GPR 5: o orçamento e o cronograma do projeto, incluindo a definição de marcos e pontos de controle, são estabelecidos e mantidos;

- GPR 6: os riscos do projeto são identificados e o seu impacto, probabilidade de ocorrência e prioridade de tratamento são determinados e documentados;
- GPR 7: os recursos humanos para o projeto são planejados considerando o perfil e o conhecimento necessários para executá-lo;
- GPR 8: (até o nível F) os recursos e o ambiente de trabalho necessários para executar o projeto são planejados;
- GPR 8: (a partir do nível E) os recursos e o ambiente de trabalho necessários para executar os projetos são planejados a partir dos ambientes padrão de trabalho da organização;
- GPR 9: 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;
- GPR 10: um plano geral para a execução do projeto é estabelecido com a integração de planos específicos;
- GPR 11: a viabilidade de atingir as metas do projeto é explicitamente avaliada considerando restrições e recursos disponíveis. Se necessário, ajustes são realizados;
- GPR 12: o Plano do Projeto é revisado com todos os interessados e o compromisso com ele é obtido e mantido;
- GPR 13: o escopo, as tarefas, as estimativas, o orçamento e o cronograma do projeto são monitorados em relação ao planejado;
- GPR 14: os recursos materiais e humanos bem como os dados relevantes do projeto são monitorados em relação ao planejado;
- GPR 15: os riscos são monitorados em relação ao planejado;
- GPR 16: o envolvimento das partes interessadas no projeto é planejado, monitorado e mantido;
- GPR 17: revisões são realizadas em marcos do projeto e conforme estabelecido no planejamento;
- GPR 18: registros de problemas identificados e o resultado da análise de questões pertinentes, incluindo dependências críticas, são estabelecidos e tratados com as partes interessadas;

- GPR 19: 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;
- GPR 20: (a partir do nível E) equipes envolvidas no projeto são estabelecidas e mantidas a partir das regras e diretrizes para estruturação, formação e atuação;
- GPR 21: (a partir do nível E) experiências relacionadas aos processos contribuem para os ativos de processo organizacional;
- GPR 22: (a partir do nível E) um processo definido para o projeto é estabelecido de acordo com a estratégia para adaptação do processo da organização;
- GPR 22: (a partir do nível B) os objetivos de qualidade e de desempenho do processo definido para o projeto são estabelecidos e mantidos;
- GPR 23: (a partir do nível B) o processo definido para o projeto que o possibilita atender seus objetivos de qualidade e de desempenho é composto com base em técnicas estatísticas e outras técnicas quantitativas;
- GPR 24: (a partir do nível B) subprocessos e atributos críticos para avaliar o desempenho e que estão relacionados ao alcance dos objetivos de qualidade e de desempenho do processo do projeto são selecionados;
- GPR 25: (a partir do nível B) medidas e técnicas analíticas são selecionadas para serem utilizadas na gerência quantitativa;
- GPR 26. (a partir do nível B) o desempenho dos subprocessos escolhidos para gerência quantitativa é monitorado usando técnicas estatísticas e outras técnicas quantitativas;
- GPR 27: (a partir do nível B) o projeto é gerenciado usando técnicas estatísticas e outras técnicas quantitativas para determinar se seus objetivos de qualidade e de desempenho do processo serão atingidos;
- GPR 28: (a partir do nível B) questões que afetam os objetivos de qualidade e de desempenho do processo do projeto são alvo de análise de causa raiz.

2.2.2.2 Processo: Gerência de Requisitos (GRE)

O propósito do processo Gerência de Requisitos é controlar a evolução dos requisitos, ou seja, 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 (SOFTEX, 2011; SOFTEX, 2011b). Os resultados esperados são:

- GRE 1: o entendimento dos requisitos é obtido junto aos fornecedores de requisitos;
- GRE 2: os requisitos são avaliados com base em critérios objetivos e um comprometimento da equipe técnica com estes requisitos é obtido;
- GRE 3: a rastreabilidade bidirecional entre os requisitos e os produtos de trabalho é estabelecida e mantida;
- GRE 4: revisões em planos e produtos de trabalho do projeto são realizadas visando identificar e corrigir inconsistências em relação aos requisitos;
- GRE 5: mudanças nos requisitos são gerenciadas ao longo do projeto.

2.3 DESENVOLVIMENTO ÁGIL DE SOFTWARE

Desenvolvimento Ágil de Software é um processo evolutivo, altamente colaborativo, disciplinado, com abordagem no desenvolvimento de software focado em qualidade, onde software potencialmente utilizável é produzido em intervalos regulares de tempo para revisão e correção por parte dos *stakeholders* (AMBLER, 2009c).

O termo “Desenvolvimento Ágil” foi cunhado em 2001 no “Manifesto para o Desenvolvimento Ágil” (*The Manifesto for Agile Software Development*). O manifesto foi assinado por dezessete notáveis desenvolvedores, produtores e consumidores de software conhecidos como “Aliança Ágil” (*Agile Alliance*) (PRESSMAN, 2006), e declarou quatros valores e dose princípios do desenvolvimento ágil (BECK, 2001). Os valores são:

- Indivíduos e interações mais que processos e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano.

Os valores do Manifesto Ágil são suportados por um conjunto de doze princípios que exploram detalhadamente a base filosófica dos métodos ágeis de software (AMBLER, 2009c). Esses princípios são (BECK et al., 2001):

- Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor;
- Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adéquam a mudanças, para que o cliente possa tirar vantagens competitivas;
- Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos;
- Pessoas relacionadas a negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto;
- Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho;
- O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara;
- Software funcional é a medida primária de progresso;
- Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes;
- Contínua atenção a excelência técnica e bom design aumenta a agilidade;
- Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito. É essencial;
- As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis;
- Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e aperfeiçoam seu comportamento de acordo.

Os valores e os princípios do Manifesto Ágil fornecem uma base filosófica sólida para o desenvolvimento eficaz de software (AMBLER, 2009c).

A agilidade pode ser aplicada a qualquer processo de software. Entretanto, para conseguir isso, é essencial que o processo seja projetado de modo que permita à equipe de projeto adaptar tarefas e aperfeiçoá-las, conduzir o planejamento para que se entenda a fluidez de uma abordagem de desenvolvimento ágil, eliminar tudo menos os produtos de trabalho mais essenciais e mantê-los simples, e enfatizar uma estratégia de entrega incremental que forneça o software funcionando ao cliente o mais rápido possível para o tipo de produto e ambiente operacional (PRESSMAN, 2006).

Atualmente, existe um vasto número de Modelos de Processo Ágeis que satisfazem aos princípios do Manifesto Ágil. Há muitas semelhanças (em filosofia e prática) entre essas abordagens (PRESSMAN, 2006). Scrum, *Extreme Programming* (XP), *Feature Driven Development* (FDD) são alguns exemplos de modelos ágeis existentes.

2.4 AGILE UNIFIED PROCESS

O Processo Unificado Ágil (AUP - *Agile Unified Process*) é uma abordagem simplificada para o desenvolvimento de software com base no processo da IBM *Rational Unified Process* (RUP) [5]. O ciclo de vida do AUP é sequencial, iterativo e disponibiliza versões incrementais ao longo do tempo (AMBLER, 2009; AMBLER, 2006). A visão geral do AUP é ilustrada na Figura 3.

[5] O RUP (*Rational Unified Process*) é um processo de desenvolvimento de software proprietário da IBM que foi baseado no Processo Unificado (UP – *Unified Process*) criado por James Rumbaugh, Grady Booch e Ivar Jacobson combinando as melhores características de modelos convencionais de processo (ciclo de vida iterativo e incremental, arquitetura baseada em componentes, modelagem visual, etc.) e modelagem utilizando a abordagem da Orientação a Objetos (OO – *Object-Oriented*), através da notação UML *(Unified Modeling Language*) (PRESSMAN, 2006; KRUCHTEN, 2003).

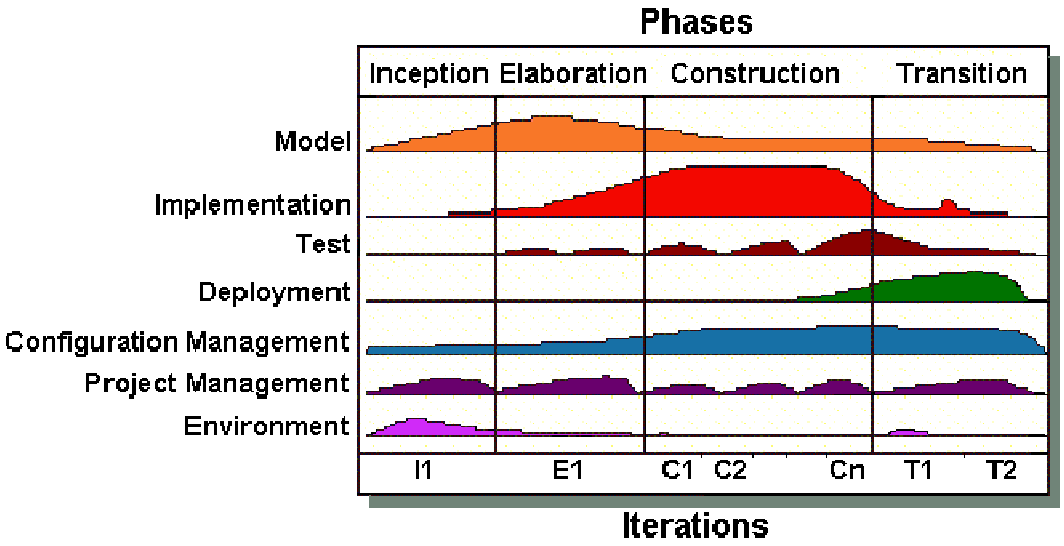

FIGURA 3 – Visão Geral do AUP.
Fonte: AMBLER (2006)

De acordo com a Figura 3, o eixo horizontal ilustra as fases do AUP e o eixo vertical as disciplinas, de modo que seja evidenciada a concentração de atividades de cada disciplina em cada fase do ciclo de vida do AUP.

2.4.1 Fases do AUP

As fases do ciclo de vida do AUP são realizadas de forma sequencial durante o projeto (AMBLER, 2009; AMBLER, 2006). São quatro fases:

- Iniciação *(Inception*): o objetivo é identificar o escopo inicial do projeto, uma arquitetura potencial para o sistema, e obter o financiamento inicial do projeto e aceitação dos *stakeholders*;
- Elaboração (*Elaboration*): o objetivo é provar a arquitetura do sistema;
- Construção (*Construction*): o objetivo é construir o software, trabalhando de forma incremental, que atenda às necessidades de maior prioridade dos *stakeholders*;
- Transição (*Transition*): o objetivo é validar e implantar o sistema no ambiente de produção.

A seguir uma descrição detalhada a respeito de cada fase do AUP.

2.4.1.1 Fase de Iniciação

Alcançar o consenso entre os *stakeholders* sobre os objetivos do projeto e obter financiamento são as principais metas da fase de Iniciação (AMBLER, 2006). As principais atividades da fase incluem:

- Definir o escopo do projeto. Isso inclui a definição, em alto nível, do que o sistema deverá fazer. Igualmente importante é definir o que o sistema não vai fazer. Isto estabelece os limites dentro dos quais a equipe irá operar;
- Estimar o custo e o cronograma. Em alto nível, o cronograma e o custo do projeto são estimados. As estimativas são utilizadas nas iterações em fases posteriores, mais especificamente nas iterações iniciais da fase de Elaboração. Isso não deve ser interpretado no sentido de que todo o projeto é planejado neste momento. As tarefas que devem ser concluídos em um futuro próximo são detalhadas de forma mais precisa, enquanto tarefas mais futuras são estimativas com margens maiores de erro. Segundo AMBLER (2006), foi finalmente reconhecido pela maioria na indústria que não é possível planejar um projeto inteiro em seu pontapé inicial (*kick-off*) com qualquer grau de confiança aceitável. O melhor que pode ser feito é planejar em curto prazo com precisão e em longo prazo da melhor forma possível;
- Definir riscos. Os riscos para o projeto são definidos primeiramente nesse ponto. A gestão de riscos é importante em um projeto AUP. A lista de riscos muda com o tempo assim que riscos são identificados, mitigados, evitados e/ou materializados e tratados. Riscos conduzem a gestão do projeto, como os riscos de maior prioridade conduzem a programação de iterações. Riscos de maior prioridade, por exemplo, são tratados em iterações iniciais, ao invés dos riscos de baixa prioridade;
- Determinar a viabilidade do projeto. O projeto deve fazer sentido na perspectiva técnica, operacional e de negócios. Em outras palavras, o sistema deve ser capaz de ser construído, e uma vez implantado, deve ser possível executá-lo. Além disso, o sistema deve ser economicamente viável, caso contrário, deve ser cancelado;

- Preparar o ambiente do projeto. Isso inclui reservar espaço de trabalho para a equipe, requisitar pessoas que serão necessárias, obter hardware e software que são necessários imediatamente, e compilar uma lista de hardware e software antecipado que será necessário mais tarde. Além disso, o AUP deve ser adaptado para atender às necessidades exatas de equipe.

Para finalizar a fase de Iniciação, a equipe deve passar pelo marco (*milestone*) de Objetivos do Ciclo de Vida (LCO - *Lifecycle Objectives*). As principais metas são comprovar se a equipe entende o esforço do escopo do projeto e o apoio dos *stakeholders* para o financiamento do projeto. Se a equipe passa este marco, o projeto passa para a fase de Elaboração, caso contrário, o projeto pode ser redirecionado ou cancelado (AMBLER, 2006).

2.4.1.2 Fase de Elaboração

O principal objetivo da fase de Elaboração é provar a arquitetura do sistema a ser desenvolvido. O ponto é garantir que a equipe possa realmente desenvolver um sistema que satisfaça os requisitos e a melhor maneira de fazer isso é construir um esqueleto do sistema chamado de "protótipo arquitetônico". O objetivo é escrever software de alta qualidade, que reúne vários casos de uso de alto risco (do ponto de vista técnico) e mostrar que o sistema é tecnicamente viável (AMBLER, 2006).

É importante notar que os requisitos não estão completamente especificados neste ponto. Eles são detalhados apenas o suficiente para compreender os riscos de arquitetura e para assegurar que existe um entendimento do escopo de cada requisito de modo que o planejamento subsequente possa ser realizado. Riscos arquitetônicos são identificados e priorizados e os mais significativos são abordados durante essa fase (AMBLER, 2006).

Durante a fase de Elaboração, a equipe também deve ser preparar para a fase de Construção. Como a equipe ganha experiência com a arquitetura do sistema, eles começam a configurar o ambiente para a fase de Construção, obtendo hardware, software e ferramentas. Do ponto de vista do gerenciamento de projetos,

a equipe é alocada e os recursos são solicitados e / ou contratados. Planos de comunicação e colaboração são finalizados (especialmente importante se a equipe deve ser fisicamente distribuída) (AMBLER, 2006).

Para finalizar a fase de Elaboração, a equipe deve passar pelo marco Arquitetura do Ciclo de Vida (LCA - *Lifecycle Architecture*). As principais questões abordadas são se a equipe desenvolveu um protótipo funcional da arquitetura que mostra se a equipe tem uma estratégia viável para a construção do sistema e se os *stakeholders* estão dispostos a continuar com o financiamento do projeto. Se a equipe passar por esse marco, o projeto vai para a fase de Construção, caso contrário, o projeto pode ser redirecionado ou cancelado (AMBLER, 2006).

2.4.1.3 Fase de Construção

O foco da fase de Construção é desenvolver o sistema até o ponto em que esteja pronto para o teste de pré-produção (homologação). Nas fases anteriores, a maioria dos requisitos já foi identificada e a arquitetura do sistema foi estabelecida. A ênfase se move agora para a priorização e entendimento dos requisitos, modelagem da solução (através da utilização da técnica de *Model Storming*[6]), e, em seguida, codificação e testes do software. Se necessário, as primeiras versões do sistema são implantadas, seja internamente ou externamente, para obter o *feedback* do usuário (AMBLER, 2006).

Para fechar da fase de Construção, a equipe deve passar pelo marco Capacidade Operacional Inicial (IOC - *Initial Operating Capacity*). A questão principal é saber se a versão corrente do sistema está pronta para a implantação no ambiente de teste de pré-produção para os testes de sistema e aceitação. Se a equipe passa por este marco, o projeto vai para a fase de Transição, caso contrário pode ser redirecionado ou cancelado (AMBLER, 2006).

---

[6] *Model Storming: Just in Time (JIT) Modeling*, também chamada de Sessão de Modelagem Ágil, é uma atividade na qual diversas pessoas focalizam o desenvolvimento de um ou mais modelos. As pessoas se reúnem em torno de uma ferramenta de modelagem compartilhada (por exemplo, um quadro branco), e exploram o problema até que estejam convencidos de que o entenderam. *Model Storming* é uma técnica utilizada pelo Desenvolvimento Dirigido por Modelagem Ágil (AMDD – *Agile Model Driven Development*) (AMBLER, 2004; AMBLER, 2009b).

2.4.1.4 Fase de Transição

A fase de Transição concentra-se em entregar o sistema em produção. Pode haver testes extensivos que ocorrem durante esta fase, incluindo testes beta[7]. O ajuste dos produtos, bem como o retrabalho para tratar defeitos significativos (os *stakeholders* podem optar por aceitar a existência de alguns defeitos conhecidos na versão corrente), são feitos nessa fase (AMBLER, 2006).

O tempo e o esforço despendido na fase de Transição variam de projeto para projeto. O pacote do software implica na manutenção, distribuição e documentação do software. Sistemas internos são geralmente mais simples de implantar do que sistemas externos. Sistemas de alta visibilidade podem exigir testes beta extensivos por pequenos grupos antes da liberação para a população maior (AMBLER, 2006).

Para fechar a fase de Transição, a equipe deve passar pelo marco Lançamento do Produto (PR – *Product Release*). A questão principal é saber se o sistema pode ser implantado com segurança e eficácia em produção. Se a equipe passa por este marco, o projeto é movido para produção. Se o projeto falhar, ele pode ser redirecionado ou cancelado (AMBLER, 2006).

2.4.2 Marcos do AUP

O AUP possui quatro marcos que sinalizam o final de cada fase. Em cada um deles deve existir uma "Revisão de Marco", que verifica se a equipe cumpriu com êxito os critérios dos marcos. A Figura 4 ilustra as fases e os marcos do AUP (AMBLER, 2006).

[7] *Beta Testing*, também chamado de Teste Beta, tem o objetivo de reduzir os impactos para os usuários, para que posteriormente o *software* esteja disponível para os *stakeholders*.

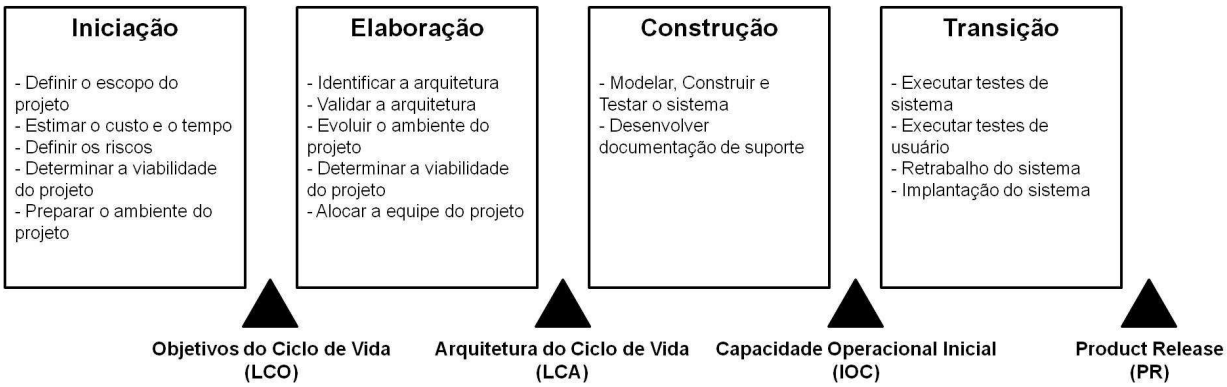

FIGURA 4 – Fases e Marcos do AUP.
Fonte: Adaptado de AMBLER (2006)

A seguir uma breve descrição de cada marco.

2.4.2.1 Marco da Fase de Iniciação: Objetivos do Ciclo de Vida (LCO)

No marco LCO (*Lifecycle Objectives*), presente no final da fase de Iniciação, os *stakeholders* avaliam o estado do projeto (AMBLER, 2006). Eles devem concordar com o seguinte:

- Concordância do escopo. Os *stakeholders* chegam a um acordo quanto ao escopo do projeto;
- Definição inicial dos requisitos. Há um consenso de que um conjunto correto de requisitos foi levantado, em alto nível, e há uma compreensão compartilhada desses requisitos;
- Concordância do plano. Os *stakeholders* estão de acordo com o custo inicial e estimativas de tempo;
- Aceitação dos riscos. Os riscos foram identificados, avaliados e estratégias aceitáveis para mitigá-los foram identificadas;
- Aceitação do processo. O AUP foi inicialmente adaptado e aprovado por todas as partes;
- Viabilidade. O projeto faz sentido nas perspectivas de negócio, técnico e operacional;
- Plano de projeto. Planos adequados estão em vigor para a próxima fase (Elaboração);

- Adequação ao portfólio. O escopo do projeto está de acordo com portfólio geral de projetos da empresa.

2.4.2.2 Marco da Fase de Elaboração: Arquitetura do Ciclo de Vida (LCA)

No marco LCA (*Lifecycle Architecture*), presente no final da fase de Elaboração, os *stakeholders* avaliam o estado do projeto (AMBLER, 2006). Eles devem concordar com o seguinte:

- Visão estabilizada. A visão do projeto é estabilizada e realista;
- Arquitetura estável. A arquitetura é estável o suficiente para satisfazer os requisitos. A arquitetura foi prototipada onde necessário para mitigar os principais riscos arquitetônicos;
- Aceitação dos riscos. Os riscos foram avaliados para garantir que eles tenham sido corretamente entendidos e documentados, e as estratégias para lidar com eles são aceitáveis;
- Viabilidade. O projeto faz sentido nas perspectivas de negócio, técnico e operacional;
- Plano de projeto. Planos de iteração detalhados para as próximas iterações da fase de Construção, bem como um plano de projeto de alto nível, estão definidos;
- Cumprimento Empresarial. A arquitetura do sistema reflete a realidade da arquitetura corporativa.

2.4.2.3 Marco da Fase de Construção: Capacidade Operacional Inicial (IOC)

No marco IOC (*Initial Operating Capacity*), presente no final da fase de Construção, os *stakeholders* devem concordar com o seguinte (AMBLER, 2006):

- Estabilidade do sistema. O software e a documentação de apoio são aceitáveis (estáveis e maduros) para implantar o sistema para os usuários;

- *Stakeholders* preparados. Os *stakeholders* (e o negócio) estão prontos para o sistema ser implantado (embora talvez ainda seja necessário treinamento);
- Aceitação dos riscos. Os riscos foram avaliados para garantir que eles tenham sido corretamente entendidos e documentados e estratégias para lidar com eles são aceitáveis;
- Aceitação dos custos e estimativas. As despesas correntes são estimadas aceitavelmente e razoavelmente feitas para os futuros custos e cronogramas;
- Plano de projeto. Planos de iteração detalhados para as próximas iterações da fase de Transição, bem como um plano de projeto de alto nível, estão definidos;
- Cumprimento Empresarial. Os produtos de trabalho produzidos pela equipe cumprem com as normas empresariais adequadas.

2.4.2.4 Marco da Fase de Transição: Lançamento de Produto (PR)

No marco PR (*Product Release*), presente no final da fase de Transição, os *stakeholders* devem concordar com o seguinte (AMBLER, 2006):

- Aceitação dos *stakeholders* especialistas no domínio do negócio. Os *stakeholders* especialistas no domínio do negócio estão satisfeitos e aceitam o sistema;
- Aceitação de operações. As pessoas responsáveis pela operação do sistema estão satisfeitos com os procedimentos e documentação;
- Aceitação do suporte. As pessoas responsáveis pelo suporte ao sistema estão satisfeitos com os procedimentos e documentação;
- Aceitação dos custos e estimativas. As despesas correntes são estimadas aceitavelmente e razoavelmente feitas para os futuros custos;

2.4.3 Disciplinas do AUP

As disciplinas do AUP são executadas de forma iterativa, definindo as atividades que os membros da equipe de desenvolvimento devem realizar para construir, validar e entregar o software de forma que atenda as necessidades do negócio (AMBLER, 2009; AMLBER, 2006). As disciplinas do AUP são:

- Modelo (*Model*)
- Implementação (*Implementation*)
- Teste (*Test*)
- Implantação (*Deployment*)
- Gerenciamento de Configuração (*Configuration Management*)
- Gerenciamento de Projetos (*Project Management*)
- Ambiente (*Environment*)

A seguir, uma descrição detalhada de cada disciplina do AUP.

2.4.3.1 Disciplina: Modelo

O objetivo da disciplina de Modelo (*Model*) é entender o negócio da organização, o domínio do problema a ser abordado pelo projeto, e identificar uma solução viável para resolvê-lo. De acordo com a fase em que o projeto se encontra durante o ciclo de vida, as atividades da disciplina variam (AMBLER, 2006):

- Na fase de Iniciação, a modelagem de requisitos deve ser inicial e de alto nível e deve contar com a participação ativa dos especialistas do domínio ou do negócio para definição do escopo inicial do projeto, que fornece informações suficientes para uma estimativa aproximada (AMBLER, 2006). Deve-se considerar:

o Escrever casos de uso com informações suficientes para dar uma ideia geral;

o Identificar o(s) processo(s) de negócio através da criação de Diagramas de Fluxo de Dados (DFD – *Data Flow Diagram*);

- Identificar as principais entidades de negócio e suas relações, trabalhando em um modelo de domínio simples;
- Identificar as principais regras de negócios e requisitos técnicos. Nesse momento, o nome das entidades de negócio, normas e requisitos técnicos são suficientes;
- Iniciar o desenvolvimento de um glossário descrevendo os termos técnicos e de negócio importantes;
- Compreender a estrutura organizacional da empresa;
- Tratar os requisitos como uma pilha hierarquizada, que evolui ao longo do tempo. Nela estão os casos de uso, regras de negócio e requisitos técnicos. Através da hierarquização, esta abordagem dá suporta a gestão de mudanças.

- Na fase de Elaboração são identificados os riscos técnicos do projeto. Os requisitos dos produtos de trabalho, em particular os casos de uso e requisitos técnicos, revelam potenciais riscos técnicos para o projeto. Estes riscos podem incluir a introdução de novas tecnologias na organização, uma nova utilização de tecnologias existentes ou interface com sistemas externos / legados. Os riscos de maior prioridade devem ser abordados durante todo desenvolvimento do sistema (AMBLER, 2006). Deve-se considerar:
  - Modelagem arquitetônica. Ao construir o protótipo da arquitetura, será necessário construir alguns modelos para abordar partes individuais da arquitetura;
  - Prototipação de interfaces de usuário. Em paralelo com o desenvolvimento do protótipo arquitetônico, deve também ser considerada a construção de protótipos de interface de usuário das principais telas do sistema. Não é necessário fazer muitos protótipos, pois suas necessidades são suscetíveis à mudança e, portanto, o trabalho terá que ser descartado. O objetivo, no momento, deve ser o de compreender as principais telas / páginas da interface de usuário com o entendimento de que eles vão mudar durante a fase de Construção, e para identificar a aparência (*look and feel*) básica do sistema.

- Na fase de Construção, é utilizada a técnica de análise de *Model Storming*. Durante as iterações, é necessário trabalhar com a colaboração dos

stakeholders para entender seus requisitos (AMBLER, 2006). Questões importantes são:

- A participação ativa dos *stakeholders* e inclusão de ferramentas e técnicas simples de modelagem são fundamentais;
- Avaliar a possibilidade de esclarecer os detalhes dos casos de uso visualmente, utilizando fluxogramas ou diagramas de atividade da UML ao invés de descrições de texto. Explorar as regras de negócio e requisitos técnicos da mesma maneira;
- Pode ser necessário construir interfaces com sistemas existentes ou banco de dados legados;
- Ao invés de descrições de casos de uso, descrições de regras de negócios, e descrições de requisitos técnicos, é mais eficaz escrever casos de teste de aceitação em seu lugar. Assim, não é necessário explorar os requisitos tanto em um documento de requisitos como em uma descrição de teste;
- Como a interface do usuário é o sistema de muitos dos *stakeholders*, provavelmente será descoberto que eles preferem se concentrar no que as telas e relatórios se parecem em vez do desenvolvimento de outros produtos de trabalho. Portanto, é necessário prototipar sempre que possível;
- O glossário do projeto deve ser mantido atualizado;

- Durante as iterações da fase de Construção, a meta é fazer apenas modelagem (utilizando a técnica de *Model Storming*) suficiente para pensar na concepção do requisito, ou parte dele, antes de implementá-lo. Modeladores Ágeis modelam diretamente com os desenvolvedores, e não simplesmente entregam modelos. Além disso, muitas vezes assumem o papel de desenvolvedor (AMBLER, 2006). Podem ser criados os seguintes modelos:
  - Diagramas de Sequencia da UML. Estes diagramas representam a lógica dinâmica do código-fonte. Geralmente são descartados, a menos que seja utilizada uma boa ferramenta CASE com capacidades de engenharia reversa. Também podem ser utilizados quadros como ferramentas para criação de novos diagramas (POW – *Plain Old Whiteboads*);
  - Modelo de implantação. Normalmente é preciso criar algum tipo de "diagrama de visão geral" que retrata a arquitetura de implantação / rede do sistema;

- Diagramas de forma livre. Normalmente criados em quadros e, depois, apagados quando não mais necessários. Documentos de Visão Geral do sistema normalmente incluem vários diagramas de forma livre;
- Diagramas de classe da UML. Para fazer qualquer classe de diagramação, AMBLER (2006) recomenda o uso de uma ferramenta de modelagem que gera o código-fonte;
- Modelo de ameaça à segurança. Se as questões de segurança são uma preocupação, então deve ser considerado modelá-las para esclarecer ameaças potenciais, bem como a forma de lidar com essas ameaças;
- Modelos físicos de dados. Segundo AMBLER (2006), é o modelo mais importante do projeto, que deve ser considerado o uso de uma ferramenta CASE para desenvolver e manter ao longo do tempo, especialmente uma ferramenta que gere código DDL (*Data Definition Language*).

As decisões críticas do projeto devem ser documentadas. Assim que são tomadas, deve ser considerado gravar qualquer informação que não pareça óbvia, ou que no futuro alguém realmente gostaria de saber (AMBLER, 2006).

- Na fase de Transição, novamente é utilizada a técnica de *Model Storming* para tentar entender a causa de um defeito. Além disso, é finalizada a documentação de visão geral do sistema. De acordo com AMBLER (2006), o melhor momento para finalizá-la é durante esta fase quando o escopo do sistema está realmente estabilizado. Na documentação pode ser adicionado um resumo do alcance do sistema e os diagramas críticos de arquitetura.

2.4.3.2 Disciplina: Implementação

O objetivo da disciplina de Implementação é transformar o(s) modelo(s) em código executável e executar um nível básico de testes, em especial, testes de unidade. De acordo com a fase em que o projeto se encontra durante o ciclo de vida, as atividades da disciplina variam (AMBLER, 2006).

- Durante a fase de Iniciação, são realizadas as atividades:

o Prototipagem técnica. Pode ser preciso extrair um pequeno aspecto de um requisito, a fim de entendê-lo suficientemente, o que permite estimar o esforço necessário para sua construção. Estes protótipos são tipicamente pequenos e, posteriormente descartados, uma vez que possuem a função apenas de esclarecer dúvidas técnicas;

o Prototipação de interface de usuário. Para a maioria dos *stakeholders*, a interface de usuário (UI – *User Interface*) - as telas, relatórios e manuais - é o sistema. Quando a interface do usuário é potencialmente complexa, ou quando os *stakeholders* querem ver o que eles vão ter antes de comprá-lo, deve ser considerada a prototipagem, pelo menos das páginas / telas. O protótipo da interface de usuário seria usado para convencer os *stakeholders* que suas necessidades foram entendidas.

- Na fase de Elaboração, a arquitetura deve ser provada. A atividade crítica dentro desta fase é identificar uma arquitetura potencial e então provar que ela funciona com o desenvolvimento do protótipo do sistema, diminuindo assim a maior parte do risco técnico do projeto (AMBLER, 2006).
- Na fase de Construção, primeiro é realizado o teste. É utilizada a abordagem baseada em TDD[8] (*Test Driven Development*) para todos os aspectos da implementação (AMBLER, 2006). As atividades são:

o A Construção é realizada continuamente. De acordo com AMBLER (2006), compilações diárias é um bom começo, mas o ideal é gerar um *build* [9] do sistema sempre que o código-fonte mude. Além disso, automatizar esse processo usando uma ferramenta que monitora o sistema de controle de versão em caso de alterações do código-fonte e realiza um novo *build*, traz agilidade;

o Evoluir a lógica de domínio. Implementar a lógica de negócio nas classes de domínio / negócio do sistema;

o Evoluir a interface do usuário. A interface de usuário é o sistema para a maioria dos *stakeholders*, por isso, é necessário tornar o software mais usável

[8] Desenvolvimento Dirigido por Testes (TDD – *Test Driven Development*) é uma abordagem para o desenvolvimento de software onde é escrito primeiramente um caso de teste automatizado que define uma melhoria ou uma nova funcionalidade e em seguida, é produzido o código-fonte para que o teste seja validado e posteriormente o código passe por refatoração (*refactoring*) (BECK, 2003).

[9] O processo de *build* é a compilação do código-fonte do sistema para execução em uma configuração específica. Um *build* é a versão de um sistema.

possível, utilizando os conceitos de usabilidade e estratégias de projetos de interface de usuário;

o Evoluir o esquema de dados. O esquema de dados deve ser criado em conjunto com o domínio e código de interface de usuário. Deve ser realizado o *refactor* [10] do banco de dados assim que realizado o *refactor* dos outros tipos de código;

o Desenvolver interfaces para acesso legado. Conforme os requisitos do sistema, pode ser necessário o acesso a funcionalidades existentes de sistemas legados. Isto pode ser feito através de uma variedade de meios, incluindo *web services*, procedimentos armazenados, e assim por diante;

o Escrever *scripts* de conversão de dados. Em casos onde é necessário acessar fontes de dados legadas que muitas vezes sofrem de problemas de projeto e/ou qualidade, AMBLER (2006) sugere que sejam escritos scripts de conversão de dados para formatar o(s) dado(s) em um formato utilizável.

- Na fase de Transição, a principal atividade é a correção de defeitos. O foco é deslocado para correção dos defeitos encontrados como resultado dos testes (AMBLER, 2006).

2.4.3.3 Disciplina: Teste

O objetivo da disciplina de Teste é a realização de uma avaliação objetiva para garantir a qualidade do produto. Isto inclui encontrar defeitos, validando se o sistema funciona conforme projetado e verificar se os requisitos são cumpridos. De acordo com a fase em que o projeto se encontra durante o ciclo de vida, as atividades da disciplina variam (AMBLER, 2006).

- Durante a fase de Iniciação, são realizadas as atividades (AMBLER, 2006):

[10] *Refactoring* é uma técnica disciplinada para reestruturar um código-fonte existente, alterando a sua estrutura interna sem alterar seu comportamento externo (FOWLER, 1999).

- Planejamento inicial do teste. Inicialmente deve ser em alto nível. O objetivo principal é identificar quantos testes serão necessários, quem será o responsável por isso, o nível necessário de participação dos *stakeholders*, e os tipos de ferramentas e ambientes necessários;
- Análise inicial do gerenciamento dos produtos de trabalho do projeto. Ao final desta fase, o plano de projeto inicial e visão devem estar disponíveis. Estes produtos de trabalho são frequentemente revisados, normalmente como parte das revisões de marco pelos *stakeholders* chave do projeto;
- Revisão dos modelos iniciais. Pode ser optado por rever os modelos iniciais (por exemplo, modelo de arquitetura e requisitos) com os *stakeholders*, especialmente se será comunicado o escopo e arquitetura potencial do sistema para uma ampla gama de pessoas que estiverem ativamente envolvidas no desenvolvimento dos modelos.

- Durante a fase de Elaboração, são realizadas as atividades (AMBLER, 2006):
  - Validar a arquitetura. Deve ser utilizada a abordagem de Desenvolvimento Dirigido por Testes (TDD) para a construção do protótipo técnico que comprova a arquitetura do sistema. Um aspecto importante da revisão de marco é a validação da arquitetura, que pode ser algo tão simples como uma apresentação de sua visão geral e os resultados dos esforços de prototipagem para os *stakeholders*, ou, pode ser algo tão complexo como uma revisão formal de todo o trabalho durante esta fase;
  - Evoluir o modelo de teste. A equipe irá desenvolver um conjunto de testes de regressão, composto de testes de unidade dos esforços de TDD durante a implementação, os testes de aceitação dos esforços de modelagem e os testes do sistema. Poderá ser preciso também manter a rastreabilidade entre os requisitos, testes e código-fonte para mostrar como foram validados os requisitos implementados.

- Durante a fase de Construção, o software é testado. Além de testes de unidade pelos desenvolvedores, será necessário fazer testes de instalação dos *scripts* de implantação, esforços de teste de sistema, tais como testes de carga /

stress e testes de aceitação do usuário. Deve-se evoluir também o modelo de teste (AMBLER, 2006).

- Durante a fase de Transição, são realizadas as atividades (AMBLER, 2006):

o Validar o sistema. O foco será nos testes de sistemas, testes de integração, testes de aceitação e teste beta. O objetivo é testar completamente o sistema dentro do(s) seu(s) ambiente(s) de teste de pré-produção (homologação);

o Validar a documentação. A documentação do sistema (documento de visão geral de sistema, de usuário, de suporte, e de operações), e materiais de treinamento deverá ser validada. Isto pode ser feito nas revisões ou como parte do teste beta;

o Finalização do modelo de teste. O conjunto de testes de regressão deve continuar e ser atualizado conforme necessário, até que o sistema esteja pronto para ser implantado em produção. O relatório de defeitos vai se tornar mais formal, os defeitos encontrados serão registrados junto com detalhes adequados, de modo que os desenvolvedores poderão realizar as correções.

2.4.3.4 Disciplina: Implantação

O objetivo da disciplina de Implantação é planejar a entrega do sistema e a execução do plano para tornar o sistema disponível para os usuários finais. De acordo com a fase em que o projeto se encontra durante o ciclo de vida, as atividades da disciplina variam (AMBLER, 2006):

- Durante a fase de Iniciação, são identificadas as janelas de lançamento (*releases*[11]) potenciais. Definir a janela de lançamento no início do projeto ajuda nos esforços de planejamento. Além disso, é iniciado o planejamento de implantação em alto nível. Este esforço incidirá sobre o planejamento da primeira *release* do sistema, identificando o seu público e a janela potencial do lançamento. O principal objetivo deve ser o de determinar uma estratégia de implantação geral: com base na

[11] A *release* de um sistema é um conjunto de itens que são entregues ao usuário.

compreensão do projeto e na viabilidade de liberar o software de uma só vez ou em *releases* separadas (AMBLER, 2006).

- Durante a fase de Elaboração, é atualizado o plano de implantação. Uma parte importante da definição da arquitetura é definir as configurações de implantação para o sistema. Cada configuração de implantação individual poderia ser documentada de acordo com algum tipo de modelo de implantação que define como os componentes de software reais são organizados e implantados nos componentes de hardware. Compreender as configurações ajuda na identificação dos diferentes tipos de instalações que é necessário fazer, o que por sua vez, deve fornecer uma visão geral sobre o esforço de implantação (AMBLER, 2006).
- Durante a fase de Construção: são realizadas as atividades (AMBLER, 2006):

o Desenvolver *scripts* de instalação e desinstalação. À medida que o sistema é desenvolvido, deve também ser escrito e testado os scripts de instalação e de desinstalação necessários para implantação do sistema no ambiente de teste de pré-produção. Este script deve ser escrito de modo que possa ser simplesmente reconfigurado para implantação em produção;

o Desenvolver notas de lançamento. As notas de lançamento devem resumir as "coisas boas para saber" sobre a versão atual do sistema que está sendo construída;

o Desenvolver documentação inicial. Além de entregar o software, será necessário também entregar a documentação do sistema (operações, suporte, visão geral do sistema e documentação de usuário), bem como os materiais de treinamento. Não deve ser colocada muita informação de trabalho nos documentos nesse momento, pois o sistema ainda está em evolução;

o Atualizar o plano de implantação. À medida que o desenvolvimento do sistema progride, o plano de implantação também deve evoluir. Muitas vezes será preciso negociar o plano de implantação com as áreas de operações e os serviços de apoio, bem como com outros projetos que também estão planejando a implantação de software, de modo que o projeto se encaixe no plano de implantação de sistemas corporativo;

o Implantar o sistema em ambientes de pré-produção (homologação). O sistema deve ser implantado regularmente em ambientes de pré-produção para

testes, bem como para demonstrações aos *stakeholders*. Quanto mais experiência for obtida fazendo isso, mais fácil a real implantação em produção é provável que seja.

- Durante a fase de Transição, são realizadas as atividades (AMBLER, 2006):

o Finalizar o pacote de implantação. Para finalizar o empacotamento do software, é preciso definir *baselines*[12] de implantação, e realizar o *build* final do software;

o Finalizar a documentação. A maioria da documentação do sistema (operações, suporte, visão geral do sistema e documentação de usuário) é tipicamente escrita durante esta fase, pois a funcionalidade do sistema está estabilizada;

o Anunciar a implantação. Deve ser anunciado o cronograma de implantação previsto, incluindo as datas previstas de treinamento e instalação. É preciso treinar a operação, suporte e comunidades de usuários conforme apropriado durante esta fase;

o Treinar as pessoas. Treinar clientes do projeto - usuários, gerentes, pessoal de operações e pessoal de suporte - é sempre uma parte importante da implantação;

o Implantar o sistema em produção. Neste ponto, será executada qualquer conversão de dados necessária, que pode ser um trabalho de uma só vez ou uma conversão gradual dos dados. Pode ser optado por executar o novo sistema em paralelo com o sistema existente por várias semanas para garantir que ele realmente funciona em produção. Pode ser decidido também por implantar o sistema para um subconjunto de usuários, chamado de lançamento piloto, para verificar se ele funciona para o pequeno grupo antes de optar por implantar o sistema para todos. Pode ser preciso implantar uma versão do software para o departamento de suporte para que eles possam simular problemas de produção quando os usuários contatá-los para obter ajuda.

[12] Uma *baseline* é uma imagem de uma versão de cada artefato no repositório do projeto.

2.4.3.5 Disciplina: Gestão de Configuração

O objetivo da disciplina de Gestão de Configuração é gerir o acesso aos produtos de trabalho do projeto. Isso inclui não apenas controlar as versões dos produtos de trabalho ao longo do tempo, mas também controlar e gerenciar suas alterações. De acordo com a fase em que o projeto se encontra durante o ciclo de vida, as atividades da disciplina variam (AMBLER, 2006):

- Durante a fase de Iniciação, é realizada a configuração do ambiente. Para isso são realizadas as seguintes atividades (AMBLER, 2006):

  o O repositório de Gestão de Configuração (CM - *Configuration Management*) deve ser instalado, caso não tenha sido anteriormente;

  o A estrutura de pastas/diretórios apropriada, que deve seguir a orientação existente, precisa ser criada para a equipe do projeto;

  o Membros da equipe do projeto deverão receber permissão de acesso às pastas do projeto e ter instalado qualquer software necessário para o projeto em suas estações de trabalho;

  o Membros da equipe de projeto podem também precisar ser treinados em conceitos básicos CM e nas ferramentas utilizadas;

Todos os produtos de trabalho devem ser colocados sob controle do repositório de Gestão de Configuração, além do trabalho realizado por todos da equipe. Os dados devem ser enviados para o repositório (*checking-in*) e/ou obtidos (*checking-out*) conforme a necessidade, resolvendo conflitos de atualização e definindo uma *baseline* dos produtos de trabalho quando versões principais são acordadas.

Nas fases posteriores (Elaboração, Construção, Transição), os produtos de trabalho devem ser mantidos sob controle do CM.

2.4.3.6 Disciplina: Gestão de Projeto

O objetivo da disciplina de Gestão de Projeto é a de dirigir as atividades que acontecem no projeto. Isto inclui a gestão de riscos, o direcionamento das pessoas (atribuição de tarefas, monitoramento de progresso, etc.), e a coordenação das pessoas e dos sistemas fora do escopo do projeto para ter certeza de que seja entregue no prazo e dentro do orçamento. De acordo com a fase em que o projeto se encontra durante o ciclo de vida, as atividades da disciplina variam (AMBLER, 2006).

- Durante a fase de Iniciação, são realizadas as seguintes atividades (AMBLER, 2006):

o Iniciar a construção da equipe. Neste ponto, é preciso de alguém com habilidades de modelagem para trabalhar com os *stakeholders* para identificar os requisitos iniciais do sistema e com as pessoas técnicas para identificar uma arquitetura potencial;

o Construir relacionamentos com os *stakeholders* do projeto. O apoio e a participação dos *stakeholders* são fundamentais para o sucesso do projeto;

o Determinar a viabilidade do projeto. O projeto deve ser financeiramente, tecnicamente, operacionalmente e politicamente viável. Em outras palavras, ele deve fazer sentido do ponto de vista comercial, deve ser possível construí-lo, deve ser possível mantê-lo em funcionamento uma vez que implantado em produção, e sua organização deve ser capaz de tolerar o projeto;

o Desenvolver um cronograma de alto nível de todo o projeto. O cronograma deve mostrar o esforço do projeto organizado em iterações, indicando as revisões dos marcos mais importantes, tarefas que abordam dependências críticas que a equipe possui com outras equipes, e a data final prevista do projeto;

o Desenvolver um plano de iteração detalhado para a próxima iteração. O planejamento detalhado é realizado apenas no momento (JIT – *Just in Time*). Quando há grandes eventos, como uma revisão ou uma tarefa que aborda uma dependência crítica do projeto, na próxima iteração o gerente de projeto deve trabalhar com as pessoas afetadas previamente para planejar as atividades apropriadas. Com relação ao planejamento detalhado para uma iteração, a melhor

maneira é simplesmente reunir a equipe no início de uma iteração e trabalhar com eles para planejar o seu trabalho. Segundo AMBLER (2006), a pessoa que vai fazer o trabalho muitas vezes é a pessoa mais adequada para planejá-la;

- o Gerenciar riscos. Há sempre riscos em um projeto de desenvolvimento de software: de negócio, técnicos e organizacionais. Durante todo o projeto, devem ser identificados os riscos e desenvolver estratégias para mitigá-los;
- o Obter apoio e financiamento dos *stakeholders*. O escopo do projeto deve estar entendido, viável, com os riscos compreendidos, e possuir um plano de viabilidade para prosseguir. Deve ser trabalhado com os *stakeholders* antes da revisão de marco para ganhar o seu apoio, caso contrário, haverá o risco de falhar na revisão;
- o Fechar a fase. Será preciso manter a revisão do marco de Objetivos do Ciclo de Vida (LCO), cujo principal objetivo é formalizar o apoio dos *stakeholders* para o projeto.

- Durante a fase de Elaboração, são realizadas as seguintes atividades (AMBLER, 2006):
  - o Construir a equipe. A medida que o projeto prossegue, pode ser necessário adicionar pessoas a ele. Durante esta fase as pessoas do projeto precisam ter habilidades de arquitetura e implementação, frequentemente sendo necessário treinamento para as pessoas no desenvolvimento de novas habilidades;
  - o Proteger a equipe. A política é uma triste realidade em qualquer organização, e um bom gerente de projeto deve proteger os membros da equipe tanto quanto possível;
  - o Obtenção de recursos. A equipe precisa de financiamento, instalações (salas, mesas), hardware, software, e assim por diante para desempenhar seus trabalhos;
  - o Gerenciar riscos. Os esforços na gestão de riscos devem continuar;
  - o Atualizar o plano do projeto. Os esforços no planejamento devem continuar;
  - o Fechar a fase. Será preciso manter a revisão do marco de Arquitetura do Ciclo de Vida (LCA), cujo principal objetivo é mostrar que a arquitetura funciona e que aborda os principais riscos técnicos enfrentados pela equipe de projeto.

- Durante a fase de Construção, são realizadas as seguintes atividades (AMBLER, 2006):
  - Gerenciar a equipe. Continuar a evolução da equipe (desenvolvedores e testadores são mais necessários) e continuar com a proteção para obter os recursos que eles precisarem;
  - Gerenciar riscos. Continuar os esforços de gestão de risco;
  - Atualizar o plano do projeto. Durante a fase de Construção deve ser assegurado que as principais dependências envolvidas com a implantação bem-sucedida do sistema tenham sido identificadas. Devem ser consideradas as necessidades de apoio e grupos de operações, treinamento do usuário final, e planejamento do teste beta;
  - Fechar a fase. Será preciso manter a revisão do marco de Capacidade Operacional Inicial (IOC), cujo principal objetivo é mostrar que a equipe desenvolveu um sistema que está potencialmente pronto para ser implantado em produção.

- Durante a fase de Transição, são realizadas as seguintes atividades (AMBLER, 2006):
  - Gerenciar a equipe. A equipe deve incluir, na sua maioria, desenvolvedores, testadores e responsáveis pela implantação;
  - Fechar a fase. Será preciso manter a revisão do marco de Lançamento do Produto (PR), cujo principal objetivo é mostrar que o sistema passou no teste e é aceitável para os *stakeholders*;
  - Iniciar o próximo ciclo do projeto. Os sistemas são desenvolvidos e implantados em produção de forma incremental. Durante a fase de transição da *release* N, deve ser iniciado o esforço da iniciação do projeto para a release N +1.

2.4.3.7 Disciplina: Ambiente

O objetivo da disciplina de Ambiente é o de apoiar o restante do esforço, garantindo que o próprio processo, guias (normas e orientações), e ferramentas (hardware, software, etc.) estejam disponíveis para a equipe quando necessário. De

acordo com a fase em que o projeto se encontra durante o ciclo de vida, as atividades da disciplina variam (AMBLER, 2006):

- Durante a fase de Iniciação, as seguintes atividades são realizadas (AMBLER, 2006):

o Configurar o ambiente de trabalho. Instalar estações de trabalho e software conforme necessário. Esta será uma tarefa contínua à medida que pessoas são adicionadas à equipe ao longo do tempo;

o Identificar a categoria do projeto. Muitas empresas desenvolvem várias versões base do seu processo de software, por exemplo, um para equipes pequenas, uma para a substituição de sistemas legados, um para instalações de sistemas comerciais, e assim por diante. Isso dá uma vantagem na adaptação do AUP para atender às necessidades exatas de cada equipe de projeto, pois várias customizações já foram realizadas.

- Durante a fase Elaboração, as seguintes atividades são realizadas (AMBLER, 2006):

o Evoluir o ambiente de trabalho. À medida que o projeto avança, o entendimento dos requisitos evolui, a estratégia de arquitetura evolui, e a visão do todo. O resultado final é a evolução do ambiente através da instalação de novas ferramentas, ou remoção de ferramentas que não são mais necessárias;

o Adequar os materiais do processo. O AUP deve ser adequado para atender as necessidades da equipe. Isto pode envolver tarefas como a edição dos materiais do processo e a escrita de um pequeno documento indicando o que não vai ser feito.

- Durante a fase de Construção, as seguintes atividades são realizadas (AMBLER, 2006):

o Apoiar a equipe. Membros da equipe do projeto vão precisar de ajuda utilizando e/ou configurando várias ferramentas para atender às suas necessidades. Eles também precisaram de ajuda para escolher os modelos (*templates*) de documentação adequada e seguir a orientação empresarial;

o Evoluir o ambiente de trabalho;

- Configurar ambientes de treinamento. À medida que o planejamento de implantação progride, pode ser descoberto que será preciso treinar os usuários finais, pessoal de apoio, e/ou pessoal de operações. Este esforço de treinamento pode exigir salas de formação e/ou versões de treinamento do sistema disponíveis, muitas vezes durante a fase de Transição. Pode ser preciso começar a definir esses ambientes no final da fase de Construção.

- Durante a fase de Transição, as seguintes atividades são realizadas (AMBLER, 2006):
  - Operações de instalação e / ou apoio de ambientes. Para apoiar a equipe e, às vezes, o pessoal de operações, muitas vezes, é preciso ter uma versão do sistema configurada para que eles usem para simular defeitos de forma segura;
  - Recuperar licenças de software. A medida que o projeto é concluído pode ser necessário desinstalar licenças de software de estações de trabalho das pessoas que já não as precisam mais, para que as licenças possam ser disponibilizadas para outros dentro da organização.

3 PROCEDIMENTOS METODOLÓGICOS

O objetivo do presente trabalho foi determinar a aderência do processo de desenvolvimento ágil AUP (*Agile Unified Process*) ao modelo de processo MPS.BR no nível de maturidade G (Parcialmente Gerenciado). Para isso, foram analisados os processos descritos nesse nível e seus resultados esperados em detalhes, a fim de determinar o grau de aderência do AUP a cada um deles. Foi elaborada uma escala ordenada com três categorias, com a determinação de classificação do atendimento do AUP. As três categorias são:

- Não Atendido: Há pouca evidência de que o resultado esperado esteja presente no AUP, ou seja, não existem atividades e/ou itens de revisões de marco definidos que concretizem o resultado esperado analisado;
- Parcialmente Atendido: Existem evidências de que o resultado esperado esteja presente no AUP, porém não é atendido completamente, ou seja, existem atividades e/ou itens de revisões de marco definidos, porém não o suficiente para concretizar o resultado esperado;
- Atendido: Há evidências significativas de que o resultado esperado esteja presente no AUP, através de atividades e/ou marcos de revisões de marco definidos.

Após a análise individual dos resultados esperados de cada processo, é possível concluir em qual das três categorias cada resultado se encontra, e posteriormente, qual a porcentagem de aderência do AUP em cada processo.

A capacidade do processo, ou seja, os Atributos de Processo (AP) exigidos pelo nível G de maturidade não foram avaliados pelo presente trabalho, por se tratarem de atributos que avaliam o grau de refinamento e institucionalização com que os processos são executados na unidade organizacional avaliada. Como o presente trabalho não está associado à aplicação do AUP em nenhuma unidade organizacional, a análise da capacidade do processo não foi realizada.

4 RESULTADOS E DISCUSSÃO

A seguir são apresentados os resultados e interpretações da análise de aderência dos processos Gerência de Projetos (GPR) e Gerência de Requisitos (GRE) que compõe o nível G do Modelo MPS.BR.

4.1 ANÁLISE DA ADERÊNCIA DO PROCESSO GERÊNCIA DE PROJETOS

A seguir, é apresentada a análise de conformidade do AUP com os resultados esperados do processo GPR do Modelo MPS.BR. Somente os resultados esperados para o nível G foram avaliados.

4.1.1 GPR 1: O escopo do trabalho para o projeto é definido.

No AUP, durante a Fase de Iniciação, uma de suas principais atividades é a definição do escopo do projeto. Essa atividade é abordada especificamente pela Disciplina Modelo através da modelagem inicial dos requisitos do sistema juntamente com a participação dos *stakeholders* do projeto, considerando a construção de artefatos como: casos de uso em alto nível, Diagramas de Fluxos de Dados (DFDs) para identificação de fluxos de trabalho e pilha hierarquizada de requisitos. Além disso, o marco LCO, que marca o final da Fase de Iniciação, exige a concordância do escopo do projeto pelos *stakeholders*. Assim, essa prática é considerada **Atendida**.

4.1.2 GPR 2: As tarefas e os produtos de trabalho do projeto são dimensionados utilizando métodos apropriados.

De acordo com o AUP, na Fase de Iniciação, na Disciplina Modelo são definidos, inicialmente em alto nível, os requisitos do sistema, as principais regras e entidades de negócio e requisitos técnicos, através da participação ativa dos *stakeholders*. Ainda na Fase de Iniciação, a Disciplina Implementação sugere a prototipação das principais telas para obter um maior entendimento para que posteriormente a estimativa de esforço necessário para a construção das mesmas seja mais precisa.

Na Disciplina Gestão de Projetos, durante a atividade de desenvolvimento do plano de iteração da fase seguinte, é sugerida a realização de uma reunião com toda a equipe do projeto para o planejamento do trabalho a ser realizado, pois segundo AMBLER (2009), a pessoa que vai fazer o trabalho muitas vezes é a pessoa mais adequada para planejá-la. Na Fase de Construção, na Disciplina Modelo, é sugerida a utilização da técnica de análise de *Model Storming* para entendimento e modelagem da solução dos requisitos.

Apesar de todas as atividades de entendimento, modelagem e planejamento dos requisitos, não fica claro no AUP qual método é utilizado para estimar o tamanho dos requisitos, ou seja, a dimensão das funcionalidades. Assim, essa prática é considerada **Parcialmente Atendida**.

4.1.3 GPR 3: O modelo e as fases do ciclo de vida do projeto são definidos.

Na Fase de Iniciação do AUP, a Disciplina Ambiente define uma atividade para identificação da categoria do projeto, ou seja, de acordo com o tipo de projeto que deve ser executado, o processo AUP deve ser adaptado para atender às necessidades exatas do projeto. Além disso, o marco LCO, que marca o final da Fase de Iniciação, exige a aceitação da customização do processo por todas as partes envolvidas no projeto. Assim, essa prática é considerada **Atendida**.

4.1.4 GPR 4: 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 (até o nível F).

Na Disciplina Gestão de Projetos, na Fase de Iniciação, existe a atividade para desenvolver o cronograma do projeto em alto nível, mostrando o esforço do projeto organizado em iterações e a data final prevista para o projeto. Além desta, existe a atividade para determinar a viabilidade do projeto, em termos financeiros, técnicos, operacionais e políticos. Todavia, não fica claro no AUP se é realizada uma consulta a dados históricos de esforço e custo de tarefas semelhantes presentes em outros projetos, ou referências técnicas, para auxiliar as estimativas no projeto corrente. Neste caso, a prática é considerada **Não Atendida**.

4.1.5 GPR 5: O orçamento e o cronograma do projeto, incluindo a definição de marcos e pontos de controle, são estabelecidos e mantidos.

No AUP, na Fase de Iniciação, a Disciplina Gestão de Projetos define as atividades para definir o cronograma em alto nível, mostrando o esforço do projeto organizado em iterações, indicando as revisões de marcos e tarefas de dependências críticas, além do desenvolvimento de um plano de iteração detalhado, contendo as atividades apropriadas para a iteração. Em cada fase, a Disciplina inclui atividades de revisão e atualização do plano de iteração, e em todos os marcos do AUP, são exigidas a comprovação da viabilidade do projeto e a adequação do plano de iteração para a próxima fase a ser executada. Assim, essa prática é considerada **Atendida**.

4.1.6 GPR 6: Os riscos do projeto são identificados e o seu impacto, probabilidade de ocorrência e prioridade de tratamento são determinados e documentados.

Nas Fases de Iniciação, Elaboração e Construção existem as atividades para gerenciar os riscos do projeto, sejam eles de negócio, técnicos e organizacionais. Os riscos devem ser identificados e estratégias para mitigá-los desenvolvidas. Segundo AMBLER (2009), os riscos conduzem a gestão de projetos, de forma que os riscos de maior prioridade sejam tratados em iterações mais próximas do que os riscos de baixa prioridade. Além disso, os marcos que finalizam as fases anteriormente citadas exigem a aceitação dos riscos, ou seja, a identificação, avaliação e desenvolvimento de estratégias de mitigação. Assim, essa prática é considerada **Atendida**.

4.1.7 GPR 7: Os recursos humanos para o projeto são planejados considerando o perfil e o conhecimento necessários para executá-lo.

No AUP, na Fase de Iniciação, a Disciplina de Gestão de Projetos define a atividade para iniciar a construção da equipe, primeiramente selecionando indivíduos com habilidades de modelagem para trabalhar junto aos *stakeholders* para o levantamento inicial dos requisitos do sistema e posteriormente repassar essas informações para a equipe técnica iniciar a identificação da arquitetura potencial do sistema. Já na Fase de Elaboração, essa mesma disciplina define novamente essa atividade, porém com foco na identificação de indivíduos com habilidades técnicas, para a implementação da arquitetura e das funcionalidades do sistema, identificando, se necessário, a necessidade de treinamento e capacitação da equipe de acordo com as habilidades requeridas para o projeto. Assim, essa prática é considerada **Atendida**.

4.1.8 GPR 8: Os recursos e o ambiente de trabalho necessários para executar o projeto são planejados (até o nível F).

Na Disciplina Gestão de Projetos do AUP, na Fase de Elaboração, é definida a atividade para a obtenção de recursos, onde é obtido o financiamento para a equipe, instalações físicas e softwares necessários para a realização de seus trabalhos. Além disso, na Disciplina Ambiente, são definidas atividades desde a Fase de Iniciação para montagem do ambiente de trabalho da equipe e sua evolução e adequação durante todas as fases do desenvolvimento, acompanhando as necessidades correntes do projeto. Assim, essa prática é considerada **Atendida**.

4.1.9 GPR 9: 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.

De acordo com o AUP, a Disciplina de Gestão de Configuração define que todos os produtos de trabalho do projeto devem ser colocados em um repositório para a Gestão de Configuração (CM), onde é definida a organização dos artefatos em uma estrutura de diretórios/pastas apropriada, o versionamento, bem como o acesso a esses artefatos. Porém, não fica claro no AUP em que momento são definidos os produtos de trabalho relevantes para o projeto, bem como a forma de coleta dos dados. Assim, essa prática é considerada **Parcialmente Atendida**.

4.1.10 GPR 10: Um plano geral para a execução do projeto é estabelecido com a integração de planos específicos.

No AUP, os marcos LCO, LCA e IOC que marcam o final das fases de Iniciação, Elaboração e Construção respectivamente, exigem que o Plano de Projeto esteja adequado e detalhado com informações sobre as iterações que deverão ocorrer na fase subsequente, além de informações em alto nível do projeto como um todo. A tarefa de atualização do Plano de Projeto em cada fase é atribuída a Disciplina Gestão de Projetos. O Plano de Projeto agrupa informações a respeito dos

planos de iterações, cronograma do projeto, lista de riscos, estimativas e orçamento. Assim, essa prática é considerada **Atendida**.

4.1.11 GPR 11: A viabilidade de atingir as metas do projeto é explicitamente avaliada considerando restrições e recursos disponíveis. Se necessário, ajustes são realizados.

Em todos os marcos do AUP são exigidas análise de viabilidade do projeto, considerando fatores técnicos, de negócio e operacionais (Fase de Iniciação e Elaboração) e aceitação dos custos e cronograma do projeto pelas partes envolvidas (Fase de Construção e Transição). Se as exigências dos marcos não forem satisfeitas, o projeto deve ser redirecionado ou cancelado, ficando a cargo da Disciplina de Gestão de Projetos essa análise. Assim, essa prática é considerada **Atendida**.

4.1.12 GPR 12: O Plano do Projeto é revisado com todos os interessados e o compromisso com ele é obtido e mantido.

No AUP, os marcos LCO, LCA e IOC que marcam o final das fases de Iniciação, Elaboração e Construção respectivamente, exigem que o Plano de Projeto esteja adequado para as fases subsequentes. Além disso, existe a análise de viabilidade/aceitação do projeto pelas partes interessadas em todos os marcos do AUP. Assim, essa prática é considerada **Atendida**.

4.1.13 GPR 13: O escopo, as tarefas, as estimativas, o orçamento e o cronograma do projeto são monitorados em relação ao planejado.

No AUP, o Plano de Projeto, que contém informações de estimativas, plano de iterações, cronograma e orçamento, são revistos e confirmados a cada revisão de marco. A Disciplina de Gestão de Projetos também define como seu objetivo o monitoramento do progresso do projeto. Todavia, não fica claro no AUP quais atividades devem ser desempenhadas para monitorar o progresso do projeto em relação ao que foi planejado, nem em que momento elas devem ser realizadas. Diante disto, essa prática é considerada **Parcialmente Atendida**.

4.1.14 GPR 14: Os recursos materiais e humanos bem como os dados relevantes do projeto são monitorados em relação ao planejado.

No AUP, com relação aos recursos materiais, a Disciplina Ambiente define a atividade para construir e evoluir o ambiente de trabalho ao longo do projeto, analisando as necessidades de cada fase que o projeto esteja. Sobre os dados do projeto, a Disciplina Gestão de Configuração define que todos os produtos de trabalho sejam mantidos sob o controle do repositório de CM. A respeito do monitoramento de recursos humanos, não fica claro no AUP quais atividades devem ser realizadas. Em geral, as tarefas de monitoramento desempenhadas além das revisões de marcos são pouco claras. Diante disto, essa prática é considerada **Parcialmente Atendida**.

4.1.15 GPR 15: Os riscos são monitorados em relação ao planejado.

A Disciplina Gestão de Projetos do AUP define atividades para a gestão dos riscos do projeto durante todo o ciclo de vida, envolvendo a investigação e desenvolvimento de estratégias para mitigá-los. Além disso, os marcos LCO, LCA e IOC que marcam o final das fases de Iniciação, Elaboração e Construção respectivamente, exigem que os riscos identificados sejam avaliados para garantir

que tenham sido corretamente entendidos e documentados, e as estratégias para lidar com eles sejam aceitáveis. Assim, essa prática é considerada **Atendida**.

4.1.16 GPR 16: O envolvimento das partes interessadas no projeto é planejado, monitorado e mantido.

No AUP, na Disciplina Gestão de Projetos, durante a Fase de Iniciação, é definida a atividade para construir relacionamentos com os *stakeholders* do projeto, com o objetivo de conquistar seu apoio e participação durante todo o projeto. Além disso, o principal objetivo de todos os marcos do AUP é conseguir a concordância dos *stakeholders* diante de todos as pontos abordados nas revisões de marcos. Assim, essa prática é considerada **Atendida**.

4.1.17 GPR 17: Revisões são realizadas em marcos do projeto e conforme estabelecido no planejamento.

A Disciplina Gestão de Projetos do AUP define atividades para desenvolver e atualizar o plano de iterações e Plano de Projeto em cada fase do ciclo de vida. Nesses planos são definidas as revisões dos marcos do projeto, que ocorrem sempre no final de cada fase. Nessas revisões, é verificado se os critérios do marco foram cumpridos com êxito. Assim, essa prática é considerada **Atendida**.

4.1.18 GPR 18: Registros de problemas identificados e o resultado da análise de questões pertinentes, incluindo dependências críticas, são estabelecidos e tratados com as partes interessadas.

No AUP, em cada revisão de marco, se alguma exigência não for aceita pelos *stakeholders*, deve ser avaliado pela Disciplina Gestão de Projetos o redirecionamento ou, até mesmo, o cancelamento do projeto. Porém, não fica claro quais atividades devem ser realizadas para tratar esses problemas ou se é desenvolvido um plano de ação para tratar questões como essa. Diante disto, essa prática é considerada **Não Atendida**.

4.1.19 GPR 19: 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.

No AUP, não fica claro em que momento, além das revisões de marcos, ocorre o monitoramento das atividades do projeto para a comparação entre o que foi planejado com o que foi realizado. Consequentemente, as atividades de replanejamento que devem ser realizadas quando um desvio for encontrado, também não são claras. Diante disto, essa prática é considerada **Não Atendida**.

4.1.20 Resultado da Análise de Aderência do Processo GPR

O resultado da análise de aderência do AUP com os dezenove resultados esperados do processo GPR para o nível G do Modelo MPS.BR é exibido na Figura 5.

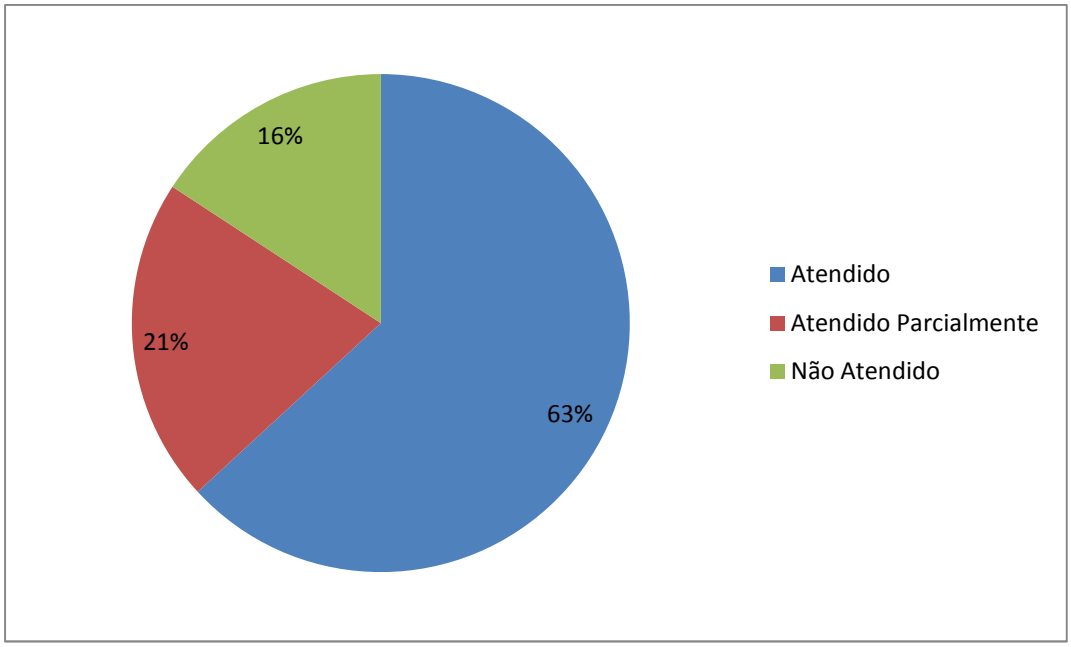

FIGURA 5 – Conformidade do AUP com os resultados esperados do processo GPR no nível G do Modelo MPS.BR

Na análise realizada, 63% dos resultados esperados encontram-se **Atendidos** pelo AUP, 21% encontram-se **Parcialmente Atendidos** e 16% como **Não Atendidos**. Percebe-se que o AUP não está em total conformidade com as exigências do processo GPR no nível G do Modelo MPS.BR, e o resultado deve-se principalmente a:

- Falta de informação a respeito da(s) técnica(s) de estimativa utilizada(s) para dimensionar o projeto;
- Ausência de práticas para alimentar e/ou consultar dados históricos de projetos anteriormente executados;
- Falta de informação a respeito das formas de coletas de dados dos projetos, e respectivo armazenamento e distribuição;
- Ausência de práticas para monitoramento dos dados do projeto, além das revisões de marcos que ocorrem ao final de cada fase do ciclo de vida;
- Falta de informação a respeito das práticas realizadas quando desvios e/ou problemas são identificados no projeto;

4.2 ANÁLISE DA ADERÊNCIA DO PROCESSO GERÊNCIA DE REQUISITOS

A seguir, é apresentada a análise de conformidade do AUP com os resultados esperados do processo GRE do Modelo MPS.BR.

4.2.1 GRE 1: O entendimento dos requisitos é obtido junto aos fornecedores de requisitos.

No AUP, durante a Fase de Iniciação, a Disciplina Modelo tem o objetivo de definir o escopo inicial do projeto, levantando os requisitos em alto nível junto aos *stakeholders*. É sugerida a criação de artefatos como casos de uso em alto nível e Diagramas de Fluxos de Dados (DFDs) para melhor explorar as informações repassadas pelos *stakeholders*. Assim, essa prática é considerada **Atendida**.

4.2.2 GRE 2: Os requisitos são avaliados com base em critérios objetivos e um comprometimento da equipe técnica com estes requisitos é obtido.

No AUP, na Disciplina Modelo, durante a Fase de Iniciação, os requisitos são levantados em alto nível e tratados como uma pilha hierarquizada, que evolui ao longo do tempo. A priorização dos requisitos é tarefa dos *stakeholders* do projeto. Nesta pilha, estão os casos de uso, regras de negócio e requisitos técnicos. O marco LCO, que marca o final da Fase de Iniciação, exige que haja uma compreensão compartilhada dos requisitos inicialmente levantados por parte dos *stakeholders* e da equipe do projeto. Porém, não fica claro no AUP quais critérios são utilizados para avaliar os requisitos do projeto. Diante disto, essa prática é considerada **Parcialmente Atendida**.

4.2.3 GRE 3: A rastreabilidade bidirecional entre os requisitos e os produtos de trabalho é estabelecida e mantida.

No AUP, não estão claras as atividades necessárias para determinar a dependência entre os produtos de trabalho do projeto, nem a definição do mecanismo para rastrear essas dependências de forma bidirecional. Assim, essa prática é considerada **Não Atendida**.

4.2.4 GRE 4: Revisões em planos e produtos de trabalho do projeto são realizadas visando identificar e corrigir inconsistências em relação aos requisitos.

Durante a Fase de Transição do AUP, na Disciplina Modelo, é indicada a atividade para finalização da documentação do sistema e realização de ajustes e correções de inconsistências entre os artefatos, pois, de acordo com AMBLER (2009), esse é o melhor momento para realizar essas atividades, pois o escopo do sistema está realmente estabilizado. Assim, essa prática é considerada **Atendida**.

4.2.5 GRE 5: Mudanças nos requisitos são gerenciadas ao longo do projeto.

No AUP, na Disciplina Modelo, durante a Fase de Iniciação, os requisitos são levantados em alto nível e tratados como uma pilha hierarquizada, que evolui ao longo do tempo. A priorização dos requisitos é tarefa dos *stakeholders* do projeto, que, além disso, podem, quando necessário, incluir novos requisitos, priorizá-los novamente e/ou alterá-los. Porém, não está claro no AUP se existem atividades para avaliar o impacto da mudança em outros requisitos, no cronograma, no custo e nos riscos do projeto. Diante disto, essa prática é considerada **Parcialmente Atendida**.

4.2.6 Resultado da Análise de Aderência do Processo GRE

O resultado da análise de aderência do AUP com os cinco resultados esperados do processo GRE do Modelo MPS.BR é exibido na Figura 6.

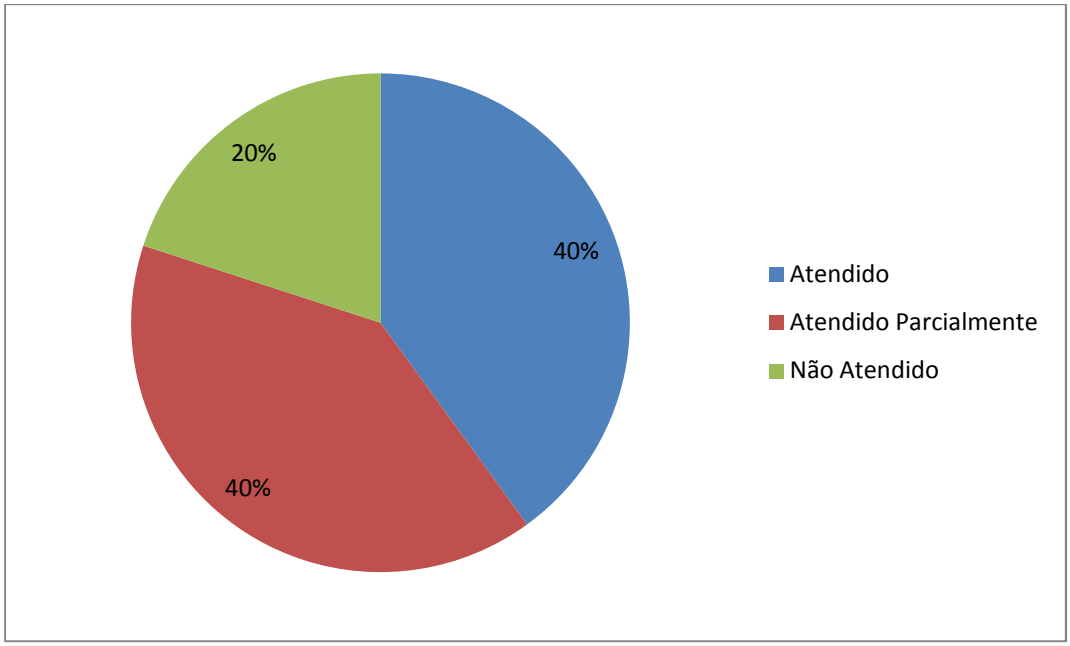

FIGURA 6 – Conformidade do AUP com os resultados esperados do processo GRE do Modelo MPS.BR

Na análise realizada, 40% dos resultados esperados encontram-se **Atendidos** pelo AUP, 40% encontram-se **Parcialmente Atendidos** e 20% como **Não Atendidos**. Percebe-se que o AUP não está em total conformidade com as exigências do processo GRE do Modelo MPS.BR, e o resultado deve-se principalmente a:

- Falta de informação a respeito dos critérios utilizados para avaliação objetiva dos requisitos;
- Ausência de práticas para estabelecer a dependência entre os produtos de trabalho do projeto;
- Falta de informação a respeito de atividades realizadas para avaliar o impacto das mudanças nos requisitos ocorridas durante o andamento do projeto.

4.3 CONCLUSÃO SOBRE A ANALISE DA ADERÊNCIA DOS PROCESSOS

A análise da aderência das atividades e/ou itens de revisões de marco definidos pelo AUP com os processos definidos pelo MPS.BR nível G, mostrou que cerca de 20% dos resultados esperados não são atendidos. Em contrapartida, mais do que 40% desses resultados são atendidos, e mais do que 20% são parcialmente atendidos. Esse resultado ocorre, principalmente, pela falta de documentação e formalização de atividades que mapeiem os resultados esperados pelo MPS.BR.

5 CONSIDERAÇÕES FINAIS

Com a crescente adoção de métodos ágeis de desenvolvimento de software entre as empresas, conhecidos por serem leves e pregarem a satisfação do cliente como principal prioridade no desenvolvimento de software, e, em contrapartida, a procura das empresas pela melhoria de seus processos através da adoção de modelos de referências de processo, como o Modelo MPS.BR, o presente trabalho analisou a aderência do método ágil AUP com os processos definidos pelo Modelo MPS.BR no nível de maturidade G.

O AUP, assim como o desenvolvimento ágil em geral, é guiado por valores e princípios publicados no Manifesto Ágil, podendo citar “Software em funcionamento mais que documentação abrangente” e “Indivíduos e interações mais que processos e ferramentas” como dois deles. Os valores e princípios estão refletidos nas definições de processo e atividades do AUP, que, em função disso, não se mostraram totalmente aderentes aos processos definidos pelo Modelo MPS.BR no nível G de maturidade. O Modelo exige evidências para comprovar a execução de suas definições de processo, e o AUP, em função dos princípios, não se mostrou totalmente aderente pela falta de atividades e documentação gerada que formalizasse a execução das atividades exigidas pelo MPS.BR.

Apesar disso, o AUP apresentou taxa inferior a 20% de itens não aderentes aos processos analisados, mostrando que com poucas alterações, poderá se tornar plenamente aderente a esses processos.

5.1 TRABALHOS FUTUROS

Novos trabalhos poderão ser realizados abordando a aderência do AUP com os demais níveis de maturidade definidos pelo Modelo MPS.BR, e também, quais modificações devem ser realizadas no AUP para que se torne totalmente aderente a esses níveis.

REFERÊNCIAS

AMBLER, S. W. **The Agile Unified Process (AUP)**, 2009. Disponível em <http://www.ambysoft.com/unifiedprocess/agileUP.html>. Acesso em 21 de Agosto de 2011.

_____. **Agile Model Driven Development (AMDD): The Key to Scaling Agile Software Development**. 2009b. Disponível em: <http://www.agilemodeling.com/essays/amdd.htm>. Acesso em 10 de Julho de 2011.

_____. **The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments**. Dezembro de 2009c. Disponível em ftp://public.dhe.ibm.com/common/ssi/ecm/en/raw14204usen/RAW14204USEN.PDF. Acesso em 10 de Julho de 2011.

_____. **The Agile Unified Process v1.1** 2006. Disponível em: <www.ambysoft.com/downloads/agileUP.zip>. Acesso em 09 de Julho de 2011.

_____. **Modelagem Ágil: Práticas eficazes para a Programação eXtrema e o Processo Unificado**. Porto Alegre: Bookman, 2004.

BECK, K. et al. **Manifesto for Agile Software Development**. 2001. Disponível em: <http://www.agilemanifesto.org/>. Acesso em 28 de Agosto de 2011.

BECK, K. **Test-Driven Development: By Example***.* Addison-Wesley Professional, 2003.

FOWLER, M; BECK, K. **Refactoring: Improving the Design of Existing Code**. Addison-Wesley Professional, 1999.

FUGGETTA, A. **Software process: a roadmap**. In: ICSE International Conference on Software Engineering, 22, 2000, Limerick – Ireland. ICSE '00 Proceedings of the Conference on The Future of Software Engineering, New York – USA, ACM, 2000. p. 25-34. Disponível em: <http://dl.acm.org/citation.cfm?id=336512.336521>. Acesso em Setembro de 2011.

GOLDENSON, D.; GIBSON, D. **Demonstrating the Impact and Benefits of CMMI®: An Update and Preliminary Results**. Outubro de 2003. Disponível em <http://repository.cmu.edu/cgi/viewcontent.cgi?article=1540&context=sei>. Acesso em Fevereiro de 2012.

IEEE COMPUTER SOCIETY. **Guide to the Software Engineering Body of Knowledge 2004 Version SWEBOK ® A project of the IEEE Computer Society Professional Practices Committee**. Society, 2004. Disponível em: <http://www.computer.org/portal/web/swebok/html/contents>. Acesso em Setembro de 2011.

KALINOWSKI, M. et al. **MPS.BR: Promovendo a Adoção de Boas Práticas de Engenharia de Software pela Indústria Brasileira**. In: CIbSE - XIII Congreso Iberoamericano en "Software Engineering", Universidad del Azuay, Cuenca - Ecuador, 12-16 Abril 2010. Disponível em <http://www.softex.br/mpsbr/_artigos/artigo.asp?id=3012>. Acesso em Fevereiro de 2012.

KRUCHTEN, P. **Introdução ao RUP - Rational Unified Process**. 2 ed. Rio de Janeiro: Ciência Moderna Ltda, 2003.

NOGUEIRA, M. **Qualidade no Setor de Software Brasileiro: Uma Avaliação das Práticas das Organizações**. Abril de 2006. Tese (Doutorado em Engenharia de Sistemas e Computação) – Programa de Pós-Graduação de Engenharia da Universidade Federal do Rio de Janeiro, 2006. Disponível em <http://teses.ufrj.br/COPPE_D/MauroOddoNogueira.pdf>. Acesso em Fevereiro de 2012.

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

SANTOS, G.; WEBER, K. **Lições Aprendidas na Gestão do Programa MPS.BR**. 2008. Disponível em <http://www.softex.br/mpsbr/_livros/licoes/mpsbr_pt.pdf>. Acesso em 18 de Agosto de 2011.

SOFTEX. **MPS.BR - Melhoria de Processo do Software Brasileiro - Guia Geral**. Agosto de 2011. Disponível em: <http://www.softex.br/mpsbr/_guias/default.asp>. Acesso em Setembro de 2011.

_____. **MPS.BR - Melhoria de Processo do Software Brasileiro - Guia de Implementação – Parte 1: Fundamentação para Implementação do Nível G do MR-MPS**. Agosto de 2011b. Disponível em: <http://www.softex.br/mpsbr/_guias/default.asp>. Acesso em Setembro de 2011.

_____. **MPS.BR - Melhoria de Processo do Software Brasileiro - Guia de Implementação – Parte 2: Fundamentação para Implementação do Nível F do**

MR-MPS. Agosto de 2011c. Disponível em: <http://www.softex.br/mpsbr/_guias/default.asp>. Acesso em Setembro de 2011.

_____. **MPS.BR - Melhoria de Processo do Software Brasileiro - Guia de Implementação – Parte 3: Fundamentação para Implementação do Nível E do MR-MPS**. Agosto de 2011d. Disponível em: <http://www.softex.br/mpsbr/_guias/default.asp>. Acesso em Setembro de 2011.

_____. **MPS.BR - Melhoria de Processo do Software Brasileiro - Guia de Implementação – Parte 6: Fundamentação para Implementação do Nível B do MR-MPS**. Agosto de 2011e. Disponível em: <http://www.softex.br/mpsbr/_guias/default.asp>. Acesso em Setembro de 2011.

_____. **MPS.BR - Melhoria de Processo do Software Brasileiro - Guia de Implementação – Parte 7: Fundamentação para Implementação do Nível A do MR-MPS**. Agosto de 2011f. Disponível em: <http://www.softex.br/mpsbr/_guias/default.asp>. Acesso em Setembro de 2011.

SOFTWARE ENGINEERING INSTITUTE. **CMMI® para Desenvolvimento – Versão 1.2**. Agosto de 2006. Disponível em: <http://www.sei.cmu.edu/library/abstracts/whitepapers/CMMI-Dev-V1-2-Portuguese-translation.cfm>. Acesso em 02 de Abril de 2011.

_____**. CMMI® or Agile: Why Not Embrace Both!**. Novembro de 2008. Disponível em: <http://www.sei.cmu.edu/library/abstracts/reports/08tn003.cfm?DCSext.abstractsource=SearchResults>. Acesso em Agosto de 2011.

TRAVASSOS, G. H.; KALINOWSKI, M. **iMPS: Resultados de desempenho de organizações que adotaram o Modelo MPS**. 2008. Disponível em: <http://www.softex.br/mpsbr/_livros/imps/imps.pdf>. Acesso em Setembro de 2011.

VERSIONONE. **6th Annual State of Agile Development Survey**, 2011. Disponível em <http://www.versionone.com/pdf/2011_State_of_Agile_Development_Survey_Results.pdf>. Acesso em Fevereiro de 2012.