UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ADEQUAÇÃO DO PROCESSO UNIFICADO NO DESENVOLVIMENTO DE SISTEMAS GRP (*GOVERNMENT RESOURCE PLANNING*)

ORIENTADOR: EDGARD LAMOUNIER JR., Ph.D.
CO-ORIENTADOR: ALEXANDRE CARDOSO, Dr.

ORIENTADO: MAURO BORGES FRANÇA

Uberlândia
2012

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ADEQUAÇÃO DO PROCESSO UNIFICADO NO DESENVOLVIMENTO DE SISTEMAS GRP (*GOVERNMENT RESOURCE PLANNING*)

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciências, conferido pela Universidade Federal de Uberlândia – UFU, por meio da Faculdade de Engenharia Elétrica.

Banca Examinadora:

Edgard Lamounier Junior – Orientador – UFU

Alexandre Cardoso – Co-orientador – UFU

Adriano Alves Pereira – UFU

Marcos Wagner de Souza Ribeiro – UFG

Uberlândia
2012

ADEQUAÇÃO DO PROCESSO UNIFICADO NO DESENVOLVIMENTO DE SISTEMAS GRP (*GOVERNMENT RESOURCE PLANNING*)

MAURO BORGES FRANÇA

Dissertação apresentada perante a banca abaixo como requisito parcial à obtenção do grau de Mestre em Ciências, conferido pela Universidade Federal de Uberlândia – UFU, por meio da Faculdade de Engenharia Elétrica.

Prof. Edgard Lamounier Jr. PhD
Orientador

Prof. Alexandre Cardoso, Dr.
Coordenador do Curso de Pós-Graduação

Uberlândia
2012

DEDICATÓRIA

À minha esposa Ana Carolina

Aos meus filhos Guilherme e Vitor

Aos meus pais Toninho e Neusa

Aos meus irmãos Mauricio e Marina

AGRADECIMENTOS

Agradeço primeiro a DEUS pelos caminhos que tem proporcionado em minha vida.

Agradeço a minha esposa Ana Carolina, incondicional apoio para realização deste projeto, sempre esteve ao meu lado me apoiando e compreendendo minhas ausências em momentos de dedicação a este trabalho. Muito obrigado, amo muito você.

Aos meus filhos Guilherme e Vitor, meus agradecimentos mais profundos, pois aceitaram minha ausência em momentos importantes de suas vidas.

Ao meu pai pelo bom exemplo, companheirismo, afeto, amor e carinho. Mesmo neste momento difícil de sua vida não deixou de expressar por meio de gestos seu apoio incondicional.

A minha mãe iluminada, agradeço o imenso amor e carinho, o que me proporcionaram muitas alegrias em minha vida, considero uma guerreira por tudo, mas principalmente pelos momentos atuais.

Aos meus irmãos que sempre me apoiaram e torceram por mim em toda minha trajetória.

Aos meus amigos Edgard e Alexandre que simplesmente canalizaram suas energias não só em orientações, mas também em motivações e momentos de amizade. Este projeto não teria êxito sem a participação de vocês e por isso muitíssimo obrigado por tudo.

Aos meus amigos da DTIC que me ajudaram, apoiaram e participaram deste projeto.

Aos meus amigos professores de computação do IFTM que também tiveram papel fundamental.

Ao Instituto Federal do Triângulo Mineiro pelo acolhimento e reconhecimento profissional.

Enfim a todos, que direta ou indiretamente participaram neste projeto, meus sinceros agradecimentos.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Obter um processo de software adequado ao desenvolvimento confiável, de boa aderência e que não dispenda tempo demasiado na geração de artefatos é forte motivação para responder demandas de mercado, portanto requer pesquisa. Um dos grandes desafios neste processo é adequar as propostas de diferentes metodologias de desenvolvimento de software à realidade de trabalho e força tarefa de uma equipe de Tecnologia da Informação. Esta pesquisa propõe a adequar uma metodologia baseada em processos unificados, visando sua adaptação, em particular, ao desenvolvimento de software para repartições públicas. Um dos motivos desta proposta se baseou pelo fato da inexistência de um padrão de desenvolvimento em equipes desta natureza. O presente estudo trata-se da aplicação de uma metodologia de desenvolvimento de sistemas para construção de sistema integrado com características de um GRPs (*Governnement Resource Plannig*). Como modelo de aplicação, utilizaram-se três projetos com características próximas nos aspectos de complexidade e força trabalho. Estes projetos foram trabalhados por equipes distintas na fábrica de software do Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro – IFTM. Para alcançar as metas estabelecidas à pesquisa foi aplicada durante seis meses e apresentou resultados quantitativos quanto à definição de processos padrões e geração de artefatos. Portanto, a pesquisa conduz à aplicação dos resultados obtidos em Instituições com características semelhantes.

Palavras-Chave: Processo de Desenvolvimento de Software, Processo Unificado, Metodologias Ágeis, GRPs (*Governnement Resource Plannig*)**.**

ABSTRACT

Obtaining a suitable and trustful software development process, with good quality and that not requires great time in generating artifacts is a strong motivation to respond to market demands. Therefore, it requires strong research. One of the great challenges in this process is to appropriate different software development methodologies to the reality of an Information Technology group task force. This research proposes to adjust a unified process based methodology in order to adapt, particularly, to the software development within the public sector. One of the reasons for this proposal is based upon the fact that none development pattern has been established in such groups, yet. The present study is related to integrated systems that have Government Resource Planning (GRPs) characteristics. As an application model, three projects that have similar complexity aspects and task force characteristics have been used. These projects were undertaken by distinct times within the software factory of the Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro – IFTM. In order to reach established goals the research was conducted during six months and has presented quantitative results when taking the definition of standard patterns and artifacts generation into account. Therefore, this research conducts to apply the obtained results in Institutions that holds similar characteristics.

Keywords: Software Development Process Unified Process, Agile Methodology, GRPs (*Governnement Resource Plannig*)**.**

1. Introdução

1.1. Motivação

A Engenharia de Software nasceu da necessidade de desenvolver, por métodos de engenharia, a produção de programas de computadores, envolvendo processos e métricas de tempo, custo, envolvimento de pessoal, resultados e possíveis avanços (SOMMERVILLE, 2011).

Neste contexto, é notório dizer que desenvolver software de qualidade é um grande desafio para as fábricas de software, independente de seu porte. E, encontrar o processo adequado ao desenvolvimento é fator de estudos por vários pesquisadores no mundo. O trabalho de Machado (2000) relata que o surgimento de padrões internacionais para processos de software, como a Norma ISO/IEC 12207 e os modelos de maturidade (CMM, TRILLIUM, BOOTSTRAP e ISO/IEC 15504) influenciaram as organizações a direcionar seus esforços não só na definição de processos, mas também, no estabelecimento de mecanismos para melhorá-los, continuamente. No entanto, o desenvolvimento de software continua, em muitos casos, sendo uma atividade fortemente dependente das habilidades individuais dos desenvolvedores, haja vista que várias organizações ainda não possuem processos definidos e pouco conhecimento quanto à maturidade de processos.

Ressalta-se ainda que as empresas utilizam algum processo de desenvolvimento de software, seja em pequenos, médios ou grandes projetos, mesmo que sejam processos informais. A adoção de metodologias de desenvolvimento orientam os integrantes do projeto para as atividades, ações e tarefas necessárias para desenvolver um software de alta qualidade. Um processo, segundo o IEEE, é uma “sequência de passos executados com um determinado objetivo”; segundo o CMMI, é “um conjunto de ações inter-relacionadas realizadas para obter um conjunto especificado de produtos, resultados ou serviços” (PÁDUA FILHO, 2009).

Fuggetta (2000), Pressman (2006) e Osterweil (1987), afirmam que a qualidade do processo de desenvolvimento impacta diretamente na qualidade do produto a ser desenvolvido. Ainda, segundo Greer e Conradi (2009), a necessidade de um processo definido é reconhecida como uma estratégia de redução de riscos para a gerência de projetos. O trabalho apresentado por Bertollo e Falbo (2003), acrescenta que “a

principal causa dos problemas no desenvolvimento de software é a falta de um processo de desenvolvimento claramente definido e efetivo". Portanto, fica claro que para redução de problemas de desenvolvimento de software, redução de riscos e entrega de software de qualidade, a adoção de algum processo de desenvolvimento é extremamente importante em projetos desta natureza.

Ainda destacando a importância no uso de processos de software, é notório dizer que a adoção destes trazem vários benefícios para o projeto. Dentre eles destacam-se:

- A geração de artefatos concretos produzidos durante o desenvolvimento do software tem como objetivo ajudar a descrever as funções, arquitetura e design do software. Booch, Rumbaugh e Jacobson (2006), afirmam que a documentação de um projeto de software, é a parte central de todas as atividades que levam à implantação bem sucedida de seu produto;
- Definição de linguagem única entre a equipe de desenvolvimento e os usuários envolvidos no projeto;
- Ações para estabelecer que todos os membros da equipe absorvam os conhecimentos necessários de todo processo de desenvolvimento do software;
- Identificação dos papéis entre os colaboradores, com objetivo de definir as responsabilidades para cada um dos papéis;
- Integração entre as equipes, ou seja, os colaboradores compartilham tarefas entre diferentes times;
- Compartilhar processos comuns para diversos projetos;
- Definir critérios para certificação de todos os processos por meio de auditoria;
- Melhorar a qualidade do software. Conforme afirma Koscianski e Soares (2007) a qualidade de software ainda depende, principalmente, do correto emprego de boas metodologias pelos desenvolvedores;
- Facilitar a participação de novos membros da equipe, após treinamento;
- Permitir que qualquer membro da equipe afaste do projeto para férias ou outro motivo qualquer, sem parar o processo de desenvolvimento.

Os benefícios citados deixam claro que a adoção de um processo para o desenvolvimento de qualquer sistema é essencial para o sucesso do projeto. Vale

destacar que tais processos geram artefatos que caracterizam informações sobre o domínio do software em construção e estes são elementos fortemente defendidos neste trabalho.

A adoção de uma metodologia depende de vários outros aspectos importantes, que também se destacam:

- Tamanho da equipe;
- Participação de colaboradores com perfil adequado em conformidade com a equipe;
- Conhecimento dos colaboradores em alguma metodologia já existente;
- Experiências anteriores de outras fábricas de software;
- Integração entre os membros das equipes.

Dentre estes processos, destacam-se os métodos tradicionais e os métodos ágeis. Algumas empresas julgam que o software que produzem pode ser compreendido simplesmente ao ler seu código-fonte (métodos ágeis). Outras fábricas documentam seus produtos de forma intensiva (métodos tradicionais) (SHACH, 2009).

Entretanto, centros de desenvolvimento de software de repartições públicas apresentam problemas na adoção de métodos tradicionais ou ágeis. Isto porque possuem características distintas daquelas comumente encontradas nas demais indústrias de software. Dentre estas se destacam: número reduzido de profissionais, alta rotatividade, processos demorados de licitação, equipes heterogêneas, falta de padrões, falta de documentações ou documentações incompletas.

Neste sentido, a aplicação dos métodos tradicionais em repartições públicas apresenta como principal problema a necessidade de grandes equipes para atenderem todas as fases e papéis definidos por estes modelos. Já os processos ágeis que, normalmente, são aplicados em empresas com foco comercial (com o propósito de busca por maior competitividade, produtividade e lucro), têm se mostrado pouco viável para repartições públicas. De fato, uma vez que são orientadas a codificação, fica a cargo do desenvolvedor a realização de códigos auto-documentáveis, podendo ocorrer riscos devido à alta rotatividade e a não transferência do conhecimento tácito.

Alguns trabalhos de adequações ou combinações de metodologias foram realizados nos últimos anos. Em destaque o trabalho apresentado em Alves Paim *et al*. (2011), que aborda dois aspectos interessantes quanto a investigação desta área: (1)

“Muito pouco ainda se sabe sobre os reais benefícios dos métodos ágeis, pois até o presente momento, foram realizados poucos estudos empíricos com um padrão aceitável de confiabilidade e validade (DYBÅ e DINGSøYR, 2008)(DYBÅ e DINGSøYR, 2009”. (2) “Apesar dos esforços para levantar as reais implicações do uso de métodos ágeis e de sua combinação com abordagens tradicionais, pouca evidência empírica foi apresentada até o momento concernente a processos de desenvolvimento de software (RAMSIN e PAIGE, 2008)(ROMBACH e SEELISCH, 2007)”. Estas abordagens evidenciam que a área ainda é pouco explorada e há um número ainda pequeno de pesquisas no que concerne a processos de desenvolvimento de software com adoção de técnicas mistas.

Neste escopo, surge como desafio a escolha de uma metodologia mais adequada para a realidade das repartições públicas. É importante que esta metodologia seja projetada para permitir agilidade e acompanhamento de todo o processo do setor de desenvolvimento.

1.2. Objetivo

Este trabalho propõe investigar um conjunto de técnicas computacionais e gerenciais, adequando a integração de metodologias de processos unificados e métodos ágeis às características ambientais do setor de desenvolvimento de software em repartições públicas.

1.3. Objetivos Específicos

- Avaliar metodologias e processos de software adotados para empresas que possuem um perfil de pequena e média empresa, destacando suas limitações;
- Propor uma metodologia aderente às características inerentes a uma classe comum de repartições públicas, tendo como referência a fábrica de software do Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro - IFTM;
- Desenvolver prova de conceito com a equipe de TI do IFTM;
- Avaliar a metodologia proposta por meio de análises qualitativas;
- Apresentar os resultados da aplicação da metodologia e encaminhar propostas de melhoramentos;

O resultado da avaliação em produtividade da proposta será por meio de estudo a ser aplicado no Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro localizado na cidade de Uberaba (MG). Neste estudo, serão coletados dados por meio de questionários a serem aplicados aos coordenadores de sistemas e desenvolvedores, e serão analisados junto ao processo de produtividade das equipes. Estes resultados serão avaliados em relação a situações anteriores vivenciadas pelas mesmas equipes que participarem da pesquisa.

1.4. Estrutura da dissertação

Este trabalho está divido em sete capítulos. São eles:

- Capítulo 1 – apresenta a contextualização do problema, a motivação para pesquisa e os objetivos a serem alcançados com sua aplicação.
- Capítulo 2 – apresenta conceitos e definições necessárias para o entendimento da proposta, sendo estas definições relacionadas às áreas de conhecimento envolvidas nesta pesquisa. Neste capítulo, foram abordadas algumas metodologias de desenvolvimento, atualmente, aplicadas nas indústrias de software, técnicas de gerencia de projetos e conceitos de GRP.
- Capítulo 3 – apresenta a revisão da literatura com alguns trabalhos relacionados à pesquisa em questão.
- Capítulo 4 – apresenta a proposta da MDS-GRP (Metodologia de Desenvolvimento de Sistemas – *Governnament Resource Planning*), definindo todas as fases a serem seguidos, os artefatos gerados e os processos a serem atendidos.
- Capítulo 5 – apresenta o exemplo de aplicação, contextualizando o problema da realidade antes da pesquisa e demonstrando a estrutura do Projeto GRP-IFTM.
- Capítulo 6 – apresenta os passos utilizados para validar a metodologia proposta em um dos projetos do estudo de caso. Apresenta também uma análise dos resultados obtidos, a fim de avaliar a eficácia da proposta.
- Capítulo 7 – apresenta as conclusões desta pesquisa e discute sugestões de trabalhos futuros relacionados ao tema.

2. Fundamentos

2.1. Introdução

Neste capítulo são apresentados e discutidos os conceitos fundamentais relacionados às áreas de conhecimento mencionadas nesta dissertação para uma maior compreensão da pesquisa conduzida.

2.2. Visão Geral da Engenharia de Software

Na sociedade da informação e comunicação, a Engenharia de Software possui como principal finalidade viabilizar o desenvolvimento profissional de software, por meio de técnicas que possibilitam o direcionamento de especificações, projetos e evoluções de programas. A fim de visualizar, em linhas gerais, conceitos sobre software e sua engenharia, Sommerville (2011) apresenta perguntas e respostas destinadas à compreensão da dinâmica da Engenharia de Software, na Tabela 1.

Tabela 1 - Perguntas frequentes sobre software.

Pergunta	Resposta
O que é software?	Softwares são programas de computador e documentação associada. Produtos de software podem ser desenvolvidos para um cliente específico ou para o mercado em geral.
Quais são os atributos de um bom software?	Um bom software deve prover a funcionalidade e o desempenho requeridos pelo usuário; além disso, deve ser confiável e fácil de manter e usar.
O que é engenharia de software?	É uma disciplina de engenharia que se preocupa com todos os aspectos de produção de software.
Quais são as principais atividades da engenharia de software?	Especificação de software, desenvolvimento de software, validação de software e evolução de software.
Qual a diferença entre engenharia de software e ciência da computação?	Ciência da computação foca a teoria e os fundamentos; engenharia de software preocupa-se com o lado prático do desenvolvimento e entrega de softwares úteis.
Qual a diferença entre engenharia de software e engenharia de sistemas?	Engenharia de sistemas se preocupa com todos os aspectos do desenvolvimento de sistemas computacionais, incluindo engenharia de hardware, software e processo. Engenharia de software é uma parte específica desse processo mais genérico.
Quais são os principais desafios da engenharia de software?	Lidar com o aumento de diversidade, demandas pela diminuição do tempo para entrega e desenvolvimento de software confiável.
Quais são os custos da engenharia de software?	Aproximadamente 60% dos custos de software são de desenvolvimento; 40% são custos de testes. Para software customizado, os custos de evolução frequentemente

	superam os custos de desenvolvimento.
Quais são as melhores técnicas e métodos da engenharia de software?	Enquanto todos os projetos de software devem ser gerenciados e desenvolvidos profissionalmente, técnicas diferentes são adequadas para tipos de sistemas diferentes. Por exemplo, jogos devem ser sempre desenvolvidos usando uma série de protótipos, enquanto sistemas de controle críticos de segurança requerem uma especificação analisável e completa. Portanto, não se pode dizer que um método é melhor que outro.
Quais diferenças foram feitas pela internet na engenharia de software?	A Internet tornou serviços de software disponíveis e possibilitou o desenvolvimento de sistemas altamente distribuídos baseados em serviços. O desenvolvimento de sistemas baseados em Web gerou importantes avanços nas linguagens de programação e reuso de software.

Fonte: (Sommerville, 2011).

Em virtude dessas especificações, infere-se que os softwares em todas as suas formas e campos de aplicação, devem ser desenvolvidos por meio de procedimentos de Engenharia (PRESSMAN, 2011).

2.3. Processo de desenvolvimento de software

O contexto desta pesquisa volta-se para a adequação do processo unificado para elaboração de um software governamental. Para isso, torna-se necessário a compreensão do que seja um processo de software (ou metodologia de desenvolvimento de software). Nesse sentido, para indicar a relevância do processo de desenvolvimento de software no contexto da engenharia, Pressman (2011) relata que a Engenharia de Software é uma tecnologia estruturada, conforme Figura 1.

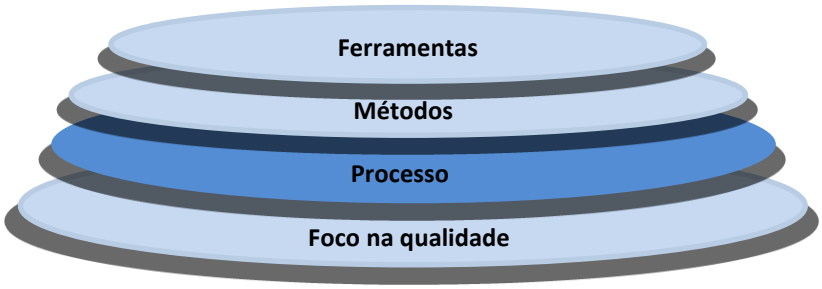

Figura 1 - Camadas da Engenharia de Software.

Ao analisar a Figura 1, a base para a Engenharia de Software são os processos, os quais interligam e mantêm as dinâmicas coesas de tecnologia e possibilitam seu desenvolvimento racional e tempestivo.

Nessa perspectiva, o processo estabelece os métodos que devem ser adotados para a conclusão efetiva da tecnologia de Engenharia de Software. Desse modo, o processo forma a base para o controle da gestão de projetos de software e determina o contexto no qual são aplicados métodos técnicos, gerados os produtos derivados (modelos, documentos, dados, relatórios, formulários e entre outros), incluindo a definição de marcos, em que a qualidade é assegurada e as mudanças são conduzidas adequadamente (PRESSMAN, 2011).

Segundo Pfleeger (2004), um processo é uma cadeia de etapas que abrangem atividades, restrições e recursos para conseguir as metas almejadas. Para Pádua Filho (2009), as definições se voltam para o IEEE no qual este define processo como uma sequência de passos realizados para um determinado objetivo. Este autor também emprega os conceitos apresentados no CMMI[1] em que processo é um conjunto de ações e atividades interligadas e realizadas para alcançar um grupo específico de produtos, resultados ou serviços. Por sua vez, Shach (2009), em uma visão mais simplificada, define processo como o modo pelo qual se produz um software. Pressman (2011) define processo como um grupo de procedimentos, ações e tarefas efetuadas na formalização de algum produto de trabalho. De acordo com Sommerville (2011), um processo de software é um grupo de atividades relacionadas que se destinam à fabricação de um produto de software.

Existe uma proximidade entre todos esses conceitos, porém ao agrupá-los infere-se que processo de software é o contíguo de atividades agrupadas em etapas que formam o desenvolvimento de um sistema computacional, a saber: especificação do projeto, determinação de pré-requisitos, análise, projeto, desenvolvimento, teste e implantação. Essas etapas envolvem profissionais que desempenham papéis pré-definidos, sendo que tais atividades originam produtos que são definidos como componentes do sistema.

---

[1]CMMI – *Capability Maturity Model Integration*: é um modelo de referência que contém práticas (Genéricas ou Específicas) necessárias à maturidade em disciplinas específicas de software. Desenvolvido pelo SEI (*Software Engineering Institute*) da Universidade Carnegie Mellon, o CMMI procura estabelecer um modelo único para o processo de melhoria corporativo, integrando diferentes modelos e disciplinas.

Pádua Filho (2009) constrói uma tabela (Tabela 2) com os termos essenciais envolvidos no conceito de processos.

Tabela 2 - Principais termos envolvidos na definição de processos

Termo	Definição
Insumo	(Praxis) Qualquer coisa que é consumida num processo.
Papel	(SPEM) Conjunto correlato de proficiências, competências e responsabilidades, desempenhado por uma ou mais pessoas.
Etapa	(Praxis) Tempo genérico para uma divisão temporal de um processo.
Processo de desenvolvimento	(UML)Conjunto de passos e diretrizes para o desenvolvimento de um produto.
Produto	1. (PMBOK) Um objeto produzido, quantificável, e que pode ser um item final ou um item componente. 2. (CMMI) Resultado que se pretende entregar a um cliente ou usuário.
Produto de trabalho	1. (SPEM) Qualquer coisa consumida, produzida ou modificada por tarefas. 2. (CMMI) Coisa útil que resulta de um processo.
Projeto	1. (CMMI) Conjunto gerido de recursos inter-relacionados, que entrega um ou mais produtos a um cliente ou usuário, com início definido e que, tipicamente, opera conforme um plano. Cf. produto. 2. (PMBOK) Um empreendimento temporário realizado para criar um produto, serviço ou resultado distinto.
Resultado	(PMBOK) Uma saída dos processos e atividades de um projeto.

Fonte: (Pádua Filho, 2009).

2.4. Modelos de desenvolvimento de software

O primeiro passo na definição do processo de software é propor um modelo de ciclo de vida. Essa proposta irá detalhar a maneira mais apropriada de se satisfazer as necessidades do usuário, chegando a quando e como este obterá a primeira versão operacional do sistema. Importante destacar que não há modelo ideal, uma vez que o software depende de fatores como a complexidade do propósito a ser atendido, o tempo para sua confecção, o custo estimado, a equipe envolvida, o ambiente operacional. Esses aspectos influenciarão diretamente na opção pelo ciclo de vida de software a ser elaborado (PFLEEGER, 2004).

Segundo Pfleeger (2004), os ciclos de vida podem se comportar de forma sequencial (fases seguem determinada ordem) e/ou incremental (divisão de escopo) e/ou iterativa (retroalimentação de fases) e/ou evolutiva (software é aprimorado).

Uma preocupação relevante, a qual será levantada por toda organização que procura adotar uma nova metodologia, é a quantidade de empresas que já empregam tal método, com sucesso, há determinado tempo. Infelizmente, é de comum acordo entre muitos grupos de TI de instituições públicas que não existe praticamente nada determinado estatisticamente. É necessário valer-se de alguns poucos estudos, conduzidos informalmente, para verificar as tendências do mercado.

2.5. Metodologias Ágeis

Na sociedade contemporânea, as metodologias ágeis têm concentrado o interesse das indústrias de software no mercado mundial, expressando evidências em diferentes casos na melhoria da produtividade. Ressalta-se que a iniciativa para criação de métodos ágeis iniciou-se em 2001, motivados pela visão de que os profissionais que lidam com o desenvolvimento de software nas mais diversas organizações se encontravam presos a processos cada vez mais burocráticos. Para tanto, um grupo de profissionais e pesquisadores de TI, coligaram-se para traçar valores e princípios de desenvolvimento de software que viabilizaram, às equipes de desenvolvimento, produzi-lo rapidamente e responder às mudanças. Esses pioneiros da metodologia ágil intitularam-se de Aliança Ágil, trabalhando por dois dias para criar tais valores. O resultado foi o Manifesto da Aliança Ágil (MARTIN, 2002), apesar de que estas metodologias que compõem o movimento ágil estão no mercado há alguns anos, sob a designação de metodologias leves. Desde então, o Manifesto Ágil é considerado oficialmente como o início do movimento ágil.

Os conceitos chave do Manifesto Ágil 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 (BECK *et al*., 2001).

Além dos conceitos chaves, esse Manifesto também enuncia os doze princípios de um processo ágil, conforme apresentado na Tabela 3 (BECK *et al*., 2001).

Tabela 3 - Doze princípios de um processo ágil

ID	Princípio
P1	A prioridade é satisfazer ao cliente através de entregas contínuas e frequentes;
P2	Receber bem as mudanças de requisitos, mesmo em uma fase avançada do projeto;
P3	Entregas com frequência, sempre na menor escala de tempo.
P4	As equipes de negócio e de desenvolvimento devem trabalhar juntas diariamente;
P5	Manter uma equipe motivada fornecendo ambiente, apoio e confiança necessários;
P6	A maneira mais eficiente da informação circular através de uma conversa face-aface;
P7	Ter o sistema funcionando é a melhor medida de progresso;
P8	Processos ágeis promovem o desenvolvimento sustentável;
P9	Atenção contínua a excelência técnica e a um bom projeto aumentam a agilidade;
P10	Simplicidade é essencial;
P11	As melhores arquiteturas, requisitos e projetos provêm de equipes organizadas;
P12	Em intervalos regulares, a equipe deve refletir sobre como se tornar mais eficaz.

Fonte: (Beck *et al*., 2001).

Com base nesses princípios, segundo Abrahamsson (2003), uma metodologia é considerada ágil quando efetua o desenvolvimento de software de modo incremental (disponibilização de pequenas versões, em iterações de curta duração), colaborativa (investidor e profissionais do software trabalhando juntos, em permanente interação), direta (método simplificado de aprender e modificar) e adaptativa (hábil em responder às mudanças até o último instante).

Nesse conceito, destacam-se as seguintes metodologias ágeis: *Extreme Programming* (XP), *Scrum*, *Crystal*, *Feature Driven Development* (FDD), *Dynamic Systems Development Method* (DSDM), *Open Source Software Development*. Embora esses métodos utilizem princípios semelhantes, são diferentes em suas práticas e formas de condução do processo de desenvolvimento. Dentre os métodos mais utilizados pela indústria de software destaca-se o XP e o SCRUM (PADUA FILHO, 2009).

2.5.1. eXtreme Programming – XP

O XP é um modelo ágil de desenvolvimento de software inventado, na década de 1990, mais especificamente em 1996, por Kent Bech no departamento de

computação da montadora de carros Daimler Chyrsler. Esse modelo detém várias diferenças em relação a outros modelos, já que pode ser empregado em projetos de alto risco e com pré-requisitos dinâmicos, conduzidos por equipes de tamanhos médio e pequeno (TELES, 2004).

Como toda metodologia ágil, o XP busca responder com velocidade as mudanças nas especificações do projeto, com base em princípios, valores e práticas bastante delineados. Esse método realça o desenvolvimento rápido, assegurando a satisfação do cliente e cumprindo as estimativas do projeto.

A Figura 2 apresenta todas as práticas do XP de forma resumida.

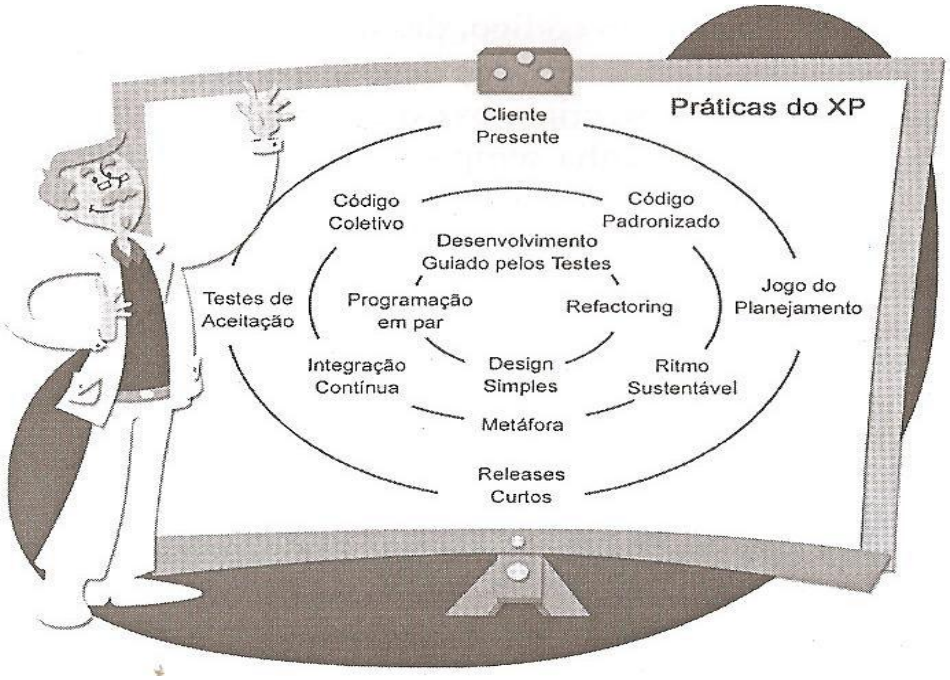

Figura 2 - Práticas do XP (Fonte: TELES, 2004).

O XP baseia-se em cinco valores que compõem o desenvolvimento: comunicação, coragem, feedback, respeito e simplicidade. Segundo Beck *et al*. (2001), o método oferece ainda 12 práticas, a saber:

- **Planejamento:** momento em que se decide o que precisa ser feito e o que pode ser adiado no projeto, lembrando que a XP se fundamenta em requisitos atuais para seu desenvolvimento e não em requisitos futuros.
- **Entregas frequentes:** a construção do software deve ser simples e, conforme as necessidades aparecem, há uma atualização do mesmo.

- **Metáfora:** são descrições do software sem os termos técnicos, promovendo a compreensão do cliente e direcionando o desenvolvimento de software.
- **Projeto simples:** o programa construído pelo método XP precisa ser o mais simples possível e atender aos requisitos atuais, não se preocupando com os requisitos futuros.
- **Testes:** XP enfoca a validação do projeto durante o processo de desenvolvimento, fazendo com que os profissionais programem o software criando inicialmente os testes.
- **Refatoração:** deve ser realizada somente se for necessária, ou seja, quando um dos profissionais da dupla, ou os dois, notam que é possível simplificar o módulo atual sem perder nenhuma funcionalidade.
- **Programação em pares:** o desenvolvimento do código é realizado em dupla, ou seja, dois profissionais trabalham em um único computador. Enquanto um deles está no controle do teclado e do mouse implementando o código, o outro observa ininterruptamente o trabalho que está sendo realizado, buscando reconhecer os erros sintáticos e semânticos e analisando como se pode aprimorar o código que está sendo desenvolvido. Esses papéis podem e devem ser alternados sucessivamente, viabilizando a aprendizagem mútua.
- **Propriedade coletiva:** o código do projeto pertence a todos os membros da equipe. Isto implica dizer que todos os profissionais ao notarem que podem adicionar valor a um código, ainda que não o tenha desenvolvido, pode efetuá-lo, desde que implemente a bateria de testes necessária. O grande benefício dessa prática é que, se um membro da equipe deixar o projeto antes de sua finalização, a equipe consegue continuar o projeto sem dificuldades, uma vez que todos conhecem as partes do software, mesmo que não seja de forma aprofundada.
- **Integração contínua:** a ideia é interatuar e arquitetar o sistema de software continuamente, mantendo os programadores em sintonia, possibilitando processos ágeis.
- **40 horas de trabalho semanal:** XP sugere não ser necessário fazer horas extras excessivamente. Caso seja preciso trabalhar mais de 40 horas pela segunda semana consecutiva, há um problema grave no projeto que deve ser

solucionado não com o aumento de horas trabalhadas, mas sim com melhor planejamento.

- **Cliente presente:** o cliente necessita estar disponível para solucionar todas as imprecisões, a fim de evitar atrasos e até mesmo construções equivocadas no software.
- **Código padrão:** uniformização na arquitetura do código, para que possa ser partilhado entre todos os profissionais da equipe.

2.5.2. SCRUM

Outra metodologia ágil que apresenta uma comunidade grande de usuários é o Scrum (BEEK, 1999). Sua meta é oferecer um processo apropriado para o projeto cujo desenvolvimento é orientado a objeto. O Scrum detém uma abordagem empírica que adota ideias da teoria de controle de processos industriais no desenvolvimento de softwares, reinserindo conceitos de flexibilidade, adaptabilidade e produtividade. O foco do método é encontrar uma maneira para que os profissionais da equipe atuem de forma flexível para se produzir o software em um ambiente de constantes mudanças.

A ideia principal do Scrum é que o desenvolvimento de softwares envolve diferentes variáveis técnicas e de ambiente: requisitos, recursos e tecnologia, que podem ser alterados durante o processo. Isso faz com que o processo de desenvolvimento seja imprevisível e complexo, demandando flexibilidade para acompanhar as mudanças. O resultado do processo deve ser um software que é realmente útil para o cliente (SCHWABER, 2004).

Importante ressaltar que metodologia do Scrum é análoga aos princípios da XP: equipes menores, requisitos pouco estáveis ou desconhecidos e iterações curtas para gerar visibilidade no desenvolvimento. Entretanto, as dimensões em Scrum diferem de XP (SOMMERVILLE, 2011).

O Scrum decompõe o desenvolvimento em iterações (*sprints*) de trinta dias. Equipes pequenas, compostas por até dez profissionais, são constituídas por projetistas, programadores, engenheiros e gerentes de qualidade. Essas equipes atuam voltadas para as funcionalidades (os requisitos, em outras palavras), determinadas no início de cada *sprint*, sendo a equipe responsável pelo desenvolvimento desta funcionalidade (SCHWABER, 2004).

No Scrum, segundo Schwaber (2004), há reuniões diárias de acompanhamento, sendo de curta duração (por volta de quinze minutos), nas quais são discutidos os pontos realizados desde o último encontro, definindo o que precisa ser desenvolvido até a próxima sessão. As dificuldades localizadas e os fatores de impedimento (*bottlenecks*) são reconhecidos e solucionados. O ciclo de vida do Scrum, de acordo com Schwaber (2004), é subdividido em três fases principais. A seguir o detalhamento destas fases e a Figura 3 representa de forma gráfica o processo.

- **Pré-planejamento** (*Pre-game phase*): os requisitos são descritos em um documento denominado *backlog*. Em seguida, são priorizadas e apresentadas as estimativas de empenho para o desenvolvimento de cada requisito. O planejamento inclui, entre outras atividades, a delimitação da equipe de desenvolvimento, as ferramentas a serem empregadas, os prováveis riscos do projeto, bem como as demandas de treinamento. Após o planejamento, é sugerida uma arquitetura de desenvolvimento, sendo que eventuais mudanças nos requisitos detalhados no *backlog* são reconhecidos, bem como seus possíveis riscos.
- **Desenvolvimento (*game phase*):** as diferentes variáveis técnicas e de ambiente são reconhecidas antecipadamente, observadas e dominadas durante o desenvolvimento. Ao invés de considerar essas variáveis apenas no início do projeto, como no caso das metodologias tradicionais, no Scrum o controle é feito continuamente, o que aumenta a flexibilidade para acompanhar as mudanças. Nesta fase o software é desenvolvido em ciclos (*sprints*) em que novas funcionalidades são adicionadas. Cada um desses ciclos é desenvolvido de forma tradicional, ou seja, primeiramente faz-se a análise, em seguida o projeto, implementação e testes. Cada um desses ciclos é planejado para durar de uma semana a um mês.
- **Pós-planejamento (*post-game phase*):** após a fase de desenvolvimento são realizadas reuniões para avaliar o progresso do projeto e apresentar o software atual para os clientes. Nessa etapa, são efetuadas a integração, os testes finais e a documentação.

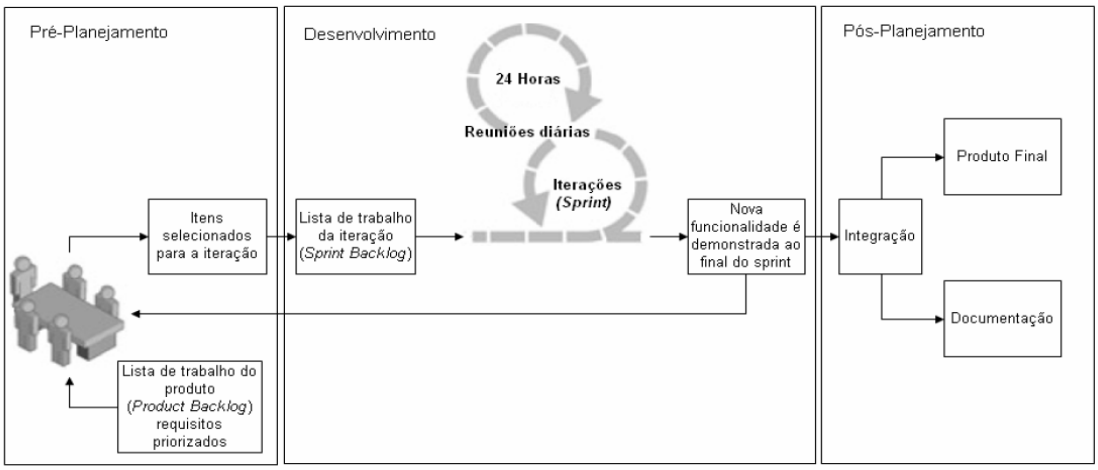

Figura 3 - Visão geral do processo SCRUM (adaptado de SCHWABER, 2004).

2.6. Metodologias Mistas

Nas últimas décadas foram criadas várias estratégias de desenvolvimento de software, conforme Rasmsin e Paige (2008), normalmente, apoiadas em metodologias ágeis ou convencionais. Segundo Alves Paim *et al.* (2011), a natureza complexa do desenvolvimento de software e a ampla variedade de métodos existentes geram comparações difíceis e imprecisas entre metodologias ágeis e tradicionais. Nessa perspectiva, Leffingwell (2006) discute essas diferenças, as quais são sumarizadas na Tabela 4.

Tabela 4 - Características do paradigma tradicional vs. paradigma ágil segundo (LEFFINGWELL, 2006)

Ponto de vista	**Tradicional**	**Ágil**
Medida de sucesso	Conformidade com o plano	Resposta à mudança, código operacional.
Cultura gerencial	Chefia e controle	Liderança / colaboração
Requisitos e arquitetura	Grande e no início	Contínuo / emergente
Garantia de teste e qualidade	Grande, planejado/teste tardio.	Contínuo / concorrente / teste cedo.
Planejamento e cronograma	Detalhado, escopo fixo, tempo e recursos estimados.	Planejamento em dois níveis, data fixa, escopo estimado.

Fonte: (Alves Paim, 2011).

Sumariamente, nota-se que, enquanto o padrão tradicional tem sido indicado a projetos de larga escala e alto risco, o paradigma ágil tem se mostrado mais adequado para projetos de baixo risco e de equipes pequenas (BOEHM; TURNER, 2004; LINDVALL; COSTA, 2004; COHEN; NORD; TOMAYKO, 2006; RAMSIN; PAIGE, 2008). Normalmente, os grandes projetos críticos podem ser prejudicados pela ausência

de rigor e previsibilidade dos métodos ágeis, ao passo que projetos de menor porte e de baixo risco podem ter um custo e prazo elevados sem necessidade, por falta de simplicidade e flexibilidade da abordagem tradicional, que comumente impõe procedimentos complexos e documentação ampla. Diante de todos esses julgamentos, a adaptação de metodologias se tornou uma prática comum na adesão ao processo de produção de software. Dessa forma, as indústrias de software buscam customizar os processos de software em sintonia com sua estrutura organizacional.

2.7. O Processo Unificado

O Processo Unificado enquadra-se no conceito geral de processo: um conjunto de ações efetuadas para transpor os pré-requisitos do cliente em um sistema de *software.* Por sua vez, o Processo Unificado é uma estrutura generalizada que pode ser personalizada, adicionando-se ou eliminando-se atividades segundo as necessidades particulares do cliente, pautadas nos recursos orçados para um projeto (SCOTT, 2003).

O Processo Unificado faz uso extensivo da *Unified Modeling Language* – UML, cujo núcleo está o modelo, o qual na dinâmica de um processo de desenvolvimento de *software* é uma simplificação da realidade que colabora na compreensão da complexidade inerente a sistemas de *software* (SCOTT, 2003).

Nesse contexto, segundo Scott (2003), a UML foi projetada para subsidiar os profissionais que compartilham as atividades de desenvolvimento de *software*, ao auxiliar na visualização do sistema, detalhando sua estrutura e seu comportamento, auxiliando na sua construção, além de documentar as decisões necessárias durante o processo. Muitas das tarefas que o Processo Unificado envolve, abarca uso de UML associado a um ou mais modelos. Suas atividades são baseadas num princípio de trabalho iterativo e incremental. Isto é, um conjunto repetitivo de tarefas e reuniões que apresentam um aumento da produtividade tanto de artefatos quanto do produto, a cada iteração. A seguir, serão apresentadas algumas características do Processo Unificado.

2.7.1. Dirigido por Casos de Uso

Um Caso de Uso é uma sequência de ações, executadas por um ou mais atores (pode ser pessoas ou entidades não humanas fora do sistema) e pelo próprio sistema, que produz um ou mais resultados de valor para um ou mais atores (JACOBSON; BOOCH; RUMBAUGH, 1999). Um dos principais aspectos do Processo Unificado é a

utilização dos Casos de Uso como forma condutora do processo de desenvolvimento de sistemas. A expressão "*dirigido por Casos de Uso"* refere-se ao fato de se utilizá-los para dirigir todo o trabalho de desenvolvimento, desde a captação inicial e negociação de requisitos até a aceitação do código (SCOTT, 2003).

2.7.2. Centrado em arquitetura

O termo centrado em arquitetura serve como alicerce para fundamentar o sistema como um todo. Inclui-se no entendimento deste termo aspectos como desempenho, escalabilidade, reuso e restrições econômicas e tecnológicas (JACOBSON; BOOCH; RUMBAUGH, 1999).

Segundo Scott (2003), no contexto do Processo Unificado, a arquitetura especifica os elementos necessários para descrição da arquitetura.

- Entendendo a Visão Global: Seguindo a modelagem de forma minuciosa e dada a devida atenção à construção dos diagramas UML, isto associada ao texto de apoio, contribuirão em muito para tornar a descrição da arquitetura o eixo central para um melhor entendimento da "visão global" do novo sistema.
- Organizando o Esforço de Desenvolvimento: Definição dos padrões arquitetônicos, a fim de auxiliar a dar forma ao esforço de desenvolvimento em vários níveis. (Cliente/servidor, três camadas e N camadas são exemplos de padrões bem conhecidos).
- Facilitando as Possibilidades de Reuso: Permitir que as equipes de desenvolvimento identifiquem oportunidades para possíveis reuso de qualquer outro componente de outros sistemas facilita o trabalho e otimiza tempo.
- Facilitando a Evolução do Sistema: Estabelecer uma arquitetura sólida para oferecer um conjunto de pontos de referência essenciais sobre os quais o trabalho de desenvolvimento futuro poderá se apoiar.
- Dirigindo os Casos de Uso: Os Casos de Usos dirigem a arquitetura de um sistema, pois dirigem o esforço de desenvolvimento.

2.7.3. As quatro fases do Processo Unificado

A vida de um sistema de software pode ser representada como uma série de ciclos. Cada ciclo termina com a liberação de uma versão do sistema para os clientes. No Processo Unificado, cada ciclo contém quatro fases. Uma fase é simplesmente o tempo decorrido entre dois marcos principais, em que gerentes tomam decisões importantes sobre se prosseguem com o desenvolvimento e, se este for o caso, o que é necessário em relação ao escopo, orçamento e cronograma do projeto (SCOTT, 2003).

A Figura 4 apresenta as fases e os principais marcos do Processo Unificado. Nesta figura, pode-se observar que cada fase contém uma ou mais iterações (SCOTT, 2003).

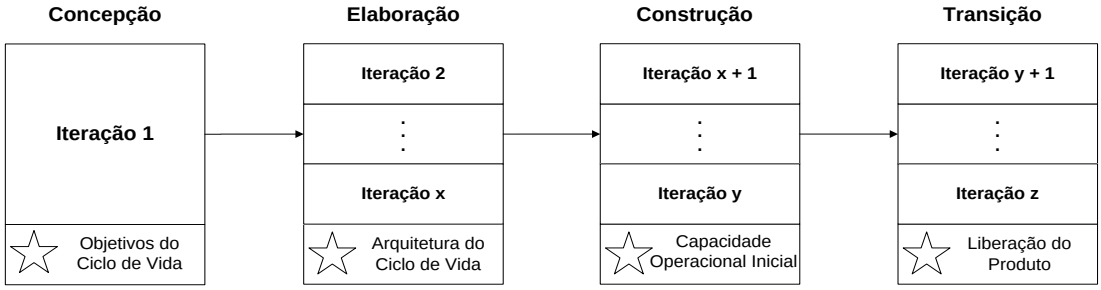

Figura 4 - Fases e Marcos Principais do Processo Unificado (SCOTT, 2003).

A seguir são apresentadas visões gerais de cada fase do Processo:

- Concepção: O objetivo principal da fase de Concepção é estabelecer a viabilidade do sistema proposto. A definição do escopo, a arquitetura candidata, a análise econômica são os aspectos predominantes desta fase.
- Elaboração: Esta fase tem como principais aspectos a captura dos requisitos funcionais válidos restantes, estabelecer uma base arquitetônica, abordar os possíveis riscos significativos e preparar o plano para orientar a próxima fase (construção).
- Construção: Nesta fase fica a responsabilidade da construção do sistema em questão. Este deverá ser capaz de operar bem em ambientes de teste, onde o próprio cliente os realiza.
- Transição: O principal objetivo da fase de Transição é entregar o sistema completamente funcional aos clientes. O principal marco associado à fase de Transição é chamado de Liberação do Produto.

2.7.4. Os Cinco *WorkFlows*

O Processo Unificado trabalha com cinco fluxos que atravessam as quatro fases do modelo: Requisitos, Análise, Projeto, Implementação e Teste.

- Requisitos: Alguns aspectos devem ser observados para definir as atividades do fluxo de requisitos, por exemplo, a definição do campo de atuação que o projeto irá operar. O campo poder ser um banco ou uma indústria. Além do campo o modelo de negócio que em linhas gerais é a necessidade do produto desejado na forma conceitual. Outro ponto importante e que deve ser destacado é busca pelo refinamento dos requisitos, que acontece por meio de reuniões entre os membros da equipe de requisitos e os clientes, detectando as funcionalidades do produto em questão (SCHACH, 2009).
- Análise: O objetivo do fluxo de Análise é analisar as necessidades dos clientes para obtenção dos detalhes a serem compreendidos a fim de resultar no desenvolvimento correto do projeto de software e que facilite a sua manutenção futura. Para exemplificar melhor seria o fluxo de requisitos como a necessidade do cliente expressa em texto natural como português e o fluxo de Análise expresso em notação UML que facilite a compreensão dos trabalhadores para os fluxos de trabalho de projeto e de implementação. (SCHACH, 2009).
- Projeto: Para o fluxo de projeto cabe a construção do modelo ideal, ou seja, o que o produto deve fazer e demonstra como o produto irá fazer isso. De certa forma o fluxo de projeto irá refinar os artefatos do fluxo de análise até que o material fique com uma forma que possa ser implementado pelos desenvolvedores (SCHACH, 2009).
- Implementação: O fluxo de implementação descreve como será implementado os elementos do fluxo de projetos, tais como os códigos-fonte, as bibliotecas e os componentes executáveis do sistema (SCOTT, 2003).
- Teste: As principais atividades do fluxo de teste visam à construção do modelo de teste que serão realizados nas unidades funcionais do sistema. Este modelo será por meio de casos de testes que serão derivados de Casos de Uso, onde os testadores efetuam teste de caixa-preta usando o texto

original de Casos de Uso e o teste de caixa-branca das realizações destes Casos de Uso, como especificado no modelo de análise.

2.7.5. Iterações e Incrementos

Outro aspecto de grande relevância ao PU é ser iterativo e incremental, onde cada fluxo de trabalho consiste em uma série de etapas, e, para realizar esses fluxos de trabalho, as etapas destes fluxos são realizadas repetidamente, até que os membros da equipe de desenvolvimento fiquem satisfeitos por terem chegado a um modelo UML acurado do produto de software que querem desenvolver (SCHACH, 2009).

A natureza dos produtos de software é tal que praticamente tudo tem de ser desenvolvido de modo iterativo e incremental. De acordo com que o projeto de software caminha os melhores diagramas UML possíveis são elaborados, ou seja, à medida que é acumulado mais conhecimento sobre o sistema real que está sendo modelado, os diagramas vão se tornando mais precisos (iteração) e abrangentes (incrementação) (SCHACH, 2009).

A Figura 5 apresenta a essência do enfoque iterativo e incremental para desenvolvimento de *software*.

Figura 5 - Desenvolvimento Iterativo e Incremental (SCOTT, 2003).

2.7.6. Artefatos, trabalhadores e atividades

Para cada *Workflow* do Processo Unificado serão envolvidos elementos chaves para sua aplicação (SCOTT, 2003):

- Artefatos: Qualquer informação envolvida no projeto pode ser considerada um artefato de sistema, por exemplo, os diagramas UML, os protótipos de

interface, textos, códigos fontes ou quaisquer outros documentos envolvidos na construção do sistema.

- Trabalhadores: O Trabalhador é definido pelo Processo Unificado como um papel que uma pessoa irá desempenhar no projeto. Qualquer indivíduo poderá atuar como mais de um tipo de trabalhador.
- Atividades: Cada *workflow* contém várias atividades dentro do projeto, portanto tais atividades vão desde a construção do modelo de domínio até implementação de uma classe.

2.7.7. O Processo Unificado da Rational – RUP

Segundo Scott (2003) o RUP (Rational Unified Process) é uma instância específica e fortemente detalhada do Processo Unificado. O RUP é um produto comercializado pela IBM Rational pelo site www.rational.com/products/rup.

O RUP é uma metodologia para gerenciar projetos de desenvolvimento de software que usa o UML como ferramenta para especificação de sistemas. O RUP é composto por um conjunto de disciplinas que fornecem diretrizes para definição das tarefas e para atribuição das responsabilidades. A Figura 6 apresenta as disciplinas e fases do RUP.

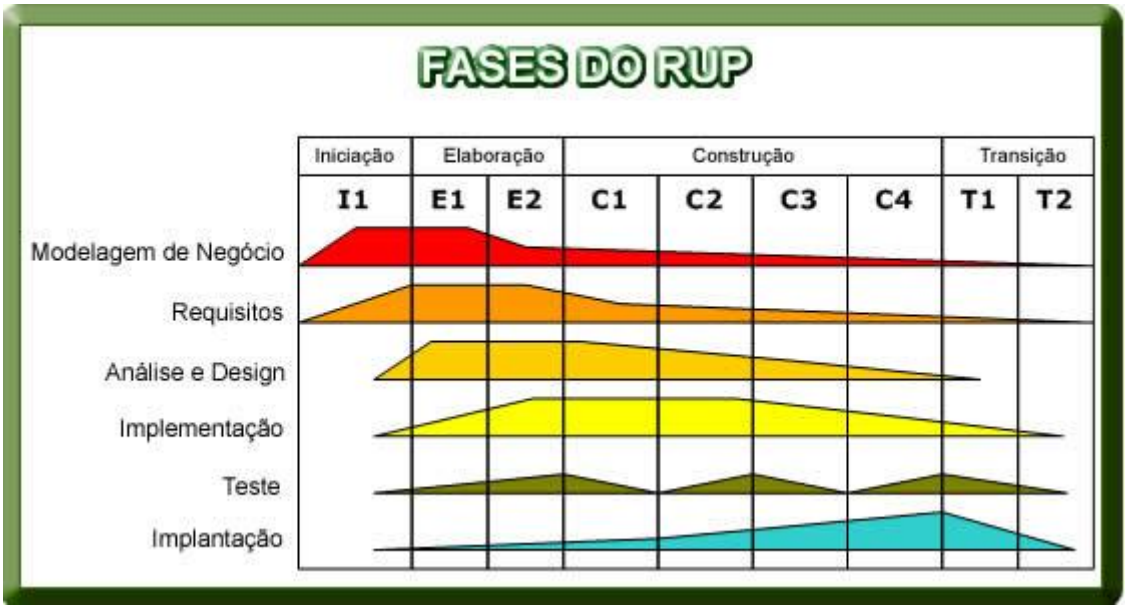

Figura 6 - Visão geral das disciplinas e fases do RUP.

O objetivo do RUP é garantir a criação de softwares de alta qualidade, que atendam às necessidades expressas pelo cliente e pelos usuários, e às restrições de prazo

e custo. Conforme mencionado no Processo Unificado o RUP também é baseado em desenvolvimento iterativo e incremental, gerenciamento de requisitos, arquitetura baseada em componentes, modelagem visual do software, verificação constante da qualidade e controle de mudanças (MARTINS, 2006).

2.8. *Government Resource Planning* (GRPs)

Atualmente, a sociedade está em plena transformação social e econômica, que alcança todos os setores, incluindo as instituições governamentais, que necessitam reorganizar-se para se adaptarem à nova realidade. Esse novo panorama demanda novos modelos de gestão integrada, voltada para a excelência, respeitando-se suas características e particularidades (FERRER *et al.*, 2004).

Esse novo modelo, conforme Ferrer *et al.* (2004) denomina-se *Government Resource Planning* (GRP), em português Sistema Integrado de Gestão Pública e possui como foco basilar o gestor público, profissional-chave para tal mudança. De fato, o GRP oferece ao gestor público um Painel de Controle densamente articulado aos fatos e dados, viabilizando tomar decisões com respostas céleres e eficientes, mesmo com recursos humanos e financeiros insuficientes.

Como o GRP são sistemas baseados em ERP (*Enterprise Resource Planning*) Mendes *et al*. (2002) considera-se que esse modelo consiste em:

- Revisão e automação de processos de negócio;
- Estratégias de redimensionamento, realocação, capacitação e valorização do servidor público;
- Redução de custos;
- Melhoria da qualidade dos serviços prestados.

O GRP traz aos órgãos da Administração Pública ganhos em escala, pois os dados e fatos tornam-se transparentes, pois permitem centralizar ou descentralizar decisões e operações, diminuindo os custos, além de aperfeiçoar o relacionamento com órgãos, setores, poderes relacionados e o cidadão, realizando o ajuste fiscal de modo certo e irreversível (FERRER *et al*., 2004).

Nessa linha de raciocínio, o GRP permite reinventar os processos, uma vez que automatiza os processos-meios, assegurando a incorporação de novas formas de criação de valor, ao deslocar os recursos humanos e físicos das atividades-meio para as

atividades-fim (FERRER *et al*., 2004). Dessa forma, valorizam-se os profissionais e as equipes, já que o servidor terá uma valorização voltada para as oportunidades de exercer e cumprir seu papel, com qualidade e eficiência. Essa estabilização será construída ao longo do processo de transformação, mediante a adaptabilidade individual, por meio de ações de capacitação, requalificação e integração.

Com efeito, o GRP leva ao aumento na qualidade, porque aprimora a interface homem/máquina na prestação de serviços (internos e externos), alterando comportamentos, expectativas, cultura de serviço, confiabilidade, segurança, entre outras dimensões da qualidade de vida (FERRER *et al*., 2004).

Configura-se, segundo Ferrer *et al*. (2004), uma nova imagem do Serviço Público, porquanto além das implicações financeiras (imediatas), esse Novo Modelo de Gestão produz resultados qualitativos de natureza comportamental, que dinamiza as mudanças na opinião pública pelo reconhecimento da qualidade dos serviços proporcionados. Nesse cenário, assegura ao gestor maior visibilidade dos processos e maior domínio sobre os resultados de suas deliberações. Na verdade, o administrador público tem garantido o amparo legal de suas decisões e a continuidade de suas ações.

2.9. Gerência de Projetos de Software

O gerenciamento de projetos, segundo Martins (2006) tem evoluído desde a década de 60. Na realidade, os grandes desafios do século XXI, os quais sugerem a necessidade no aumento da eficiência em todas as atividades, viabilizaram a difusão do conceito de gerência de projetos de software e sua utilização. Importante ressaltar que a disciplina gerência de projetos originou-se junto à indústria bélica e aeroespacial americana, sendo posteriormente utilizada na construção civil e em outras áreas da engenharia.

Na atualidade, o seu conceito passou a ser estendido a diversos segmentos da economia, inclusive à Engenharia de Software. O *Project Management Institute* (PMI) é precursor na uniformização e propagação desse conhecimento. Realmente, o PMI especificou uma gama de procedimentos que objetivam padronizar a teoria associada à gerência de projetos. A teoria de gestão definida pelo PMI está registrada em um documento chamado *Project Management Body Of Knowledge* (PMBOK) (MARTINS, 2006).

Nas pesquisas de Pádua Filho (2009), conforme o PMBoK [PMI04], um projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo. Nesse sentido, o gerenciamento de projetos de software tornou-se uma dimensão efetiva da engenharia de software, uma vez que os projetos necessitam ser gerenciados, considerando-se que a engenharia de software profissional está subordinada aos orçamentos organizacionais e restrições de cronograma.

Para Sommerville (2011), a principal atribuição do gerente de projetos é assegurar que o software atenda e supere tais restrições, além de fornecer ao mercado produtos de alta qualidade. Destaca que o êxito do projeto não é assegurado apenas por um gerenciamento adequado. Contudo, o mau gerenciamento normalmente é consequência da falha do projeto, já que o software geralmente é entregue com atraso, aumentando os custos em relação ao inicialmente estimado, quando não se consegue satisfazer as expectativas dos clientes.

O ciclo de vida de desenvolvimento de software, conforme Pfleeger (2004) detém diferentes etapas, sendo que há aquelas que são repetidas até que o produto esteja concluído, com a concomitante satisfação dos usuários. Dessa forma, os projetos de software podem ser divididos e implementados em uma sucessão de fases, sendo que cada fase comporta diferentes etapas, podendo estas serem subdivididas, quando necessário, a exemplo da Figura 7.

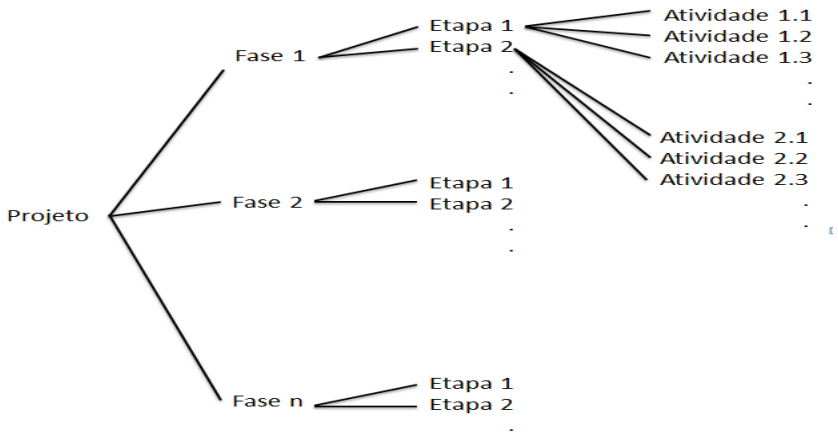

Figura 7 - Fases, etapas e atividades em um projeto (PFLEEGER, 2004).

No entanto, antes de utilizar os recursos destinados a projeto de desenvolvimento e/ou manutenção de software, normalmente o cliente busca uma estimativa do valor do projeto e qual será a sua duração.

2.10. Trabalhos que abordam aspectos de adequações

2.10.1. Adequação de processos nas indústrias de software

Um processo fabril forma-se na confecção de produtos em massa, inclusive operações centralizadas de ampla escala, atividades simples e padronizadas, controles uniformizados, funcionários especializados, mas com poucas habilidades, para a separação de trabalho, mecanização e automação do processo. Dessa maneira, a vinculação do termo indústria ao desenvolvimento de software propõe que se apliquem técnicas para produção em ampla escala, de modo coordenado e com qualidade.

Entretanto, na questão específica de uma fábrica de software, demanda uma organização mais totalizante, que considere diferentes fatores como gestão de pessoas, gestão empresarial, qualidade de software, melhoria de processos e de produtos, uso de ferramentas, entre outros. Nesse cenário, a pesquisa de Rocha *et al.* (2004) concentra no fator processo de desenvolvimento, provendo um mapeamento desde os tipos de processo segundo sua execução até as diferentes concepções de indústria de Software.

Nos anos recentes, Greenfield (2003 apud ROCHA *et al*., 2004) nos apresenta uma visão semelhante, onde o conceito de Fábrica de Software está motivado no desenvolvimento fundamentado em elementos, direcionados a modelos e a linhas de produto de software que distinguiriam uma iniciativa de fábrica, objetivando tornar a montagem de aplicações mais barata por meio de reutilização sistemática, viabilizando a formação de cadeias de produção.

Para Fernandes (2004) as fábricas de software detém um processo estruturado, manipulado e aperfeiçoado de maneira contínua, considerando os enfoques de engenharia industrial, direcionado para o atendimento a diferentes demandas de caráter e escopo distintos. Tudo isso para objetivar a geração de produtos de software, segundo os requerimentos registrados dos usuários e/ou clientes, da maneira mais produtiva e econômica possível.

Normalmente, o processo de software pode ser entendido como o conjunto de atividades indispensáveis para transpor os requisitos do usuário em software. Assim sendo, um processo de software é constituído por um conjunto de fases parcialmente ordenadas, relacionados a conjuntos de artefatos, pessoas, recursos, estruturas organizacionais e restrições e tem como objetivo produzir e manter os produtos de

software finais requeridos. Desse modo, diferentes tipos de informação devem ser agregados em um modelo de processo para indicar quem, quando, onde, como e por que os comandos são efetivados.

No processo tradicional, diferente do processo ágil, existe certa burocracia, já que integra mais tarefas para as suas ações. Assim, pode haver inadaptabilidade, em que a realidade (prazo, escopo, processo, pessoas) é diferente do planejado / registrado, e é fundamentado em *workflows* de processo; direcionado a planejamentos; segurança alta de desenvolvimento; equipes e produtos amplos; projetado para apoiar requisitos correntes e desconhecidos. Dessa forma, o modelo de desenvolvimento de software completo, inclusive suporte a ferramentas; atributos centrados nos fins das atividades; viabiliza ser instanciado e padronizado segundo o domínio da aplicação ou da organização.

Por sua vez, no processo ágil, a escala por equipes grandes ou dispersas e a mudança de cultura organizacional revisa-se o código constantemente. Dessa maneira, a equipe testará o software incluindo o cliente, além tornar a atividade diária, com arquitetura que será definida e apurada todo o momento. No processo ágil, existem interações curtas caracterizadas por minutos e horas, pois se constitui de módulos no nível de desenvolvimento do processo; orientado para as pessoas (ROCHA *et al*., 2004).

Nesse panorama, nas fábricas de software voltadas apenas para o código, que demanda maior agilidade e pouco formalismo na documentação do produto final, os processos ágeis poderiam ser bem empregados, desde que sejam considerados os requisitos mínimos de documentação e gerência requeridos a uma fábrica de software institucionalizada. Isso porque a velocidade e o dinamismo desse tipo de processo confere o ritmo necessário para o sucesso da operação, porém, deve-se preservar pequenas equipes cujos integrantes estejam reunidos em um mesmo ambiente físico.

Com relação às fábricas de componentes de software, os desenvolvedores também podem se adaptar ao escasso formalismo dos processos ágeis, optando por registrar os componentes somente por meio de ferramentas como *JavaDoc*. Nessa perspectiva, assim como na fábrica de código, o dinamismo dos métodos ágeis pode ser o diferencial da fábrica, mas, algum formalismo ou processo complementar precisa ser determinado para a catalogação e repartição dos elementos gerados. Sendo assim, as

restrições quanto a pequenas equipes cujos integrantes estejam reunidos em um mesmo ambiente físico também devem ser observadas.

Nas fábricas de projeto, segundo Rocha *et al*. (2004), que é semelhante a grande parte das fábricas que desenvolvem aplicações personalizadas, a dependência por métodos tradicionais ou vem maior destaque, pois se faz necessário registrar e não raramente, validar formalmente com o cliente as decisões de requisitos ou projeto. Esses produtos também têm um ciclo de vida maior, o que passa a ser corriqueiro, durante o desenvolvimento de um determinado projeto a entrada e saída de pessoal, o que atrapalharia o emprego de um processo orientado a pessoas como os processos ágeis.

Há também as fábricas ou indústrias que lidam com o *outsourcing* de sistemas que demanda, além de maior formalismo, maior manipulação por parte do cliente dos indicadores produzidos pela indústria de software, prática corriqueira nessa situação, orientado a processos tradicionais. É muito comum, e até mesmo exigido pelo próprio cliente, a documentação da fábrica em um modelo de qualidade identificado mundialmente, o que reforça a utilização de processos bem definidos com coleta e acompanhamento de metas de produção.

2.10.2. Adaptação de processos de software

Uma das principais implicações do amplo crescimento experimentado pela indústria do software nos últimos anos é o aumento da complexidade do software e as demandas cada vez maiores do mercado. Tal mercado requer das empresas que desenvolvem software, que os sistemas sejam desenvolvidos com prazo e custo definidos previamente e satisfaçam a padrões de qualidade. Para atender a essas exigências, foi preciso investir no processo de desenvolvimento de software, dada a intensa relação entre sua qualidade e a do software produzido.

Nesse panorama, o processo padrão é um conjunto de componentes fundamentais que se quer incorporar em todo o processo de desenvolvimento estabelecido para os projetos de certa organização de software, assegurando, dessa forma, a consonância com os padrões de qualidade e processos adotados por estas organizações (MAIA, 2004). Como em qualquer software, o processo padrão deve apontar as atividades a serem concretizadas e as amarrações entre elas. Para cada atividade, são confirmadas também as missões dos recursos humanos que a

desempenham, os artefatos de software consumidos e produzidos, como ainda o orçamento necessário.

Considerando-se a relevância do processo padrão e a sua necessidade de adaptação, busca-se, geralmente, um modelo para colaborar com o Engenheiro de Processos na adaptação do processo padrão de uma determinada indústria de software para um projeto específico, fundamentado nas particularidades do projeto, nas diretrizes de adaptação do processo padrão e nos conhecimentos sobre as adaptações realizadas anteriormente.

Convém lembrar que a adaptação é uma atividade que emprega amplamente diferentes tipos de conhecimento, sendo que há modelos sustentados no suporte ao gerenciamento para se obter o conhecimento necessário à adaptação, verificando as características do projeto, os parâmetros que norteiam a adaptação do processo padrão e o conhecimento alcançado durante as adaptações realizadas anteriormente, por meio de mecanismos hábeis em raciocinar sobre tal conhecimento, segundo (MAIA, 2004).

Com relação ao modelo de processo, trata-se de uma estrutura empregada para simbolizar tanto o processo padrão quanto os processos adaptados. Essencialmente, um modelo de processo é constituído pelo conjunto de atividades a serem efetuadas e de dependências entre si. Para toda atividade, podem ser apontados as responsabilidades dos recursos humanos que a desempenham, os artefatos de entrada e saída e os recursos suficientes.

Quanto ás características do projeto, estas podem ser utilizadas para particularizar um determinado software, tais como o tipo de aplicativo a ser desenvolvido, a crítica ao projeto, assim como a repartição geográfica das equipes. Esses projetos caracterizados simbolizam os futuros softwares a serem administrados pela organização, nos quais será aplicado o processo padrão depois de adaptado. Um projeto caracterizado é constituído essencialmente por um conjunto de características de projeto, cada um detendo um valor e um peso (relevância ao projeto).

Entretanto, para Maia (2004), existem certas normas na adaptação que são as diretrizes que a norteiam, específicas de cada processo padrão, cujo teor recomenda em que condições uma determinada atividade do processo será habilitada ou não no processo adaptado, em que a abordagem usada é orientada a atividades. O elemento

condicional de uma norma essencialmente é formada por testes sobre os valores das características de projeto.

No que se referem aos casos de adaptação, trata-se de registros de adaptações já alcançadas, de maneira a conter o conhecimento imprescindível para ser empregado em adaptações futuras, por meio do método de raciocínio baseado em casos. Este caso, na pesquisa de Maia (2004), é constituído pelo problema (projeto de software a ser implementado na organização), a solução (processo adaptado apropriado às características desse projeto) e a avaliação dos efeitos do processo adaptado no projeto de software real.

Na etapa que consiste na caracterização do projeto, fundamentado num planejamento inicial e no conjunto de características disponíveis, o Engenheiro de Processos estabelece as características do projeto de software em questão. Na próxima etapa, a reutilização dos casos, com base no projeto caracterizado, o modelo proposto cultua um método de comparação para buscar, em um repertório de casos de adaptação, o projeto mais análogo e reusar o processo adaptado vinculado, com adaptações se necessário. Nessa etapa, a intervenção do Engenheiro de Processos pode ser precisa, se mais de um caso de adaptação tiver sido resgatado ou em situações em que um componente do processo padrão tiver sido inserido ou excluído no/do processo adaptado restaurado, em decisão oposta à do modelo.

Em seguida, no ato de adaptar o processo, é realizada uma varrição nas atividades do processo padrão em busca daquelas que ainda não foram inseridas no processo originado na fase anterior. Para todas essas atividades, se não houver restrições de uso (condições a serem testadas sobre as características do projeto atual), serão incluídas diretamente no processo adaptado, ou senão, essas condições serão inicialmente testadas.

Nesse aspecto, o resultado dessa etapa é um processo adaptado que pode ser livremente modificado pelo engenheiro de processos na fase de ajuste, conforme as necessidades do projeto. Realizados às adaptações manuais necessárias, nas palavras de Maia (2004), é produzido um caso de adaptação (constituído pelo projeto caracterizado e pelo processo adaptado) e o processo adaptado concluído para ser instanciado (alocação de recursos e agentes, fixação de cronograma) e usado no projeto de software real. Todavia, vale ressaltar que o suporte à instanciação e execução de processos de software está fora do escopo deste trabalho.

3. Trabalhos Relacionados

3.1. Introdução

Este capítulo apresenta os trabalhos relacionados ao tema proposto, evidenciando pontos de contribuições e limitações em cada abordagem.

Vários esforços ocorrem no sentido de adaptar metodologias ágeis e tradicionais a fim de atender as particularidades das organizações. Uma das metodologias mais conhecida e que, constantemente, tem sido investigada nas indústrias de software para adaptação à natureza de uma empresa é o RUP – Rational Unified Process. Este possui as mesmas raízes do Processo Unificado – PU (SCOTT, 2003). Porém, existem várias outras publicações relevantes que utilizam combinação de metodologias ágeis e tradicionais (ALVES *et al*., 2011).

Acredita-se que os processos de desenvolvimento de software sempre são adequados em conformidade com o ambiente aplicado. Alguns trabalhos apresentam adequações de processos ao domínio do desenvolvimento das aplicações, considerando os cenários inerentes a tais domínios: tamanho e qualificação da equipe de desenvolvimento, estrutura da fábrica de software que hospeda a equipe, experiência da gerência e conjunto de artefatos derivados do processo.

A seguir serão apresentados quatro trabalhos correlatos à pesquisa em questão, nestes serão abordados aspectos relevantes de limitações para aplicação do estudo de caso deste trabalho.

3.2. Desenvolvimento ágil de sistemas de Realidade Virtual

A pesquisa apresentada em Mattioli *et al.* (2009) revela particularidades relacionadas à construção de Sistemas de Realidade Virtual, cujo desenvolvimento ágil de software é uma abordagem de atuação, diferenciada pela adaptabilidade, pela habilidade em admitir mudanças, sejam estas nas variáveis de mercado, na configuração do sistema, na tecnologia de criação ou nas equipes de projeto. Para tal, a metodologia ágil XP foi utilizada como metodologia base, onde buscou-se adaptá-la para atendimento as especificidades de sistemas em SRV.

As principais atividades apresentadas por Mattioli *et al*.(2009) são: 1) *Spike* de arquitetura; 2) Requisitos de interatividade; 3) Planejamento da iteração; 4)

Desenvolvimento e 5) Testes. Para Mattioli *et al.* (2009) a interatividade se transformou no elemento central de diversos sistemas de Realidade Virtual, desempenhando um papel de fundamental relevância na deliberação da aplicabilidade desses sistemas. Dessa forma, a análise criteriosa e o estabelecimento dos requisitos de interatividade, que estão intimamente relacionados à crescente tecnologia em ambientes de RV, ocupa uma posição de destaque dentro do processo de desenvolvimento de SRV.

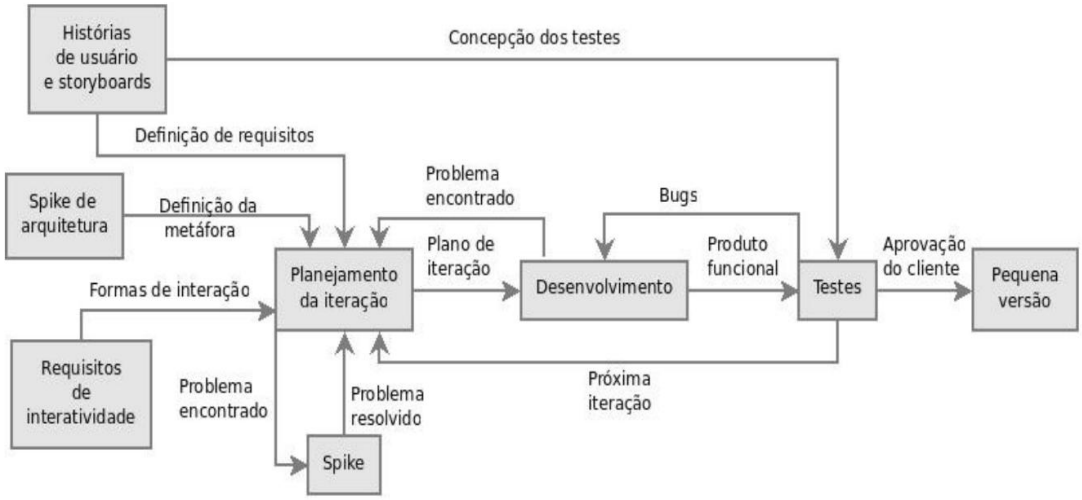

Figura 8 - Ciclo de vida do desenvolvimento ágil de um SRV [extraído de (MATTIOLI *et al.*, 2009)].

Como aspecto relevante ao trabalho de Mattioli *et al.* (2009), a fase de requisitos de interatividade ocupa posição de destaque dentro do processo de desenvolvimento, uma vez que, na construção de soluções de sistemas de Realidade Virtual é primordial ponderar a usabilidade e a interatividade com o sistema. Entretanto, para a aplicação deste modelo em fábricas de software de repartições públicas para construção do sistema integrado, observa-se como fator limitador o tempo gasto para conceber a fase de requisitos de interatividade desnecessária pela característica do projeto.

3.3. Um Método de Desenvolvimento de Sistemas de Grande Porte Baseado no Processo RUP

Este trabalho apresenta um método com as mesmas fases do processo RUP, com a diferença de que elas são realizadas não apenas no escopo do sistema, mas também de cada subsistema, havendo, ainda, um revezamento entre elas ao longo do tempo (Figuras 9 e Figura 10). Isso permite que, após a fase de Transição (implantação) do(s) primeiro(s) subsistema(s), o conhecimento adquirido e os artefatos produzidos em seu desenvolvimento sejam devidamente explorados nos subsistemas que ainda restam, além de possibilitar condição de avaliação do andamento e das estimativas do projeto

como um todo, já que foi adquirida uma amostragem de desenvolvimento do sistema, em cada uma de suas fases (SOUZA *et al.*, 2004).

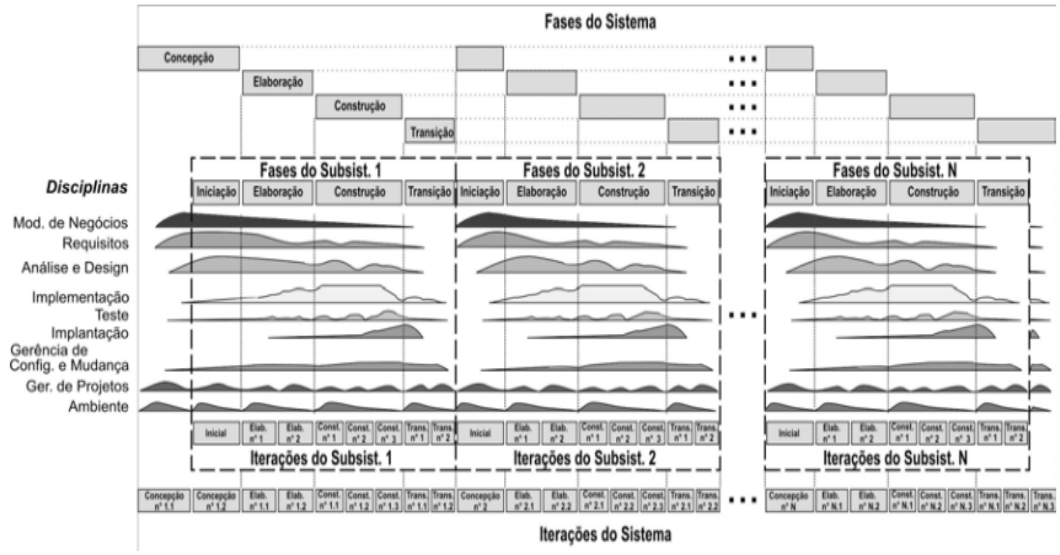

Figura 9 - Fases e disciplinas do método proposto [extraído de (SOUZA *et al.*, 2004)].

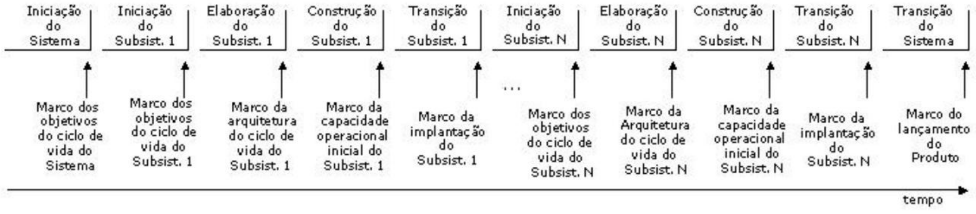

Figura 10 - Fases e marcos do método proposto [extraído de (SOUZA *et al.*, 2004)].

Este processo foi resultado de um estudo empírico no desenvolvimento de um sistema real de grande porte (que durou quatro anos e meio) para gestão de uma operadora nacional de planos de saúde no estado do Ceará (SOUZA *et al.*, 2004).

Como demonstrado, este processo tem como característica fundamental sua aplicação em grandes projetos que contemple fábricas de software bem estruturadas. Neste sentido, para aplicação deste processo em repartições públicas federais (escopo deste estudo de caso) este se torna extremamente inviável. O ponto de maior relevância da inviabilidade é utilização de todas as fases do RUP em todos os subsistemas.

3.4. Adaptação do *Rational Unified Process* (RUP) em pequenas equipes de Desenvolvimento Distribuído

O trabalho de Rocha *et al*. (2008) apresenta uma pesquisa aplicada em adaptar o Processo de Desenvolvimento baseado em RUP para uma pequena equipe utilizando Desenvolvimento Distribuído de Software (DDS).

Para viabilizar este trabalho, a pesquisa foi integrada a uma disciplina de Engenharia de Software com foco na implantação de indústrias de software que utilizam o Desenvolvimento Distribuído de Software (DDS) para a realização de projetos reais. Com a demarcação dos clientes e projetos, buscou-se simular um ambiente de fábrica de software para oferecer os produtos decididos em comum acordo com o cliente. A fábrica relatada nessa pesquisa denominada TechnoSapiens, foi integrada por nove estudantes, atuando de modo distribuído, com parte da equipe executando na mesma cidade e os demais espalhados em três outros municípios. Convém ressaltar que nenhum dos membros havia atuado antes com outro membro da equipe, trabalhando juntos pela primeira vez para o desenvolvimento do projeto.

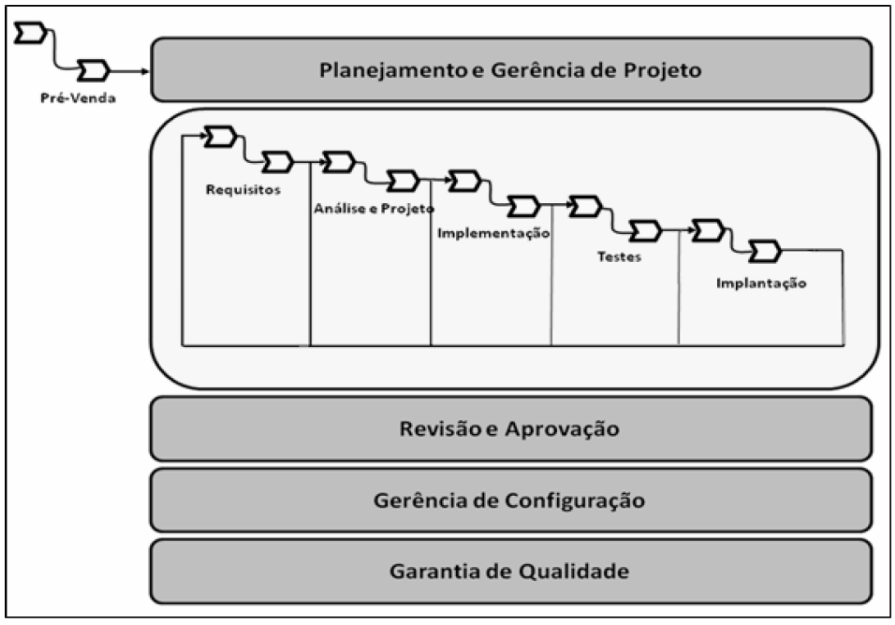

Figura 11 - Processo Adaptação do RUP para DDS [extraído de (ROCHA *et al*., 2008)].

Esses aspectos levaram à definição de uma instância de processo fundamentada no *Rational Unified Process* (RUP). O TechnoProcess, como foi chamado o processo criado pela TechnoSapiens, foi formalizado de forma a viabilizar e atender aos fatores acima referidos, em que a procura por um processo leve, flexível, mas de caráter

prescritivo, demandou reuniões para definir como o processo atenderia a essas particularidades. As principais etapas consideradas para o processo foram: pré-venda, requisitos, análise e projeto, revisão e aprovação, implementação, testes e implantação. A Figura 11 apresenta o processo desta pesquisa (ROCHA *et al*., 2008).

O projeto foi desenvolvido em duas fases: o projeto piloto, usado para fins de averiguação da estrutura da fábrica (infraestrutura, processo, entre outros) e o projeto real, que teve como objetivo utilizar as melhores práticas adotadas no projeto piloto e buscar melhorias. Em meio a pequenos impasses, o TechnoProcess necessitou ser adaptado no decorrer do projeto para realmente atender às necessidades da fábrica. Na fase de adequações do processo, foram definidos ajustes nas atividades, adequando também alguns *templates* dos artefatos propostos.

Nessa linha de pensamento, ao definir um processo de software eficiente e que se adapte ao contexto da indústria é algo de essencial relevância para o avanço de um projeto, por isso, é preciso que este seja constantemente avaliado a fim de conseguir um nível de adequação e funcionalidade cada vez maior.

Além da adaptação do processo RUP, este trabalho destaca o desenvolvimento de sistemas distribuídos, esta característica faz com que este apresente vários problemas na aplicação do processo proposto, pois demanda aspectos para mitigação dos riscos que normalmente são maiores devidos sua característica. Portanto, esta pesquisa se torna inviável em ser aplicada no estudo de caso desta pesquisa.

3.5. Integração do desenvolvimento ágil de software ao RUP

No trabalho de ALVES *et al.* (2011), defende-se a proposta de um processo híbrido Scrum-RU. Nesse caso, as decisões de projeto do processo Scrum-RUP foram, em grande parte, tomadas com base nas experiências encontradas na literatura, assim como nos principais resultados da pesquisa na área de métodos ágeis.

Um estudo de caso industrial foi implementado em uma empresa localizada em Uberlândia-MG, com o intuito de analisar como o processo Scrum-RUP impacta na produtividade de desenvolvimento, em comparação ao processo RUP que a empresa estava acostumada a utilizar.

O trabalho objetivou verificar se há diferença de produtividade entre equipes RUP e equipes Scrum-RUP. Resultados quantitativos demonstraram ganhos expressivos

de produtividade em quatro dos cinco agrupamentos de projetos semelhantes. Esses ganhos variaram entre 15% e 34%, ao passo que no grupo que não apresentou diferença significativa, o ganho foi de -1,1%.

Esses resultados indicam que houve ganho de produtividade nos projetos Scrum-RUP. Por meio deste trabalho, foi também possível inferir, com relativa segurança, que o processo foi realmente responsável pelo ganho de produtividade. A Figura 12 demonstra o ciclo de vida do processo Scrum-RUP.

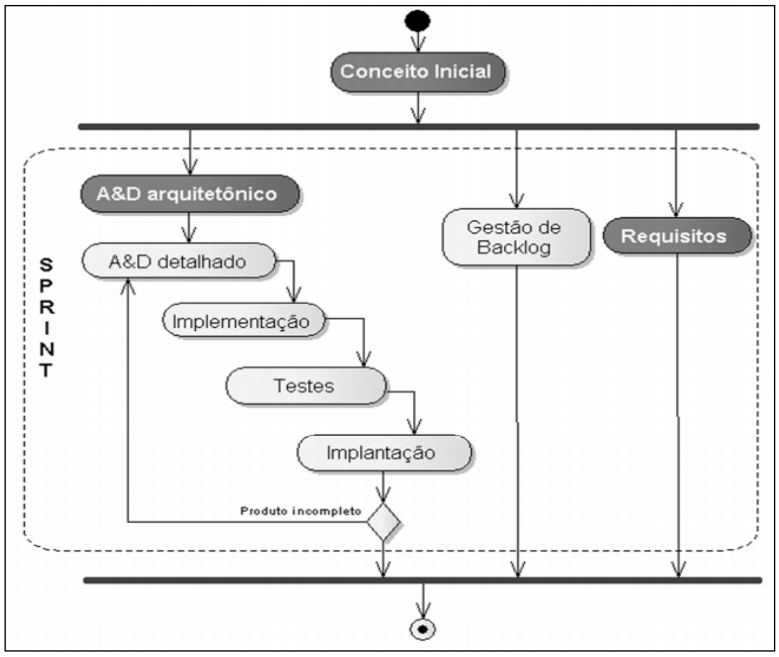

Figura 12 - Ciclo de Vida do processo Scrum-RUP [extraído de (ALVES *et al.*, 2011)].

Para aplicação deste modelo, destacam-se dois aspectos relevantes caso este processo estivesse implantado em ambientes de repartições públicas, tais como: 1) requisitos predominantes por consequência dos clientes on-site; 2) Integração contínua: Necessidade por característica do projeto em integrar todos os submódulos para um melhor entendimento do conceito GRP-IFTM.

3.6. Sumário e conclusões

Para análise comparativa dos trabalhos relacionados, a Tabela 5 apresenta os principais critérios investigados e relacionados com o processo proposto. Estes critérios foram identificados como necessários para a análise, após diversas reuniões no grupo TI do IFTM devido a demanda dos mesmos no dia-a-dia de desenvolvimento de software do grupo.

1) **Iterações e incrementos:** Dividir o projeto em partes gerenciáveis, de forma a incrementar as funcionalidades continuamente até o final da construção do produto (SCHACH, 2009).

2) **Cliente *on-site*:** Utilizado nas metodologias ágeis (SCRUM) e tem como características o cliente próximo ao projeto, ou seja, clientes e desenvolvedores próximos para resoluções de requisitos mal interpretados. Conforme Boehm (2002), Coram & Bohner (2005) e Highsmith & Cockburn (2001) os métodos ágeis funcionam melhor quando os clientes se dedicam junto com a equipe de desenvolvimento (*on site*), e quando o seu conhecimento tácito é suficiente para toda a amplitude da aplicação.

3) **Prototipagens de Interfaces:** é uma prática utilizada para demonstrar como as interfaces (telas, relatórios ou outros componentes do sistema) serão apresentadas aos usuários.

4) **Integração contínua**: Utilizado para o desenvolvimento contínuo do software, onde os membros de um time integram seu trabalho frequentemente, geralmente cada pessoa integra pelo menos diariamente – podendo haver múltiplas integrações por dia (TELES, 2004).

5) **Design simplificado:** A metodologia foca sua ênfase em interações com os clientes a fim de obter melhores resultados para camada de visão. Porém design simplificado reduz tempo e custo devido ser mais fácil de se construir, manter e entender (CÁSSIO, 2006).

6) Reuniões diárias: Como no SCRUM as reuniões são realizadas todos os dias nos mesmos horários e local. É aconselhável que seja a primeira coisa realizada como atividade do dia.

Tabela 5 - Tabela comparativa entre os trabalhos relacionados.

	Ágil SRV	**RUP – N Sub-Sistemas**	**RUP – em DDS**	**SCRUM-RUP**
Iterações e incrementos	x	x	x	x
Cliente on-site		x		
Prototipagens de Interfaces	x			x
Integração contínua	x	x	x	
Design simplificado		x	x	x
Reuniões diárias	x			x

Analisando a Tabela 5, é possível observar que nenhum dos sistemas estudados apresenta todos os critérios de forma conjunta e integrada. Entretanto, acredita-se que uma metodologia que contemple todos estes critérios possua um maior potencial no impacto de produtividade de uma equipe de TI.

Portanto, este trabalho surgiu no sentido de colaborar para o estabelecimento de uma melhor metodologia de desenvolvimento de software, criando aderência às estas particularidades para atendimento a equipes que possuem características de fábricas de software de repartições públicas. Detalhes desta metodologia são apresentados no próximo capítulo.

4. Metodologia Proposta

4.1. Introdução

Este trabalho propõe uma integração do Processo Unificado e metodologia ágil, visando, primordialmente, customizar o mesmo com sensível redução de número de fases e artefatos, a fim de obter melhor gerenciamento do processo de desenvolvimento de software em fábricas com características de repartições públicas. Espera-se assim contribuir na perspectiva da integração de módulos hierárquicos, conceitualmente definidos em projetos corporativos governamentais, referidos como GRPs (*Government Resource Planing* – Sistemas Integrados Governamentais).

A metodologia aqui proposta será referenciada por MDS-GRP que significa Metodologia de Desenvolvimento de Sistemas. O restante do capítulo é voltado para a explanação das fases e dos critérios adotados pela MDS-GRP.

4.2. Objetivo da MDS-GRP

O objetivo da MDS-GRP é direcionar as equipes de desenvolvimento de repartições públicas, a utilizar a metodologia como um guia de referência para construção de módulos e/ou submódulos que compõe sistemas integrados do tipo GRPs. É importante destacar que, embora a metodologia fosse inspirada em processos comuns de repartições públicas relacionadas ao ensino, não existe nenhuma impossibilidade de investigação do uso da mesma em projetos de outra natureza.

Esta MDS-GRP baseia-se em metodologias utilizadas em literaturas acadêmicas para desenvolvimento de sistemas e são apoiadas em técnicas como o PU (Processo Unificado), PMBOK (*Project Management Body Knowledge*) e UML (*Unified Modeling Language*).

Além disso, a MDS-GRP está organizada em papéis, artefatos, projetos e manutenções. Cada projeto contempla uma sequência de fases e cada fase define fluxos de atividades onde são realizadas reuniões de planejamento, acompanhamento e gerados artefatos de controle.

4.3. Características

É importante destacar que esta MDS-GRP será dirigida por Casos de Uso, em conformidade com os fundamentos definidos no PU-Processo Unificado. Para obter um melhor entendimento, esta característica baseia-se em uma sequencia de ações, executadas por um ou mais atores (pessoas ou entidades não humanas fora do sistema) e pelo próprio sistema, que produz um ou mais resultados de valor para um ou mais atores. Esta expressão refere-se ao fato de se utilizar este tipo de diagrama para dirigir todo o trabalho de desenvolvimento, desde a captação inicial e negociação de requisitos até a aceitação do código.

Outro fator relevante é a metodologia ser centrada na arquitetura, este termo também é importante, pois a arquitetura é a organização fundamental do sistema como um todo, ou seja, a arquitetura será o alicerce sobre o qual todo o sistema se erguerá e deverá ser uma das principais preocupações das equipes de desenvolvimento dos módulos e/ou submódulos com objetivo de obter melhores resultados.

Além disso, esta MDS-GRP é iterativa e incremental, ou seja, a execução de um ciclo de vida corresponde a uma versão do sistema liberada. A cada nova versão liberada, melhorias são adicionadas em relação à versão anterior. Esta característica (iterativa e incremental) baseia-se no aumento e refinamento sucessivo de um sistema por meio de múltiplos ciclos após definição da visão geral do submódulos por meio da especificação do projeto. Estes ciclos são formados pelos fluxos de desenvolvimento de análise, de projeto, de arquitetura do sistema, de implementação, de testes e de implantação que atravessam o conjunto das quatro fases que serão apresentadas na próxima seção.

Ressalta-se também a existência de dois outros fluxos nos quais o ciclo de vida do processo proposto é continuo: a fase de “Requisitos”, onde iterações entre as equipes de desenvolvimento e os *stakeholders* envolvidos no projeto, acontecem em momentos informais, ou seja, ocorrem por meio de e-mails, verbal ou outra fonte de comunicação, pois é comum manifestações de dúvidas para resolução de problemas que passaram despercebidos na fase do levantamento dos requisitos.

Outra fase continua é o Gerenciamento de Projetos que representa o acompanhamento das atividades por parte das equipes que estão envolvidas no projeto. Para realização do acompanhamento de tais atividades, foi implantada uma ferramenta

de apoio ao gerenciamento de projetos que possibilite o registro das tarefas realizadas nos projetos. Além disso, esta ferramenta permite também o armazenamento de forma organizada dos artefatos gerados no processo proposto e consequentemente a documentação do projeto como um todo. A Figura 13 apresenta o ciclo de vida do processo proposto.

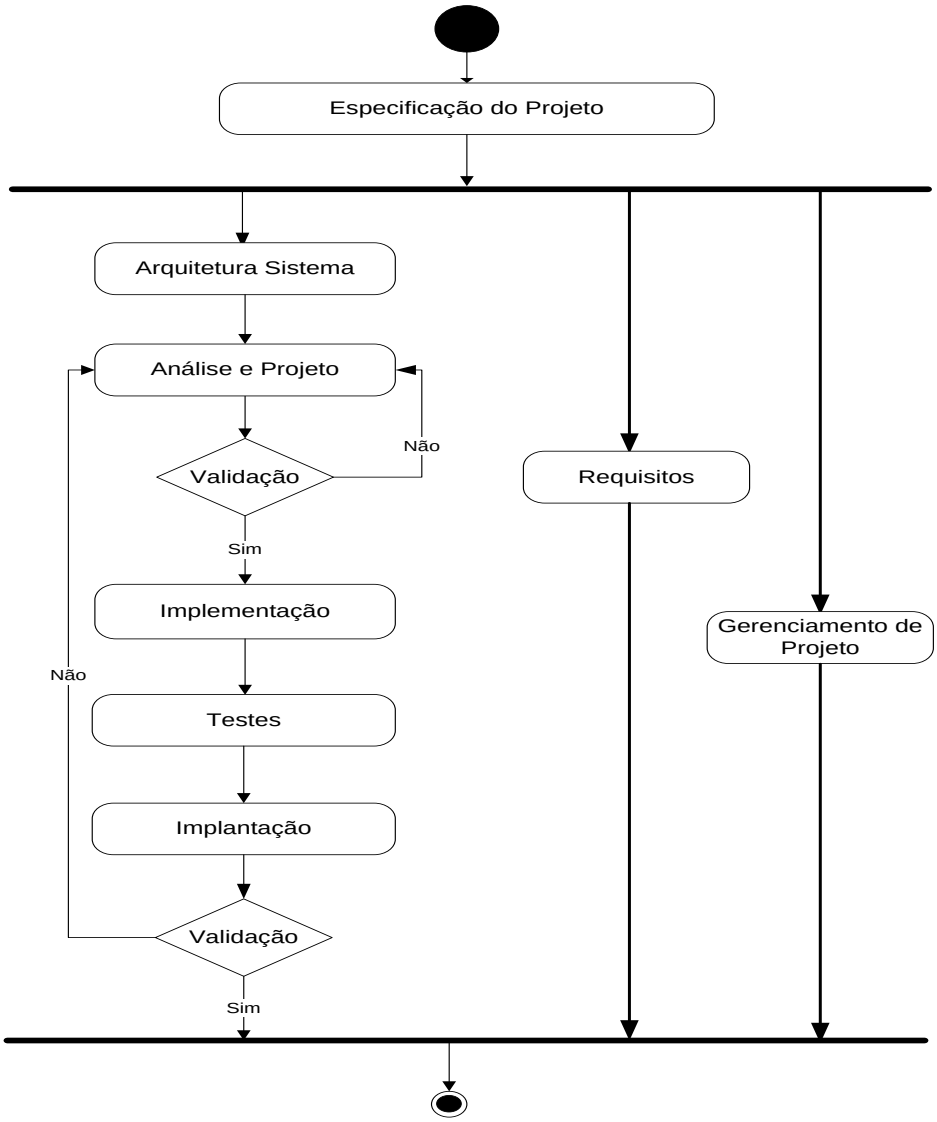

Figura 13 - Ciclo de Vida do processo MDS-GRP.

4.4. As Fases da MDS-GRP

A MDS-GRP está organizada em quatro fases: concepção, elaboração, construção e transição, conforme proposto pelo PU. Entretanto, a natureza de cada fase é diferenciada em relação aos processos tradicionais do PU. De fato, em cada fase é proposta uma adequação de desenvolvimento que foi norteada por necessidades comuns

às equipes de desenvolvimento em ambientes públicos. Tais ambientes possuem, em sua essência, características próximas para desenvolvimento de projetos de software.

A ideia proposta está relacionada com o fato de que em cada fase sejam produzidos um conjunto de atividades e, consequentemente, um conjunto de artefatos para documentar o processo de desenvolvimento. Portanto, ao final de cada fase, espera-se obter tais artefatos, sejam eles diagramáticos ou textuais, dependendo da fase em questão. Abaixo a Figura 14 apresenta a estrutura das fases da MDS-GRP.

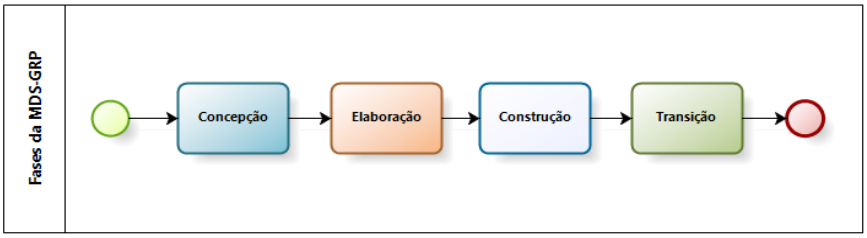

Figura 14 - Fases da MDS-GRP.

4.4.1. Fase de Concepção

A fase de Concepção tem como objetivo principal a formalização do módulo a ser desenvolvido. Esta formalização deverá ser devidamente registrada pelo cliente do módulo com objetivo de identificar a demanda necessária para atender a área de negócio. Este registro será por meio do Documento de Oficialização da Demanda – DOD que é o documento oficial dentro de qualquer entidade pública. Após o preenchimento do DOD serão realizadas análises entre o Gerente de Projetos e o patrocinador para compor a visão inicial do escopo do projeto e registrar no documento Termo de Abertura do Projeto – TAP a fim de iniciar o Plano de Projeto que contempla o planejamento para execução de todo processo de produção do módulo. A Figura 15 apresenta uma visão clara da fase de Concepção.

É por meio destes três documentos que inicia o projeto, e estes deverão apresentar as condições necessárias para construção do módulo a ser desenvolvido. A

Tabela 6 apresenta a descrição sumária dos artefatos, os papéis e o seu fluxo.

A definição do conceito inicial do módulo acontece por meio de reuniões com os *stakeholders* (todos os envolvidos), e destaca-se como principal demandante a figura do patrocinador do projeto, a fim de responsabilizar-se pela concepção e motivação de todos os envolvidos, para iniciar o projeto proposto. Assim, para cada projeto a ser

desenvolvido deverá apresentar o patrocinador como papel chave para o processo e prosseguimento na construção do módulo. Para o sucesso do projeto este patrocinador deverá participar diretamente da fase de Concepção e indiretamente de todas as outras fases.

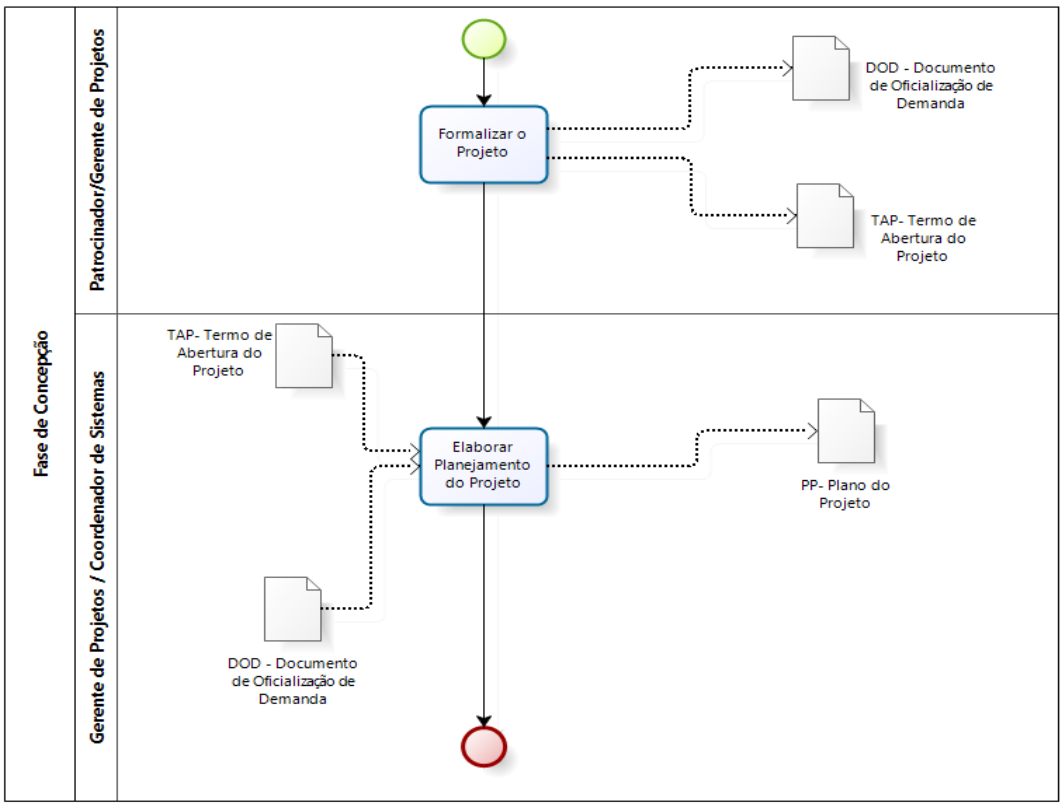

Figura 15 - Fase de Concepção do Projeto.

Tabela 6 - Artefatos padrões para fase de Concepção

Artefato	Descrição	Papéis	Fluxo de Origem
DOD - Documento de Oficialização de Demanda	Documento mediante o qual a área solicitante irá formalizar a demanda. Esta demanda deverá estar devidamente registrada no PDTI do órgão.	Patrocinador e Gerente de Projetos	Concepção
TAP – Termo de Abertura do Projeto	É o documento que autoriza formalmente o projeto. Este documento permite definir mais claramente os objetivos do projeto e quais as suas fronteiras, define o âmbito do projeto bem como o produto final.	Patrocinador e Gerente de Projetos	Concepção
PP – Plano do Projeto	Apresenta uma visão geral das atividades a serem executadas. Este documento irá concentrar informações oriundas do DOD e TAP.	Gerente de Projetos e Coordenador de Sistemas	Concepção

4.4.2. Fases de Elaboração

A fase de Elaboração será realizada pela equipe denominada “Equipe de Requisitos” e terá como papel fundamental, realizar o levantamento dos requisitos da funcionalidade demandada. Está fase deverá realizar as atividades de aproximação e comunicação entre as outras equipes que trabalham em outros módulos do projeto GRP-IFTM que será apresentado no próximo capítulo. Este alinhamento ocorrerá por meio de reuniões semanais entre os Coordenadores de Sistemas das equipes. A Figura 16 apresenta os processos da fase de Elaboração.

Para demonstrar como as iterações deverão acontecer, a equipe de requisitos deverá realizar várias reuniões com os *stakeholders*, para que estes apresentem os conceitos, as definições e os fluxos necessários para atendimento da demanda requerida. Todos estes requisitos devem ser registrados pela equipe de requisitos e posteriormente transformados em protótipos de telas/relatórios. Além dos protótipos, a equipe de requisitos deverá confeccionar os diagramas de Casos de Uso com objetivo de apresentar aos *stakeholders* do módulo o entendimento da interação realizada. Quando ocorrem inconsistências nos requisitos levantados, estes não serem validados e novas interações acontecem para validar a funcionalidade demandada. Este fluxo permanece até a validação dos requisitos. Ao obter o entendimento este deverá ser formalizado pelo artefato “Termo de Aceite”, que conterá a assinatura por um dos *stakeholders* indicado pelo patrocinador ou pelo próprio patrocinador. Ainda nesta fase a equipe de requisitos deverá sempre alimentar o documento *Product Backlog* com as funcionalidades registradas no módulo, bem como as estimativas de tempo para implementar cada uma. A Tabela 7 apresenta os artefatos gerados nesta etapa da fase de Elaboração.

Recomenda-se que para identificação de tais requisitos, o ciclo seja divido em três momentos, sendo o primeiro para levantar os requisitos da funcionalidade; o segundo para apresentar os protótipos e os diagramas de Casos de Uso e realizar as possíveis adequações; e por fim o terceiro para apresentar os ajustes realizados fechando o ciclo da fase de levantamento de requisitos devidamente validado pelo cliente.

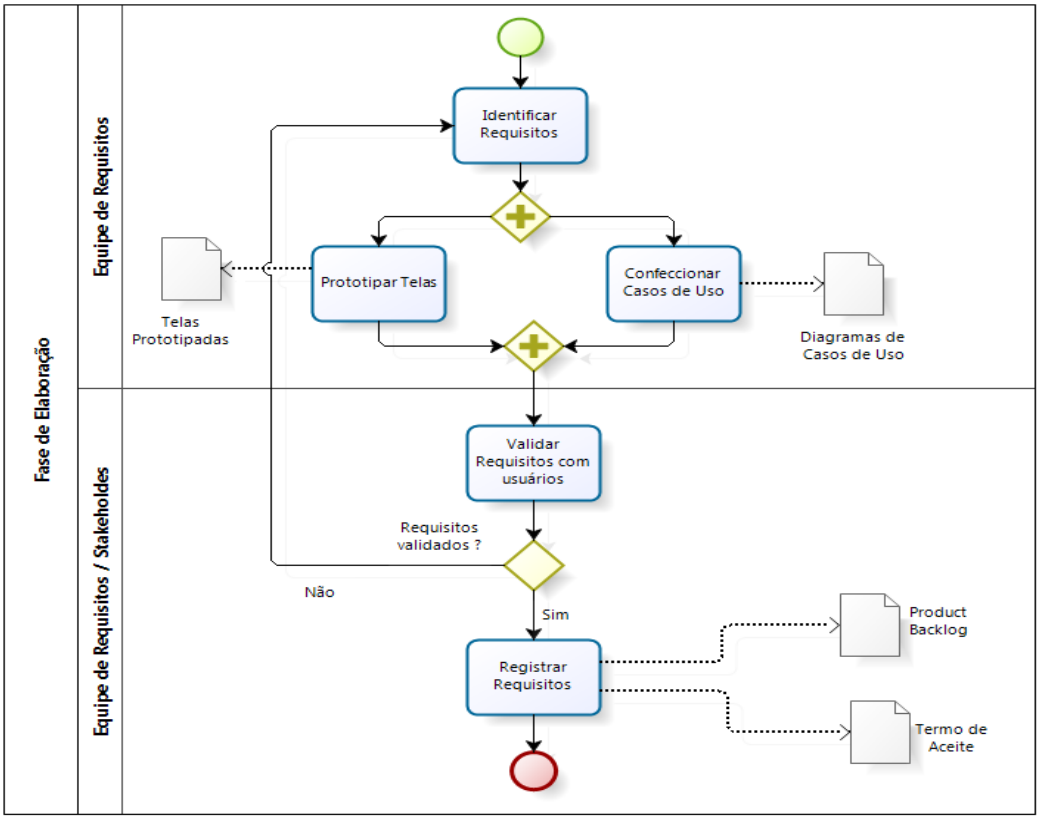

Figura 16 - Fase de Elaboração da MDS-GRP.

Tabela 7 - Artefatos padrões para fase de Elaboração

Artefato	Descrição	Papéis	Fluxo de Origem
Ata de Reuniões	Documento responsável pelo detalhamento e oficialização da reunião.	Equipe de Requisitos	Elaboração
Documento de Especificação de Requisitos	É o documento que descreve os requisitos funcionais do módulo, provendo uma descrição clara e consistente do que o módulo deverá fazer. Este documento deve conter: • Listagem de Casos de Uso; • Diagrama dos Casos de Uso; • Prototipação de Telas e Relatórios;	Equipe de Requisitos	Elaboração
TAR – Termo de Aceite de Requisitos	Documento que será celebrado no aceite dos requisitos por meio das prototipações de telas e dos Casos de Uso apresentados.	Equipe de Requisitos	Elaboração
PB- Product Backlog	É a lista de funcionalidades preparada em conjunto com o cliente. Esta lista é organizada por prioridade de entrega e deve incluir todas as funcionalidades visíveis ao cliente incluindo os requisitos funcionais.	Equipe de Requisitos	Elaboração

4.4.1. Fases de Construção

A fase de Construção refere-se em atividades de arquitetura de sistema, implementação e testes. Para esta fase a equipe responsável por todas as atividades denomina-se “Equipe de Desenvolvimento” e tem como responsabilidade realizar a codificação e os testes do módulo proposto. Para esta fase, foi também projetada a participação dos *stakeholders* na realização de testes em funcionalidades liberadas. A seguir, a Figura 17 apresenta a fase de Construção.

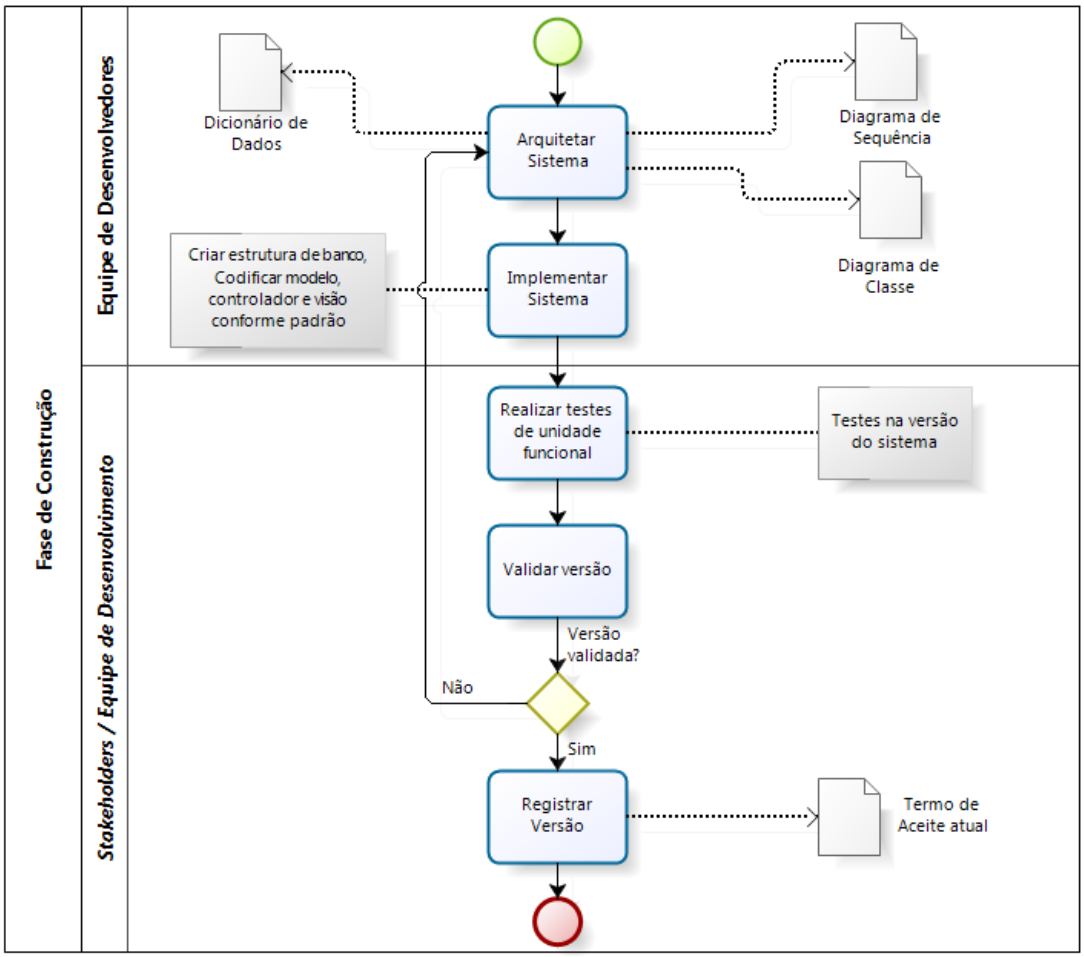

Figura 17 - Fase de Construção da MDS-GRP.

Para esta fase serão gerados além dos artefatos textuais e gráficos, os códigos fontes e toda sua estrutura de dados necessária para produção do módulo. Ressalta-se a necessidade de integração dos módulos já existentes e das estruturas de dados comuns para atendimento a todos os componentes do GRP-IFTM, portanto a equipe de desenvolvimento deve atentar para não haver inconsistências e redundâncias de funcionalidades e informações.

Outro ponto importante e que as equipes devem estar atentas, é a padronização no desenvolvimento em três camadas (modelo, controlador e visão), a fim de estabelecer uma unidade e um alinhamento entre todos os desenvolvedores de todas as equipes. Além disso, as codificações deverão ser orientadas pelos artefatos definidos pela equipe de requisitos e estes só poderão ser codificados após o termo de aceite assinado pelo requisitante envolvido na iteração. A Tabela 8 apresenta os artefatos produzidos nesta etapa.

Tabela 8 - Artefatos padrões para fase de Construção de desenvolvimento.

Artefato	Descrição	Papéis	Fluxo de Origem
Projeto de Banco de Dados	Especifica a organização do banco de dados, incluindo a sua Estrutura lógica e física, conteúdo e aplicações. Inclui o dicionário de dados e o diagrama de classe.	Equipe de Desenvolvimento	Construção
TAB – Termo de Aceite da Versão atual	Documento que será celebrado para atestar a versão atual disponível após implementação e testes da iteração	Equipe de Desenvolvimento	Construção

4.4.2. Fase de Transição

Por fim a fase de Transição culmina no fechamento do ciclo de desenvolvimento do módulo. Nesta fase será confeccionado um relatório que deverá registrar as possíveis integrações com outros módulos, o treinamento dos usuários que irão operacionalizar o módulo e o termo de aceite final do módulo devidamente assinado por seu patrocinador. A Figura 18 apresenta os processos desenvolvidos na fase de Transição e a Tabela 9 os artefatos gerados pelo processo.

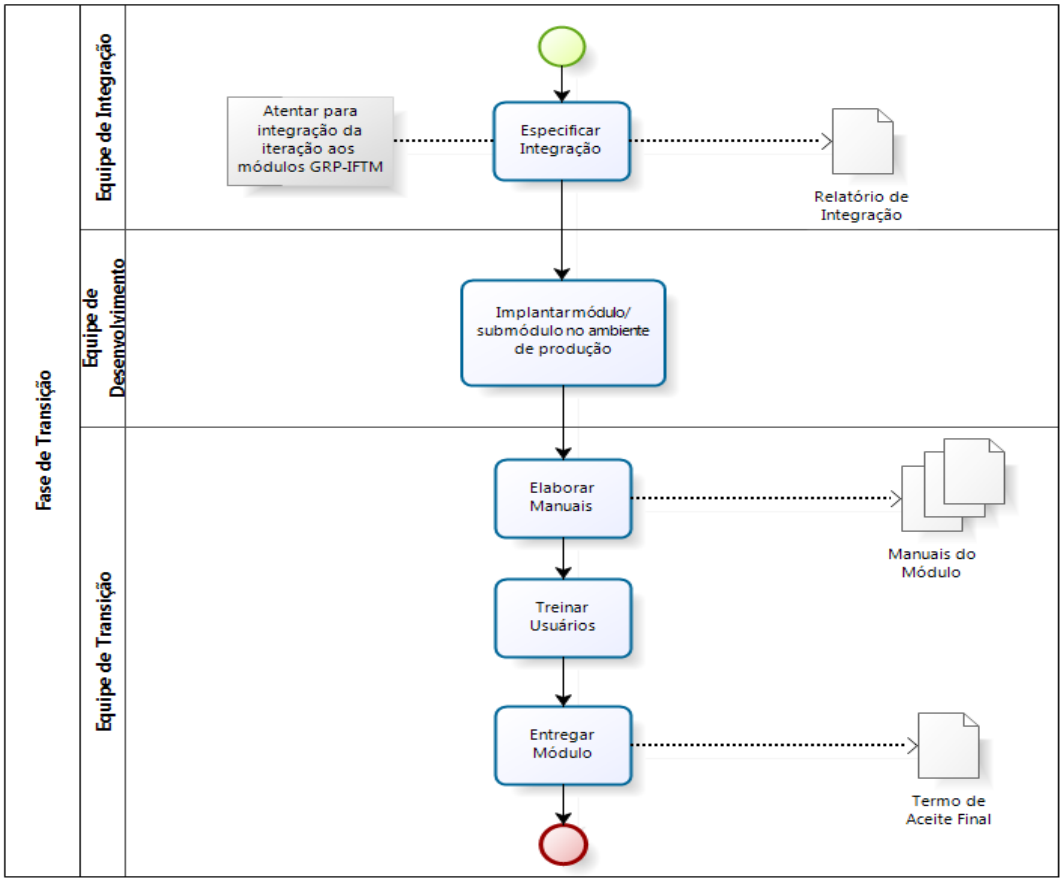

Figura 18 - Fase da Transição da MDS-GRP.

Tabela 9 - Artefatos padrões para fase de Transição

Artefato	Descrição	Papéis	Fluxo de Origem
RIM – Relatório de Integração do Módulo	Relatório contendo informações pertinentes a possíveis integrações a outros módulos;	Equipe de Integração	Encerramento
Manual do Usuário	Documento contendo todas as informações necessárias para operacionalização do módulo.	Equipe de Transição	Encerramento
TAB – Termo de Aceite final	Documento que será celebrado para atestar a versão final após todas as fases do processo de desenvolvimento	Equipe de Transição	Encerramento

4.5. Papéis

Um papel define o comportamento e responsabilidades de um profissional ou grupo de profissionais que participam do desenvolvimento do projeto. O

comportamento é representado por meio das atividades que cada papel deve desempenhar ao longo do projeto. As responsabilidades normalmente estão associadas aos artefatos que cada papel deve produzir e manter ao longo das atividades que realiza. Na prática, um mesmo papel pode ser desempenhado por mais de uma pessoa, assim como uma mesma pessoa pode assumir vários papéis ao longo do projeto.

4.5.1. Patrocinador do Projeto

Este papel é um dos mais importantes para o processo, pois o Patrocinador é quem irá patrocinar no projeto, caso isso não ocorra há grande probabilidade do projeto fracassar. O Patrocinador normalmente são os profissionais que estão à frente das Pró-Reitorias ou dos Departamentos demandante que se responsabiliza pelo projeto. Poderão existir projetos com mais de um Patrocinador. Em destaque a principal atividade do Patrocinador é garantir a sustentabilidade do projeto realizando ações estratégicas tais como:

- Preencher o documento de oficialização da demanda;
- Definir o escopo inicial do projeto;
- Aprovar os objetivos e metas do projeto;
- Tomar decisões mais importantes do projeto;
- Aprovar a conclusão de uma fase para início da seguinte;
- Ajudar a remover obstáculos do projeto;
- Envolver no planejamento do projeto.

4.5.2. Clientes do Projeto

Os Clientes do Projeto são os usuários do módulo a ser desenvolvido, ou seja, e quem o produto ou serviço é concebido e que explora, pelo menos, uma das suas funções do sistema. O Patrocinador irá selecionar alguns usuários a fim de participar na fase de execução dos projetos para levantar os requisitos necessários.

4.5.3. Gerente do Projeto

Gerencia o processo de desenvolvimento por meio da aplicação da metodologia. Define a metodologia e as diretrizes de desenvolvimento de sistemas. Atua no diálogo com os gestores e na resolução de problemas. Outra atividade importante é a definição das equipes para construção do módulo.

4.5.4. Coordenador de Sistemas

É o responsável pelo acompanhamento do projeto. É ele que oficializa as demandas de construção do software, solicita reuniões, define, valida, recebe e homologa os requisitos levantados. Tem poder de decisão sobre todo o processo que o sistema vai automatizar. Este papel define também as duas equipes que irão trabalhar na fase de Construção.

4.5.5. Equipe de requisitos

Equipe responsável por identificar, organizar e documentar todos os requisitos funcionais e não funcionais do módulo apresentados pelos clientes do projeto. São os responsáveis pelos artefatos de protótipos da interface do usuário (telas e relatórios), os diagramas de Casos de Uso, *Product Backlog* e Termo de Aceite de Requisitos.

4.5.6. Equipe de Desenvolvedores

Equipe responsável por arquitetar, implementar e testar o módulo. Estes processos geram os seguintes artefatos: diagrama de classe, diagrama de sequencia e dicionário de dados.

A seguir as principais atividades de responsabilidade das equipes de desenvolvedores:

- Construir unidades de implementação de acordo com as especificações e prazos estabelecidos;
- Efetuar teste unitário do componente com a massa de testes elaborada;
- Criar e testar código de acordo com o padrão de programação estabelecido;
- Executar todas as atividades necessárias para geração dos módulos, de acordo com o planejamento e a esta MDS-GRP;
- Apoiar o gerente de projetos em relação aos aspectos técnicos e funcionais do projeto ou manutenção;
- Gerar versão do sistema;
- Informar ao gerente de projetos o andamento e a evolução de suas atividades no projeto ou manutenção;
- Implantar o módulo no ambiente de produção.

4.5.7. Equipe de Integração

Esta equipe possui um caráter estratégico, é responsável pela elaboração do Relatório de Integração. Este relatório deverá ser explicitado toda integração do módulo a outros módulos já existentes, bem como as possibilidades de haver outras integrações com módulos previstos.

4.5.8. Equipe de Transição

A equipe de transição se responsabiliza pela elaboração de todo material para treinamento (apresentações, manuais, tutorais e outros). Estes materiais são utilizados pelos usuários na capacitação e operação dos módulos, portanto esta capacitação inicial dos módulos também faz parte do portfólio de atividades da equipe. Outra atividade de supra relevância é a entrega final do módulo que deverá ser devidamente registrada no termo de aceite final.

5. Exemplo de Aplicação – GRP do IFTM

5.1. Introdução

De forma a definir prova de conceito, aplicou-se a MDS-GRP em três equipes equivalentes da fábrica de software do Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro com objetivo de avaliar seus pontos fortes e fracos, para alcançar melhores resultados qualitativos no processo de desenvolvimento dos sistemas no âmbito do IFTM.

5.2. A Empresa

Os Institutos Federais de Educação, Ciência e Tecnologia foram criados em dezembro de 2008 com a publicação da lei 11.892 com o objetivo de atender a educação profissional e tecnológica no contexto social do Brasil. Em consequência as antigas autarquias Centro Federal de Educação Tecnológica de Uberaba – CEFET-Uberaba e Escola Agrotécnica Federal de Uberlândia – EAF-Uberlândia, uniram-se e constituíram a nova instituição, o Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro– IFTM. O IFTM atualmente conta com Reitoria na cidade de Uberaba e quatro câmpus (Uberaba, Uberlândia, Ituiutaba e Paracatu) e dois câmpus Avançado (Uberlândia e Patrocínio). Atualmente, conta com aproximadamente 700 servidores efetivos para atendimento a 3000 alunos presenciais e 5000 alunos a distância.

5.3. As Equipes

Para atendimento as demandas da área de Tecnologia da Informação e Comunicação, a estrutura organizacional da instituição possui a Diretoria de Tecnologia da Informação e Comunicação, a fim de atender todos os projetos de software no âmbito do IFTM. Entretanto, para este estudo de caso, foram envolvidos os colaboradores das equipes de sistemas conforme estão organizadas na Tabela 10. Estes foram divididos em três equipes, sendo um coordenador de sistemas e dois desenvolvedores para cada equipe alocada. Todavia, estas equipes trabalharam de forma integrada com objetivo de sincronizar as ações de desenvolvimento dos projetos de software.

Tabela 10 - Identificação das equipes

Identificação da equipe	Quantidade de colaboradores
EQ1	Um Coordenador e dois desenvolvedores
EQ2	Um Coordenador e dois desenvolvedores
EQ3	Um Coordenador e dois desenvolvedores

5.4. Contextualização do problema do estudo de caso

Os Institutos Federais são organizações novas - criados em dezembro de 2008 - com pouco tempo para conceber a nova estrutura organizacional. Apesar da absorção de antigas autarquias a concepção dos institutos possuem caraterísticas amplamente diferentes. Consequentemente os modelos administrativos e gerenciais passaram por severas mudanças com a implantação deste novo modelo de instituição, exigindo soluções emergentes na automatização destes processos.

Foram realizadas várias investigações para compor o inventário dos softwares existentes no âmbito do IFTM e observou-se, como eram desenvolvidos os sistemas implantados nas antigas autarquias. Percebeu-se que não existia nenhuma metodologia formalmente registrada, apenas ações individuais, onde pequenos sistemas eram desenvolvidos para atendimento a particularidades de setores por meio da metodologia código-produto. Diante deste contexto, não foram encontrados artefatos de nenhum dos sistemas implantados, ficando como documentações apenas códigos-fontes não estruturados de alguns softwares.

Neste cenário, o IFTM apresentou vários problemas, tais como: integração de sistemas, protocolos de comunicação entre pessoas e processos; significativo tempo gasto na detecção de solução de problemas nos sistemas e outros oriundos da falta de documentações. É importante destacar que cenários como este apresentado pelo IFTM, acredita-se que várias outras instituições com mesmas características se encontram.

Para resolver estes problemas, é importante destacar que em qualquer organização a informação é um bem precioso, e para garantir o seu fluxo nos diversos segmentos da organização, é necessário sistemas de informação que sirvam como ferramenta fundamental na realização de seus objetivos. Entretanto, a exigência do dinamismo e da efetividade nas atividades públicas vem sendo acelerada juntamente

com o ritmo das necessidades de estratégias organizacionais, diminuindo o ciclo das informações-decisões-ações.

No entanto, para atendimento de tais objetivos, apresentou-se como solução o desenvolvimento de sistemas com estas características e estes são denominados sistemas ERP (*Enterprise Resource Planning*). Outro termo também utilizado por órgãos públicos, porém pouco explorado, são os GRPs (*Governamment Resource Planning*), que são sistemas de informação que integram a maioria das operações de uma instituição em um único software.

É neste cenário que o IFTM (local do estudo de caso) se encontra. De forma a validar a proposta e verificar a adequação da mesma aos cenários rotineiros dos times de desenvolvimento de software, elaborou-se uma proposta de sistema (GRP-IFTM *Government Resource Planning* do Instituto Federal do Triângulo Mineiro), fundamentado na estratégia de desenvolvimento de software que é alvo desta pesquisa.

Para tal aplicação, foram entrelaçados os diversos segmentos do IFTM (equipes de desenvolvimento, acadêmicos, docentes, administrativos e apoio), de forma a alcançar patamares de eficácia e eficiência no cumprimento de sua missão institucional. Destacam-se os benefícios desejados, no domínio da aplicação de softwares:

- Reduzir a redundância de atividades na organização;
- Eliminar o uso de interfaces manuais;
- Otimizar o fluxo da informação e a qualidade da mesma dentro da Instituição;
- Agilizar processos de tomada de decisão;
- Gerar indicadores, mostrando uma visão real da atual situação da Instituição.

É importante ressaltar que este sistema estará integrado ao Portal "VIRTUAL-IF" (Portal de Intranet do IFTM) e aos sites de todos os câmpus do IFTM, a fim de dinamizar as informações publicas nestes portais.

5.4.1. O Projeto GRP-IFTM

Como a MDS-GRP será aplicada para construção de módulos do GRP-IFTM, esta seção irá apresentar o projeto GRP-IFTM.

Este projeto tem como objetivo, integrar os processos operacionais e gerenciais desenvolvidos no âmbito do IFTM. A Figura 19 apresenta uma visão conceitual da divisão de seus módulos, bem como suas integrações nos eixos centrais.

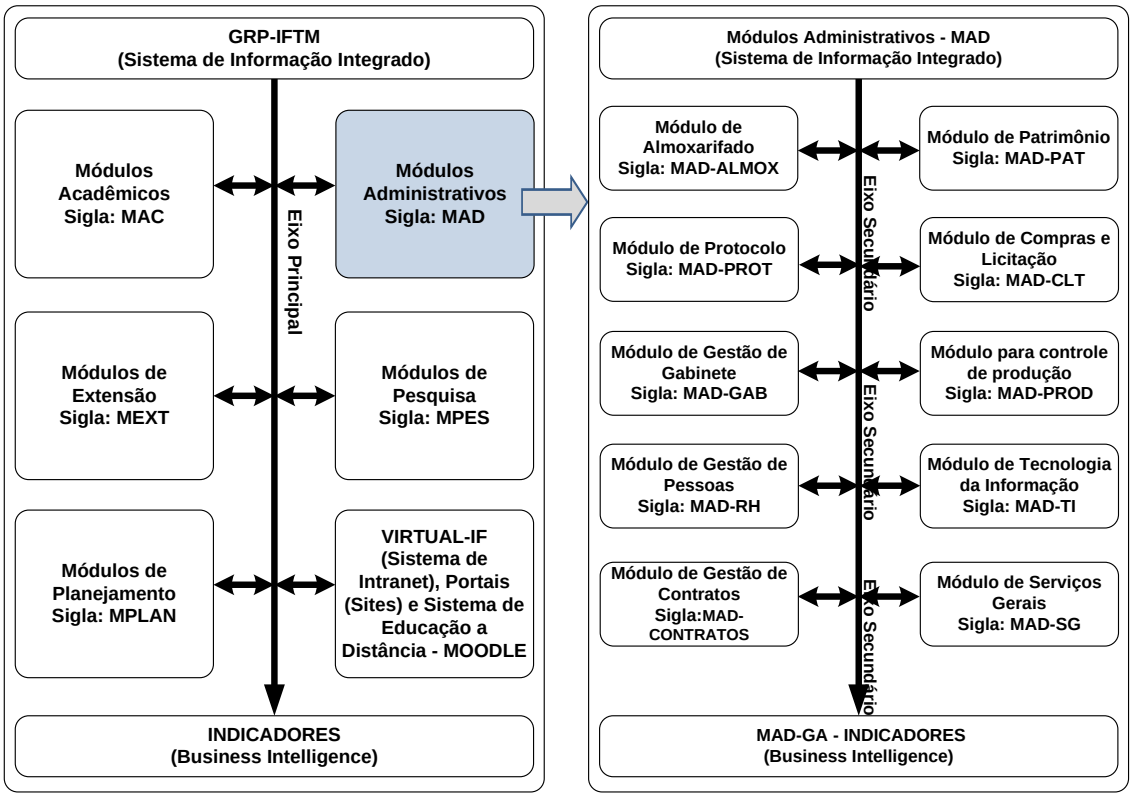

Figura 19 - Visão conceitual do projeto GRP-IFM.

Para este estudo de caso foram separados os submódulos MAD-ALMOX, MAD-GAB e MAC-PROT para aplicação da metodologia proposta.

A distribuição dos submódulos para equipes foi organizada conforme Tabela 11.

Tabela 11 - Distribuição dos submódulos entre as equipes

Equipe	Módulo	Descrição do submódulos
EQ01	MAD-ALMOX	Módulo de Almoxarifado
EQ02	MAC-PROT	Módulo de Protocolo Acadêmico
EQ03	MAD-GAB	Módulo Gerenciador de Gabinete

6. Resultados e Limitações da Metodologia

6.1. Introdução

Conforme mencionado, para desenvolver um sistema computacional com características de um GRP, é necessária a definição de métodos e técnicas para acompanhamento de todo processo de produção do mesmo. Portanto, para estes módulos iniciais aplicou-se a MDS-GRP para servir como guia de referência no ciclo de vida de desenvolvimento dos projetos. Esta MDS foi adotada por todos os desenvolvedores envolvidos, com objetivo de permitir o acompanhamento das atividades no processo de construção do GRP-IFTM.

A adoção da metodologia teve como foco melhorar as práticas na gestão de projetos de software. Este capítulo apresenta uma avaliação da adoção da MDS-GRP. A proposta aqui é apresentar os resultados e limitações identificadas durante o uso da metodologia no departamento de TI do IFTM Uberaba.

6.2. Avaliação da MDS-GRP

A fim de avaliar a MDS-GRP, foi adotado um treinamento com todos os integrantes das três equipes (EQ1, EQ2 e EQ3, conforme apresentado no Capítulo 5). Este treinamento teve como objetivo principal apresentar a MDS para as equipes, a fim de obterem os conhecimentos necessários para sua aplicação. Além disso, também foram apresentadas todas as fases, os papéis e os artefatos que seriam gerados. Finalmente, foi apresentado o ambiente de gerenciamento de projeto denominado *Redmine*, como ferramenta responsável para acompanhamento das atividades/tarefas executadas no projeto do módulo em questão e permitir o armazenamento dos artefatos produzidos.

Para avaliar a metodologia em seus diversos estágios, o projeto denominado MAD-GAB (Módulo de Gerenciamento do Gabinete – módulo que controla as atividades, demandas, documentos oficiais e outras tarefas administrativas associadas à Reitoria do IFTM) foi escolhido como estudo de caso. Esta escolha se baseou no nível de envolvimento e domínio da equipe no referido projeto. A seguir, são apresentados de forma cronológica os eventos (passos) associados com a avaliação da metodologia.

Passo 1: A primeira interação, prevista como atividade na fase de Concepção proposta, ocorreu entre o Patrocinador do projeto (normalmente, alguém da alta gestão) com o Gerente de Projetos (Figura 15). Nesta reunião, foram estabelecidas as fronteiras do projeto e obteve-se o entendimento do escopo inicial do mesmo. Com isto foi gerado o primeiro artefato projetado para a metodologia e definido como Documento de Oficialização de Demanda – DOD. Nesta mesma interação, foram apresentadas pelo patrocinador as funcionalidades macros (Agenda Privada e Pública do Reitor, Cadastro de Demandas, Emissão de Documentos e Gerenciador de Reuniões) para atendimento da demanda requerida. Estas funcionalidades foram usadas para fomentar outro artefato previsto: o Termo de Abertura do Projeto – TAP.

De posse destes dois artefatos, DOD e TAP, foi possível elaborar o Plano de Projeto (PP), onde foi estabelecido um cronograma estimado, para as próximas interações. Como tarefa para o Gerente de Projetos, coube à organização das atividades e tarefas do projeto no ambiente de Gerenciamento– *Redmine.* Dentre estas atividades, incluem-se: cadastrar os *stakeholders* envolvidos e referenciados por seu patrocinador, definir os papéis do time para trabalhar nas equipes específicas nas fases do projeto e postar os primeiros artefatos gerados nesta iteração (DOD e o TAP). Uma observação importante para atender os padrões do GRP-IFTM é a definição do nome do módulo, seguido de sua sigla para identificação no ambiente *Redmine.*

Passo 2: As próximas iterações ocorreram entre a equipe de requisitos e os *stakeholders* responsáveis pelo conhecimento da área específica discutida (chefe de gabinete, secretária executiva e assistente administrativa do gabinete). Na primeira iteração do time, foram apresentados os artefatos anteriormente gerados (DOD, TAP e Cronograma Estimado) e iniciou o primeiro ciclo de discussões para levantamento dos requisitos do sistema, orientados pelas funcionalidades do mesmo, conforme previsto na fase de Elaboração (Figura 16). Na prática, a implementação de uma funcionalidade gera vários requisitos. Por exemplo, uma funcionalidade do módulo MAD-GAB, aqui denominada “Gerencia de Demandas”, gerou vários requisitos necessários para sua construção. Para cada funcionalidade/requisitos identificados, a equipe responsável registrou estas informações em formulário próprio. Além disso, como previsto, foi elaborada uma ata com as discussões promovidas por todos os presentes.

Passo 3: Em seguida, as equipes responsáveis por cada grupo de requisitos, seguindo a metodologia, passaram a elaborar os protótipos de tela e os diagramas de Casos de Uso. A título de ilustração, a Figura 20 apresenta um protótipo de tela criado para a funcionalidade "Gerência de Demandas".

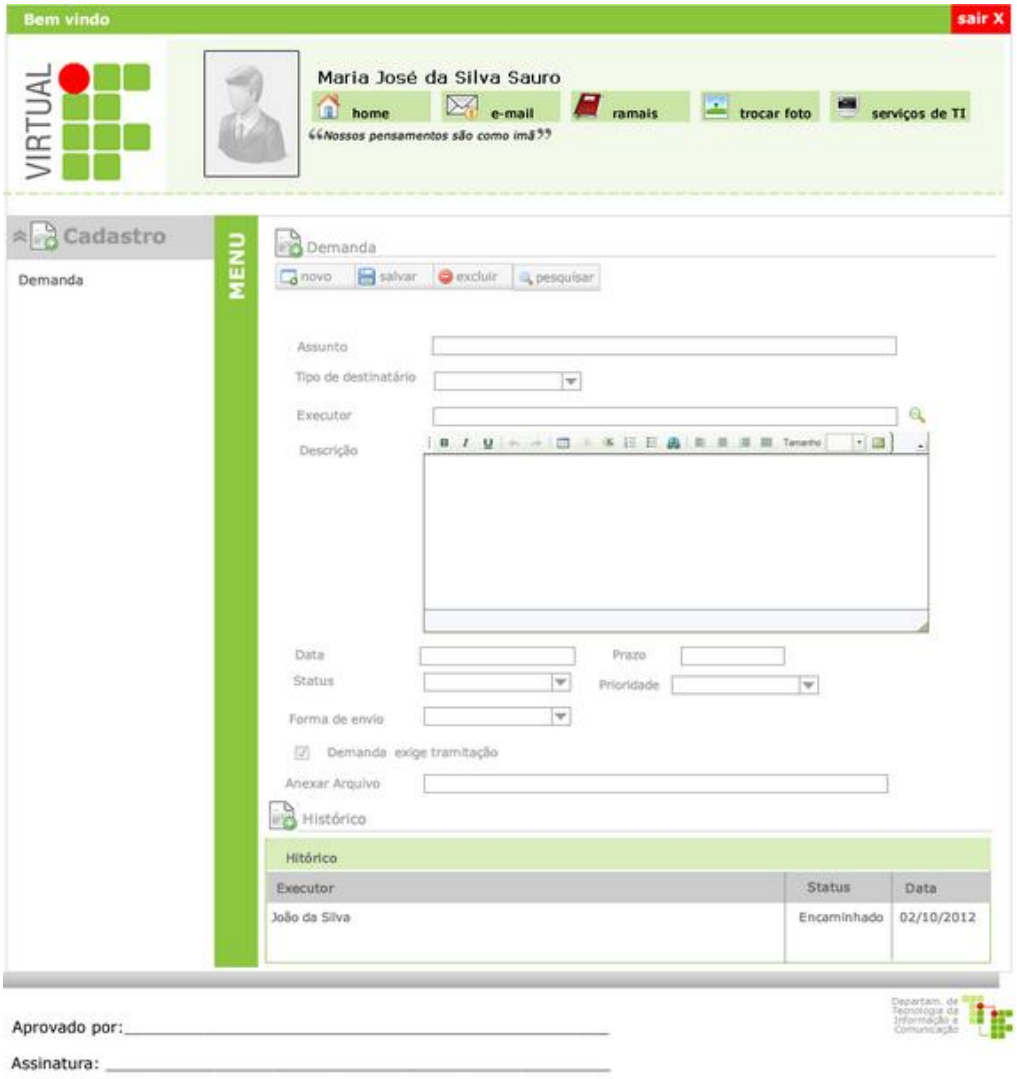

Figura 20 - Um protótipo de tela para a funcionalidade "Gerencia de Demandas".

Para aprovação do design de uma tela, é necessário ter a liberação e assinatura do Patrocinador ou um *stakeholder* designado pelo próprio Patrocinador (canto inferior esquerdo da Figura 20).

Após a realização destas atividades, cabe ao Coordenador elaborar o plano para a próxima reunião. Novamente, os artefatos e as informações resultantes das iterações são cadastrados no sistema de gerenciamento – *Redmine*. A seguir o diagrama de caso de uso da funcionalidade "Gerenciar Demandas" é apresentado.

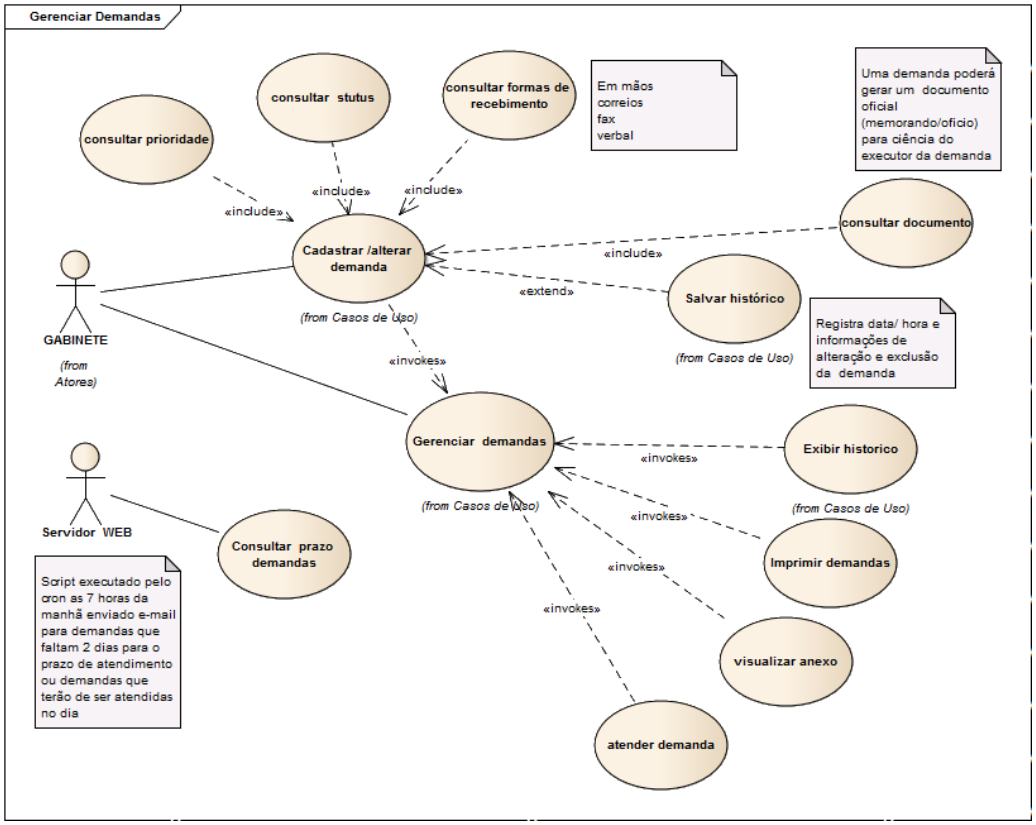

Figura 21 - O Diagrama de Caso de Uso da Funcionalidade "Gerenciar Demandas".

Passo 4: Para apresentar o trabalho produzido, foram realizadas outras iterações a fim de aprovar os requisitos referentes de cada funcionalidade. Nesta reunião, as telas e os Casos de Uso foram ajustados, de acordo com entendimento dos envolvidos. Quando existem comum acordo e aceitação dos membros do projeto, dois novos artefatos são produzidos: o *Product Backlog* e o Termo de Aceite (Figura 16). É importante destacar que o *Product Backlog* é construído e atualizado a cada iteração. Entretanto, em sua versão final, este documento apresenta uma estimativa de esforço como apoio para a equipe de desenvolvimento.

Passo 5: Agora, na fase de Construção, a equipe de desenvolvedores utilizam os requisitos (e artefatos) registrado no ambiente *Redmine*, para elaboração do Dicionário de Dados, o Diagrama de Sequencia e o Diagrama de Classes (Figura 17). Em seguida, a equipe criou os artefatos de codificações e definições da estrutura de banco necessárias para atendimento das funcionalidades. A seguir, a Figura 22 e a Figura 23 apresentam os Diagramas de Sequencia e de Classes da funcionalidade trabalhada.

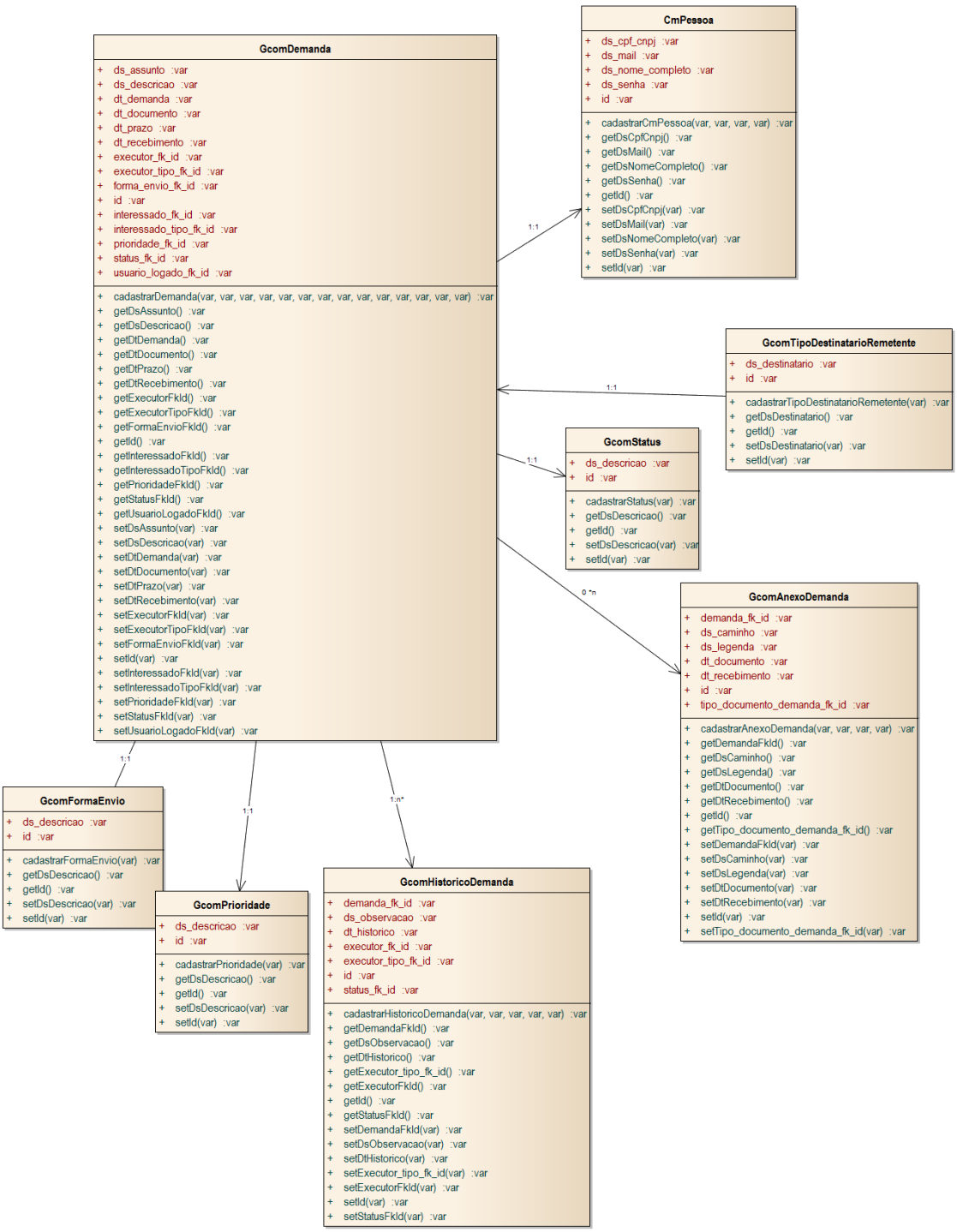

Figura 22 - Diagrama de Classe da Funcionalidade "Gerenciar Demandas".

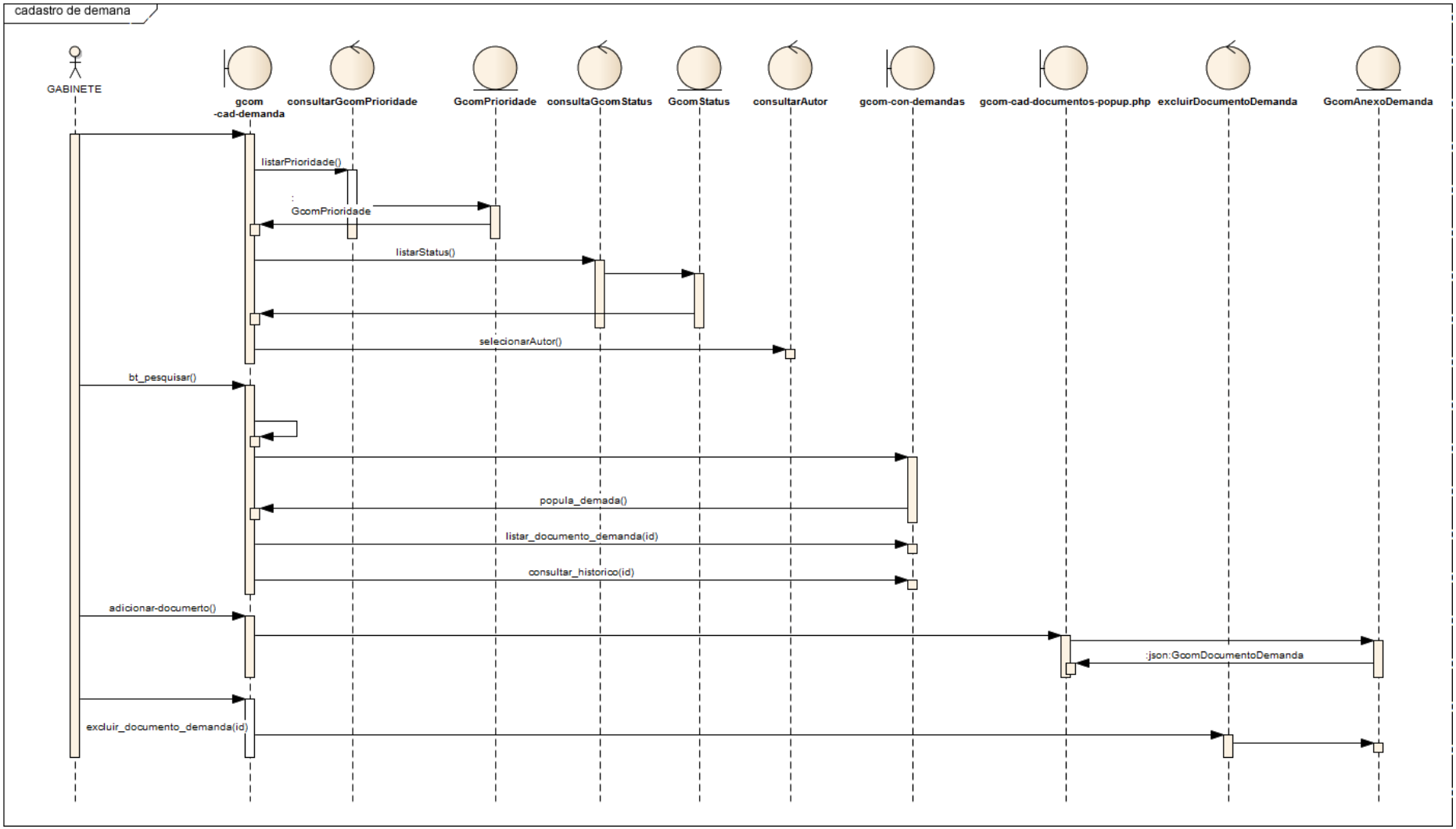

Figura 23 - Diagrama de Sequencia da funcionalidade "Cadastro de Demanda".

Conforme previsto nesta fase, a mesma equipe de desenvolvedores realiza testes de exaustão para cada funcionalidade. Com os primeiros testes realizados, a funcionalidade foi liberada no servidor de treinamento para os *stakeholders* do projeto. Após outra série de testes e validação, agora por parte dos *stakeholders,* o Termo Aceite Atual foi formalmente registrado (Figura 17).

Passo 6: Finalmente, na fase de Transição (Figura 18), a equipe de integração apresenta a funcionalidade para aquelas equipes responsáveis por outros componentes do GRP e registra as possibilidades de integração com outros módulos do sistema. O artefato esperado deste passo é o Relatório de Integração.

Passo 7: Ainda na fase de Transição, a equipe de desenvolvimento implanta a funcionalidade do módulo no servidor de produção com objetivo de entregar o componente funcional em operação. Para tanto, primeiro elabora-se a parte dos manuais referentes à funcionalidade aprovada. Esta etapa é importante para treinar os usuários em operar de forma eficaz e eficiente a funcionalidade liberada. O último artefato do ciclo é o Termo de Aceite Final, devidamente assinado pelo *stakeholder* envolvido, ou pelo Patrocinador do Projeto.

6.3. Análise da Metodologia por meio de questionários

Esta seção apresenta os resultados obtidos pela análise da aplicação da metodologia proposta nas equipes de TI do IFTM. Foram disponibilizados três projetos (Gerenciador de Gabinete, Almoxarifado e Protocolo Acadêmico) para serem trabalhados nesta pesquisa e estes foram divididos em três equipes diferentes, conforme relatado no Capítulo 5.

Para os três projetos, a MDS-GRP foi aplicada com objetivo de identificar os impactos da mesma no desenvolvimento de sistemas. É importante lembrar que a fábrica de software não possuía nenhuma metodologia adotada anteriormente. Assim, o grande desafio desta proposta foi adequar metodologia conhecidas para a realidade das equipes de TI envolvidas.

Para realizar a comparação entre as equipes, foi aplicado junto aos coordenadores de sistemas e seus desenvolvedores, um questionário. Neste questionário foram abordados aspectos relevantes para evidenciar a importância da adoção da metodologia. Para tanto, os avaliadores foram classificados como “Coordenador de

Sistemas” ou “Desenvolvedor”, e responderam dez questões de múltipla escolha sendo as opções: “Muito Satisfeito”; “Satisfeito” e “Insatisfeito”. Ainda assim, para cada questão, o questionário permitiu ao avaliador descrever livremente opiniões e/ou possíveis melhorias para adequações da versão atual da MDS-GRP. Para explanação deste campo, observou-se que o avaliador levou em consideração apenas quando optou em marcar “Satisfeito” ou “Insatisfeito”.

A seguir, são apresentados os gráficos gerados pelos questionários para demonstrar os resultados obtidos na análise das entrevistas realizadas.

Pergunta 1: ***A comunicação entre a equipe de desenvolvimento por meio da adoção da MDS-GRP obteve resultados melhores que anteriormente?***

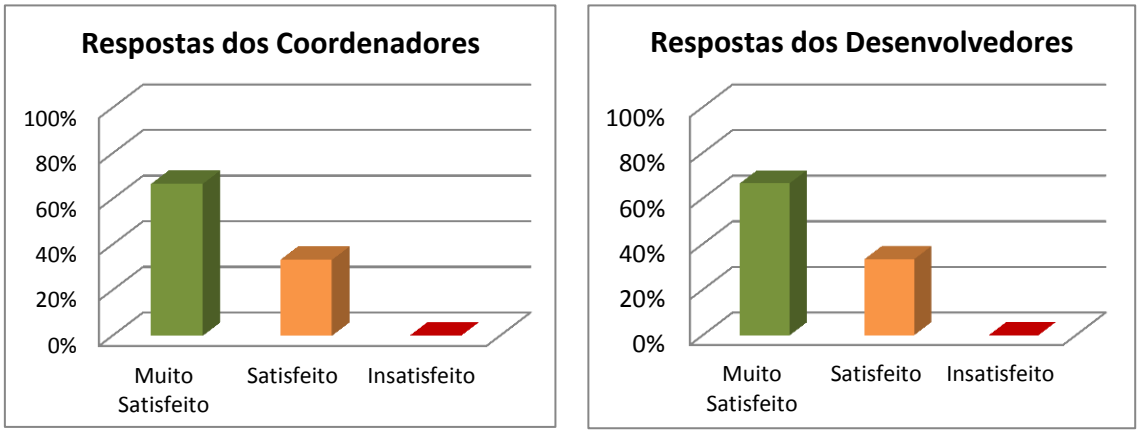

Figura 24 - Comunicação entre as equipes de desenvolvimento após MDS.

Para todos os avaliadores, a adoção da MDS-GRP resultou em ganhos significativos na comunicação entre as equipes. Entretanto, na opinião de um dos avaliadores, houve falta de comprometimento por parte das equipes e também o Gerente de Projetos demorou em tomar uma posição mais firme quanto sua adoção. Neste sentido, alguns resultados poderiam estar em melhores escalas.

A seguir relato de um outro avaliador que optou por estar Muito Satisfeito mas utilizou o espaço de explanação para dar seu parecer:

> *“A padronização permite um trabalho sincronizado e profissional com facilidades de manutenção e entendimento, tornando a capacitação das equipes equiparadas possuem o mesmo nível de entendimento de todos os módulos desenvolvidos”.*

Pergunta 2: ***Os artefatos produzidos na MDS-GRP são suficientes para o entendimento dos módulos produzidos na sua equipe?***

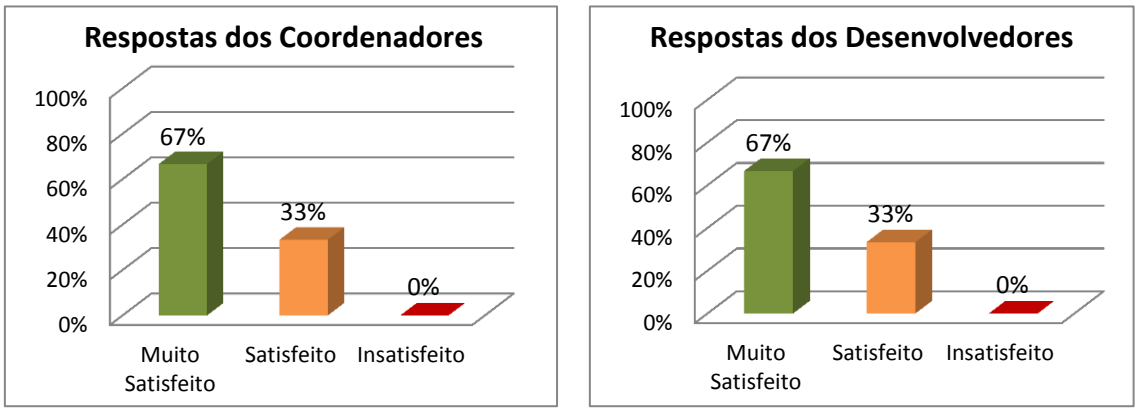

Figura 25 - Análise dos artefatos produzidos na MDS.

Um dos avaliadores relatou que os artefatos gerados na MDS-GRP conseguem exemplificar a visão da regra de negócio, da codificação das três camadas e da estrutura do banco de dados. Além disso, outro avaliador relatou a necessidade da implantação de padrões de projetos.

Pergunta 3: ***As iterações com os stakeholders apresentou melhores resultados em termos de registro dos requisitos levantados?***

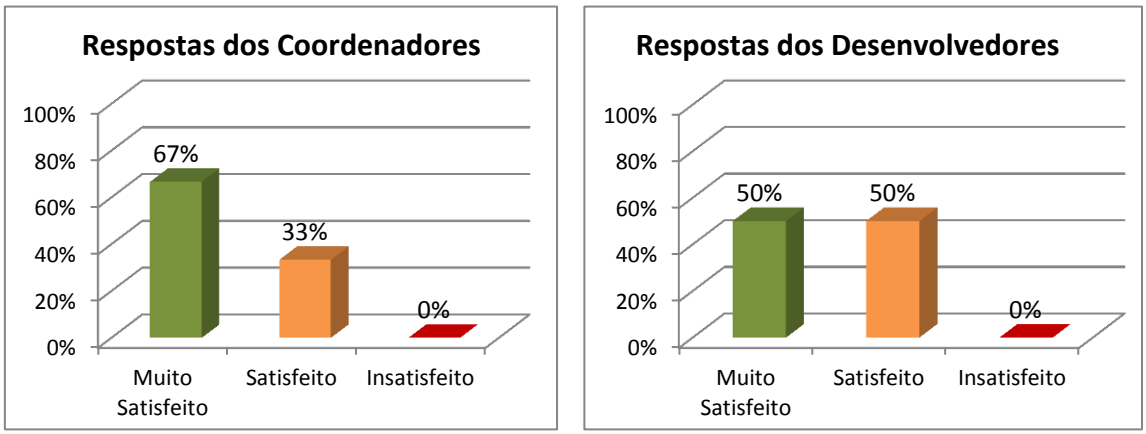

Figura 26 - Análise das interações com os *stakeholders*.

Os gráficos acima revelam as diferentes formas de interação que ocorrem entre as equipes envolvidas com os *stakeholders*. Claramente, é preciso desenvolver melhores formas de comunicação entre os *stakeholders* e os desenvolvedores. Particularmente, dois fatos interessantes foram identificados:

1) Além da falta de documentações do processo de negócio (mapa dos processos, fluxo dos processos e especificação dos processos), foram detectados problemas relacionados ao perfil do *stakeholder* em não conhecer o seu próprio processo dificultando os trabalhos da equipe de requisitos. Analisando este fato, foi considerada a hipótese de incluir o Diagrama de Atividades como mais um artefato, para facilitar o processo de comunicação entre os desenvolvedores e os *stakeholder*.

2) Falta de registro por parte da equipe de requisitos. Este falto gerou conflitos entre os desenvolvedores e os *stakeholders* sobre a existência ou não de um determinado requisito.

Pergunta **4*: As informações contidas nos artefatos da fase de Concepção apresentaram de forma satisfatória a visão geral do escopo do projeto, bem como as possíveis funcionalidades iniciais?***

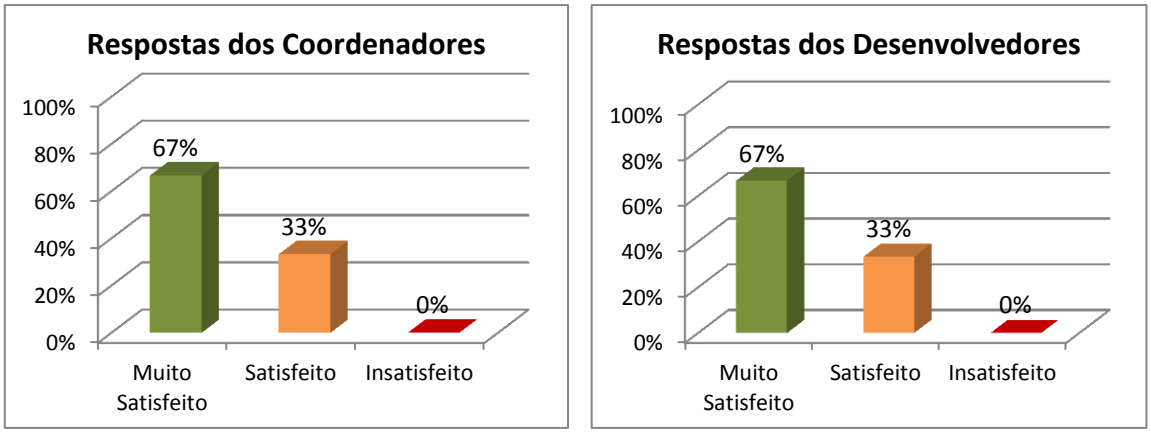

Figura 27 - Análise de melhores resultados na visão geral do escopo do projeto e em possíveis funcionalidades.

Para esta questão algumas abordagens foram importantes e devem ser levadas em conta para o aprimoramento da MDS-GRP. Abaixo, relato de um dos avaliadores na seção de explanação:

> *"A fase de Concepção possui a maior quantidade de pessoas envolvidas no projeto, as quais podem não estar preparadas para um levantamento de requisito adequado. Como sugestão seria interessante a elaboração de uma capacitação para os patrocinadores do sistema, para entender conceitos, tais como: projeto, processos, regra do negócio e auxiliar na seleção de responsáveis pelos requisitos e segmentação das funcionalidades".*

Pergunta 5: ***Você concorda que as telas prototipadas e devidamente aceitas por parte dos*** *stakeholders* ***trouxeram maior segurança para implementação?***

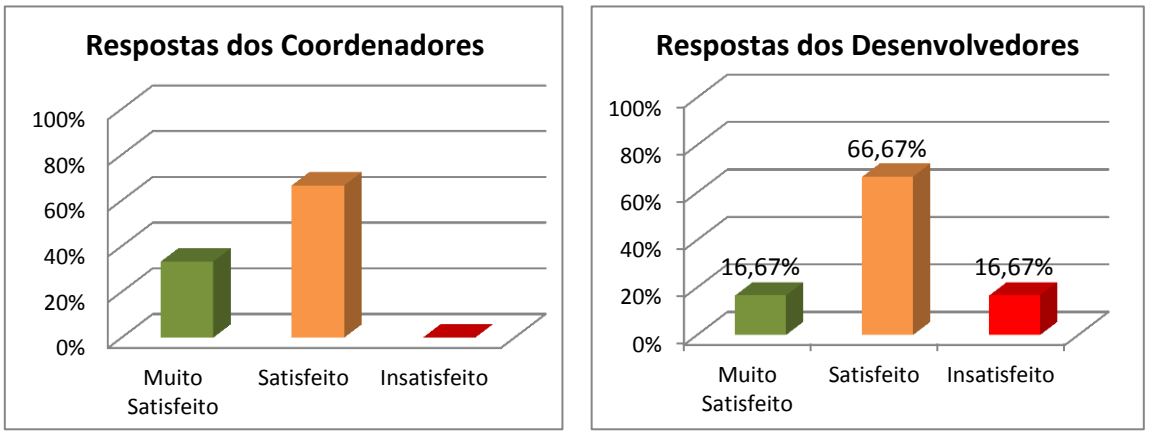

Figura 28 - Análise do uso de protótipos de telas.

Em análise das explanações realizadas pelos avaliadores e dos resultados apresentados pelos gráficos, as equipes de requisitos e de desenvolvimento encontraram dificuldades em conceber as telas prototipadas aos *stakeholders*. Tais dificuldades foram, principalmente, quanto às modificações constantes realizadas pelos últimos e muitas das vezes sem critérios de comum acordo entre os mesmos. Outro aspecto relatado pelos avaliadores, são as mudanças constantes das telas após estarem implementadas. Estas demandas surgiram após uma interação maior com as telas já prontas, na fase de Construção.

O que se pode observar é que o mecanismo de funcionamento das telas não ficou inteiramente completo na visão dos *stakeholders*. Em destaque, o percentual apresentado de 16,67% de Insatisfeito refere-se a um avaliador que relatou que concorda que as telas prototipadas traduzem visualmente e funcionalmente o que o *stakeholder* deseja. Porém, nos projetos por ele trabalhados, este recurso não foi amplamente utilizado. Fato este que justificou sua resposta.

Pergunta 6: ***Os três diagramas da fase de Construção são suficientes para o entendimento da equipe de desenvolvimento na implementação da funcionalidade levantada?***

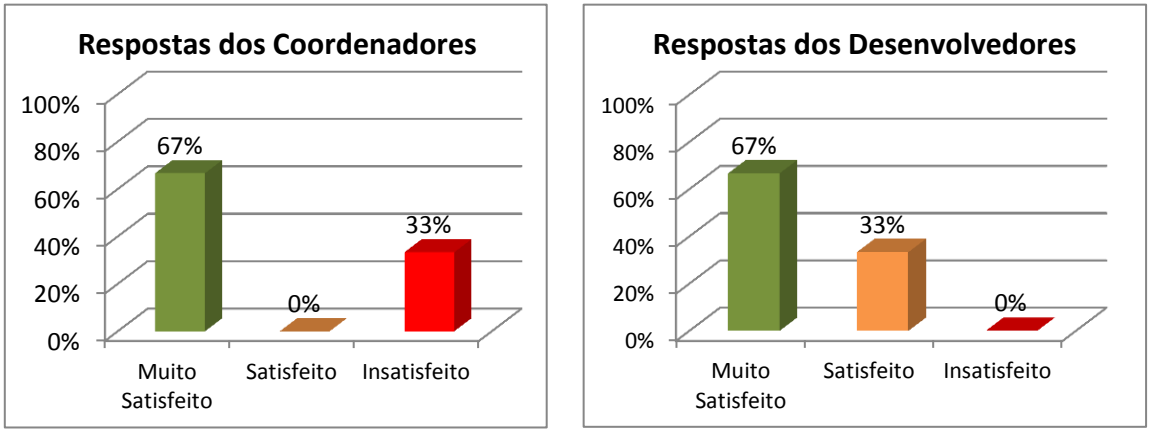

Figura 29 - Análise do entendimento da equipe de desenvolvimento por meio dos artefatos produzidos.

Neste critério, um dos avaliadores registrou no espaço de explanação o seguinte texto:

> *"Na fase de Elaboração ao fazer o PB (Product Backlog) tem participação do cliente, portanto acho que a existência do DIAGRAMA DE ATIVIDADES o cliente terá uma melhor visão do processo de desenvolvimento, pois o mesmo tem como objetivo mostrar o fluxo das atividades e suas dependências".*

Além disso, um dos avaliadores com perfil de Coordenador respondeu estar Insatisfeito, o que representou os 33% da porcentagem. Isto porque ele julga a necessidade de outros diagramas de notação UML, como por exemplo, o Diagrama de Atividades. Assim, este questionamento apontou a necessidade do Diagrama de Atividades para auxiliar no processo de desenvolvimento das funcionalidades trabalhadas.

Pergunta 7: ***Os testes de unidade funcional são suficientes para liberação dos componentes dos módulos em construção?***

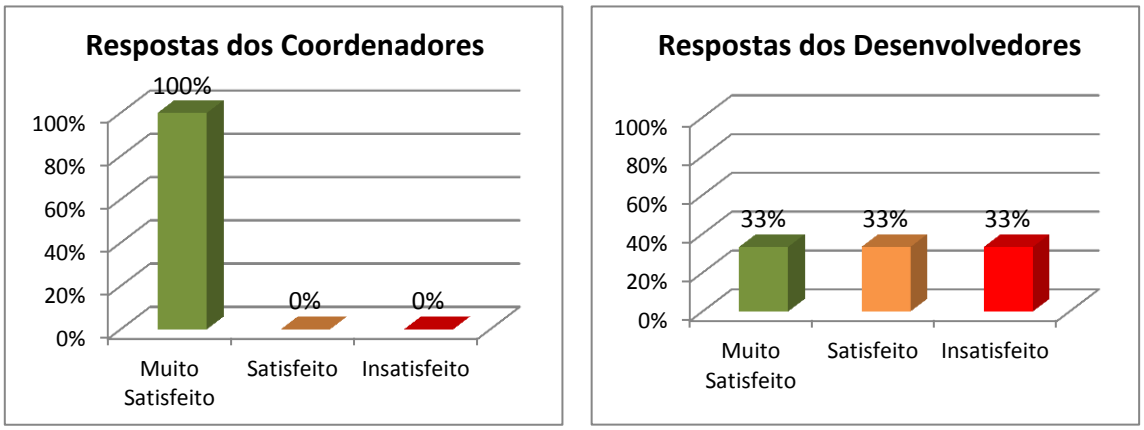

Figura 30 - Análise dos testes de unidade funcional.

Apesar da totalidade dos coordenadores julgarem suficientes os testes funcionais para liberação dos módulos, relatos importantes foram abordados pelos desenvolvedores que sugeriram que os testes também fossem realizados por equipes diferentes. Por exemplo, a equipe EQ01 realizaria os testes das unidades funcionais da EQ02 e vice-versa.

Outro ponto interessante abordado por um dos avaliadores é o envolvimento das equipes de TIs dos campus na realização dos testes, antes das funcionalidades serem liberadas para produção. Desta forma, a fase de testes envolveria um número maior de testadores. Existe aqui um sentimento de que esta conduta produzirá uma melhor segurança quanto à realização do suporte em primeiro nível aos usuários da funcionalidade. Estas avaliações refletiram a porcentagem de 33% de usuários satisfeitos.

Além disso, um conjunto de avaliadores detectou a necessidade de um documento que oriente a condução dos testes. Tal fato resultou numa quantidade de 33% de Insatisfeitos.

Pergunta 8: ***O tempo gasto na confecção dos artefatos impactou significativamente no tempo usado para a construção dos módulos***

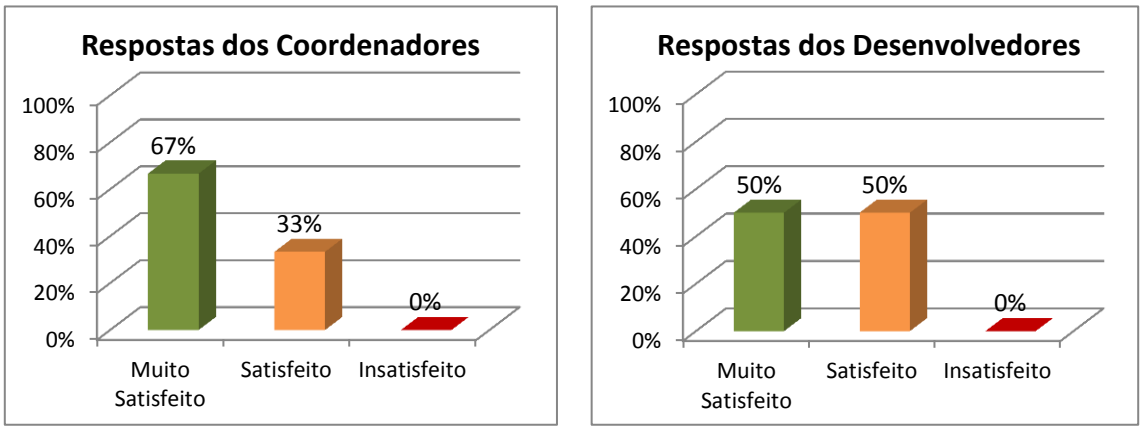

Figura 31 - Análise do tempo gasto entre a relação "artefatos x codificação".

Nesta questão foi detectada a necessidade de capacitação para alguns membros das equipes de requisitos. Este fato refletiu numa quantidade menor (33%) de Coordenadores no quesito Satisfeito, quando comparado com os Desenvolvedores (50%).

Além disso, é importante destacar que um grupo de Satisfeitos da classe de Desenvolvedores sinalizou que tiveram dificuldades na confecção dos artefatos, por falta de conhecimentos do conjunto de notações da linguagem de modelagem (UML). Neste contexto, a busca pelo conhecimento de tais notações e o aumento desta atividade (geração de artefatos) que anteriormente não eram realizados, teve como consequência um maior tempo na construção dos módulos. Finalmente, a porcentagem apresentada no perfil de Coordenador de 33% como Satisfeito sinalizou um cenário de necessidade de força trabalho para realizar todas as fases da metodologia, inclusive a confecção dos artefatos.

Pergunta 9: ***A adoção da metodologia ajudou na melhoria da integração entre as equipes e no desenvolvimento do sistema?***

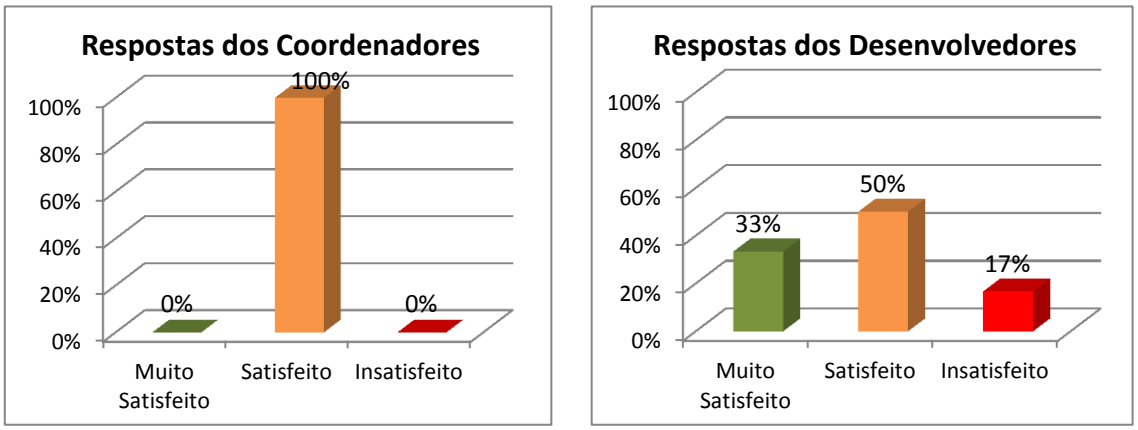

Figura 32 - Análise dos resultados obtidos na integração das equipes do projeto.

Pontos importantes foram abordados por ambos os perfis (Coordenadores e Desenvolvedores). Em análise ao gráfico do perfil de Coordenador, 100% responderam que estão satisfeitos com a integração. Porém, com opiniões e entendimento divergentes. Este conflito é justificado pelos seguintes itens:

1) A integração entre as equipes permitiu a reutilização de códigos e troca de experiências. Porém, ela necessita de cuidados no desenvolvimento de módulos que demandem implementações de forte integração, sendo necessário a aproximação destas para desenvolver a funcionalidade em questão.

2) Necessidades de alterações das tabelas comuns, afetando a dinâmica dos trabalhos da equipe.

3) A integração independe da adoção da metodologia. Este autor, particularmente, discorda deste fato, pois por meio dos processos definidos na metodologia é possível atingir maiores integrações entre as equipes para desenvolvimento do projeto proposto (GRP-IFTM).

O outro gráfico representado pelo perfil de desenvolvedor demonstrou que 50 % estão satisfeitos e 17% Insatisfeitos. Para estes Insatisfeitos sua angustia é a falta de critérios de programação, deixando aberta a forma com que cada equipe desenvolve seus módulos. Aqueles que responderam estar satisfeitos focaram no fato de disciplinar as fases e os fluxos da metodologia. Principalmente, na realização das reuniões sistemáticas diárias e principalmente, de início da semana. Para estes também seria

necessária a confecção de atas para registrar as discussões e socializar com seus desenvolvedores.

Pergunta 10: ***A adoção da MDS-GRP apresentou maior controle nas atividades rotineiras?***

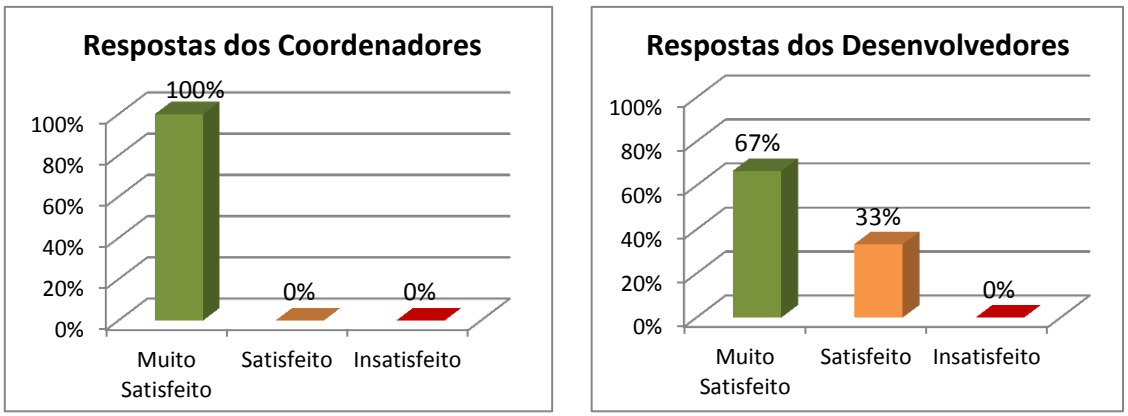

Figura 33 - Análise dos resultados quanto ao controle das atividades.

Os 33% dos desenvolvedores que classificaram como satisfeitos declararam que determinadas atividades não foram executadas conforme proposto, devendo haver maior comprometimento dos envolvidos nos projetos. Entretanto, para a maioria a MDS-GRP proporcionou ganhos nos aspectos organizacionais, melhorando o fluxo e o controle das atividades.

6.4 Considerações Finais

Este capítulo apresentou a avaliação da aplicação da MDS-GRP no estudo de caso proposto. Os gráficos apresentados neste capítulo proporcionaram visões em vários aspectos que devem ser trabalhados para obtenção de melhores resultados qualitativos na geração de artefatos textuais, gráficos e códigos-fontes. Os Coordenadores de Sistemas e os Desenvolvedores demonstraram pontos de suma importância que deverão ser discutidos entre as equipes de TIs para realizações de possíveis adequações.

Em geral, pode-se concluir que a metodologia proposta proporcionou um melhor mecanismo de desenvolvimento e documentação no cotidiano das equipes de TI envolvidos. O próximo capítulo apresenta as conclusões finais e sugestões para trabalhos futuros.

7. Conclusões e Trabalhos Futuros

7.1 Introdução

Esta pesquisa apresentou como um resultado uma integração de diferentes metodologias (processo unificado e metodologias ágeis) na busca por melhores práticas de desenvolvimento de software. Neste capítulo destacam-se os principais pontos estudados nesta dissertação. Além disso, serão apresentadas sugestões para possíveis trabalhos futuros proveniente desta pesquisa e por fim as considerações finais de sua contribuição.

7.2 Conclusões

Em um levantamento bibliográfico durante a pesquisa, constatou-se a existências de trabalhos correlacionados que adaptaram metodologias de desenvolvimento para aplicações em projetos de software. Porém, a maioria destas não considerava uma realidade com características inerentes de repartições públicas.

Diante disso, este trabalho apresentou uma proposta em adequar o Processo Unificado, apoiado por boas práticas de metodologias ágeis, para o desenvolvimento de software, neste contexto. Para tanto, foi customizado o Processo Unificado de forma a reduzir o número de fases e artefatos. Tal customização teve como objetivo principal a obtenção de melhores critérios no gerenciamento do processo de desenvolvimento de software em fábricas com características inerentes às equipes do estudo de caso proposto.

Os estudos conduzidos nesta pesquisa resultaram em uma metodologia que foi definida como MDS-GRP. As quatro fases trabalhadas no Processo Unificado (Concepção, Elaboração, Construção e Transição), também foram utilizadas para este trabalho. Porém foram customizadas cada etapa, definindo seus processos com seus respectivos fluxos, os responsáveis pela execução destes e por fim os artefatos a serem gerados.

Um estudo de caso com realidade de fábricas de software de repartições públicas foi utilizado como forma de avaliar os resultados inerentes a sua adoção. Com isso, a fábrica em questão foi do Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro.

Antes da aplicação da proposta foi apresentado o projeto GRP-IFTM que representa um sistema corporativo governamental. Este projeto já se encontrava em andamento no momento desta pesquisa. Porém sem a utilização de nenhuma metodologia formalmente aprovada. A apresentação do GRP-IFTM teve como intuito estabelecer o entendimento das principais características dos módulos a serem desenvolvidos. Neste sentido, aplicou-se a metodologia em três módulos do projeto GRP-IFTM (Gerenciador de Gabinete, Almoxarifado e Protocolo Acadêmico) com aspectos próximos nos quesitos de complexidade e tamanho, e para cada um, uma equipe com colaboradores diferentes. Porém para contextualizar tal aplicação foram apresentados passos apenas de um dos projetos para demonstrar os seus diversos estágios.

A adoção da metodologia teve como resultado, melhores práticas na gestão de projetos de software e espera-se assim contribuir na perspectiva da integração de outros módulos hierárquicos do Projeto GRP-IFTM.

É importante ressaltar que através das análises dos resultados dos questionários aplicados aos Coordenadores de Sistemas e Desenvolvedores, puderam demonstrar alguns ajustes necessários para o aprimoramento deste trabalho.

Como conclusões da aplicação da metodologia, por meio do estudo de caso apresentado, alguns pontos em destaque são abordados e conclui-se que:

- A aplicação da MDS-GRP possibilitou melhores resultados nos aspectos de comunicação entre as equipes de desenvolvimento.
- Uma revisão quanto à prática de prototipações de interfaces para apreciação dos *stakeholders* se faz necessária.
- Para obtenção de melhores resultados nas atividades relacionadas aos levantamentos de requisitos junto aos clientes, é pertinente considerar sugestões oriundas dos avaliadores quanto ao artefato Diagrama de Atividades, sendo fundamentado para melhor aderência do processo. Entretanto, foram considerados suficientes os demais artefatos trabalhados para compreensão do módulo.
- A necessidade de capacitação de alguns membros das equipes para confecção dos diagramas UML.
- Deve haver maiores detalhes para execução do processo de testes, antes da liberação do produto final.

7.3 Trabalhos futuros

Diante de todos os aspectos demonstrados neste trabalho, alguns pontos foram detectados durante o processo de aplicação e demandam maiores atenções. Portanto, devem ser relacionados para possibilidades de trabalhos futuros:

- Avaliação da aplicação do framework MPS.BR (Melhoria de Processos do Software Brasileiro) para aprimorar a aderência da MDS-GRP. Tal aplicação destina-se a melhorar a capacidade de desenvolvimento do GRP-IFTM.
- Aplicação da metodologia com outros módulos a serem desenvolvidos dentro do escopo do GRP-IFTM, a fim de avaliar o comportamento em projetos com diferentes naturezas de desenvolvimento.

- Outro aspecto é quanto aos padrões de projetos que também foram sinalizados nas avaliações desta pesquisa e demanda um estudo mais profundo para aplicação conjuntamente com esta MDS.

7.4 Considerações finais

Este trabalho apresentou transformações na fábrica de software do estudo de caso, pois atualmente a proposta deixou de ser sugestiva e se transformou em realidade. Ela está sendo utilizada como metodologia padrão para construção de todos os módulos do projeto GRP-IFTM. Percebe-se que este modelo poderá ser utilizado por organizações de mesmo porte. Portanto, este trabalho buscou apresentar ao mundo cientifico as vantagens de integrar métodos tradicionais e métodos ágeis para alcançar melhores resultados no desenvolvimento de software.

8. Referências

ALVES, N. **Integração de princípios de desenvolvimento ágil de software ao rup:** um estudo empírico. 2011. 137 f. Tese (Doutorado) - Universidade Federal de Uberlândia, Uberlândia, 2011. Cap. 4.

ALVES, N., CARVALHO, W., CARDOSO, A., LAMOUNIER JUNIOR, E.. **Um estudo de caso industrial sobre integração de práticas ágeis no RUP**. Revista Ciência e Tecnologia, América do Norte, 14, mar. 2012. Disponível em:http://revistavirtual.unisal.br:81/seer/ojs-2.2.3/index.php/123/article/view/169. Acesso em: 21 Jul. 2011.

ABRAHAMSSON, P. *et al*.. **New Directions on Agile Methods: A Comparative Analysis**. International Conference on Software Engineering. Oregon: IEEE Computer Society. 2003.

BECK, K. *et al*. **Manifesto for Agile Software Development**, 2001. Disponivel em: <http://www.agilemanifesto.org>. Acesso em: 21 Agosto de 2012.

BECK, K. **Extreme Programming Explained: Embrace Change**. Reading: Addison-Wesley, 2001.

BERTOLLO, Gleidson; FALBO, Ricardo de Almeida. **Apoio Automatizado À Definição de Processos em Níveis**. In: II Simpósio Brasileiro de Qualidade de Software, 2003, Fortaleza, Ceará. Anais do II Simpósio Brasileiro de Qualidade de Software, 2003. p. 77-91.

BOEHM, B.; TURNER, R. **Balancing Agility and Discipline:** Evaluating and Integrating Agile and Plan-Driven Methods. International Conference on Software Engineering. Washington, DC, USA: IEEE Computer Society. 2004a.

BOOCH, Grady; RUMBAUCH, James; JACOBSON, Ivar. **UML:** Guia do Usuário. 2ª ed. São Paulo: Elsevier, 2006.

COHEN, D.; LINDVALL, M.; COSTA, P. **An Introduction to Agile Methods**. Amsterdam: Elsevier, v. 62, 2004.

DYBÅ, T.; DINGSøYR, T.. **Empirical studies of agile Software development: A systematic review**. Information and Software Technology, v. 50, 2008.

DYBÅ, T.; DINGSøYR, T.. **What Do We Know about Agile Software Development?** IEEE Software, 2009.

FERNANDES, A. A.; TEIXEIRA, D. S.. **Fábrica de Software: Implantação e gestão de Operações.** São Paulo: Atlas, 2004.

FERRER, Florencia; SANTOS, Paula; SOLA, Pier Carlo. Governo digital: origem do conceito e modelo para discussão. In: FERRER, Florencia; SANTOS, Paula (Org.). E-Government: o governo eletrônico no Brasil. São Paulo: Saraiva, 2004. p.114-119.

FUGGETTA, Alfonso. **Software Process: A Roadmap**. In: 22nd International Conference on on Software Engineering (ICSE), Proceedings of the Conference on the Future of Software Engineering, New York: ACM Press, 2000. pp. 25-34.

GREER, D.; CONRADI, R.. **Software Project Initiation and Planning: An Empirical Study.** Institution Of Engineering And Technology, Northern Ireland, UK, v. 5, n. 3, p.356-368, 01 out. 2009.

ISO/IEC 12207, 1995**, Information Techonology – Software Life-Cycle Processes**

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **Unified Software Development Process**. Reading: Addison-Wesley, 1999.

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

LEFFINGWELL, D**. Scaling Software Agility**. Reading: Addison Wesley, 2006.

MACHADO, Luis Filipe Dionisio Cavalcanti. **MODELO PARA DEFINIÇÃO DE PROCESSOS DE SOFTWARE NA ESTAÇÃO TABA.** 2000. 123 f. Dissertação (Mestrado) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2000. Cap. 1.

MAIA, Anderson Baia; FREITAS, Ana Vitoria Piaggio de; NUNES, Daltro José. **Um Modelo para Auxiliar a Adaptação de Processos de Software**. In: Anais do IV

CONGRESSO BRASILEIRO DE COMPUTAÇÃO, Itajaí - Santa Catarina: Univali, 2004. p. 155 - 160.

MARTINS, José Carlos Cordeiro. **Gerenciando Projetos de Desenvolvimento de Software com PMI, RUP e UML.** 3ª ed. Rio de Janeiro: Brasport, 2006.

MATTIOLI, F.E.R.; LAMOUNIER Jr., E. A.; CARDOSO, A.; Alves, N.M.M; **Uma proposta para o desenvolvimento ágil de ambientes virtuais**. In: Anais do VI Workshop de Realidade Virtual e Aumentada – WRVA'2009, Santos, SP, 2009.

MENDES, Juliana Veiga; ESCRIVÃO FILHO, Edmundo. **Sistemas integrados de gestão ERP em pequenas empresas: um confronto entre o referencial teórico e a prática empresarial**. Gestão & Produção, São Carlos: Ufscar, v.9, n.3, p.277-296, dez. 2002.

NORD, R.; TOMAYKO, J. Software Architecture-Centric Methods and Agile Development. **IEEE Software**, 2006.

PAULA FILHO, Wilson de Pádua. **Engenharia de Software:** Fundamentos, Métodos e Padrões. 3ª ed. Rio de Janeiro: Ltc, 2009.

PFLEEGER, Shari Lawrence. **Engenharia de Software:** Teoria e Prática. 2ª ed. São Paulo: Pearson, 2004.

PMBOK. **Um guia do conjunto de conhecimentos em gerenciamento de projetos**. 4.ed.Project Management Institute, Inc. 2008

PRESSMAN, Roger S.. **Software Engineering:** A Practitioner's Approach. 6th New York: Mcgraw-hill, 2006.

PRESSMAN, Roger S.. **Engenharia de Software:** Uma Abordagem Profissional. 7ª ed. Porto Alegre: Amgh, 2011.

OSTERWEIL, L.. **Software Process Are Software Too**. In: 9th International Conference on Software Engineering (ICSE), Monterey, Estados Unidos, 1987, p. 2-13.

RAMSIN, R.; PAIGE, R. F. **Process-Centered Review of Object Oriented Software Development Methodologies**. ACM Computing Surveys, v. 40, n. 1, 2008.

ROCHA, Rodrigo. *et al*. **Uma Experiência na Adaptação do RUP em Pequenas Equipes de Desenvolvimento Distribuído**. In: II WORKSHOP DE DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE - WDDS, 2008, Campinas-SP. Anais do II Workshop de Desenvolvimento Distribuído de Software - WDDS. Campinas-SP: Universidade Estadual de Maringá (UEM), 2008. p.81-90

ROCHA, Thayssa Águila da; OLIVEIRA, Sandro Ronaldo Bezerra; VASCONCELOS, Alexandre Marcos Lins de. **Adequação de Processos para Fábricas de Software**. In: VI SIMPÓSIO INTERNACIONAL DE MELHORIA DE PROCESSOS DE SOFTWARE, 2004, São Paulo-SP. Anais do VI Simpósio Internacional de Melhoria de Processos de Software. São Paulo- SP: SIMPROS, 2004. p.131-142.

ROMBACH, D.; SEELISCH, F. **Formalisms in Software Engineering: Myths Versus Empirical Facts**. Central and East European Conference on Software Engineering Techniques. Poznan: Springer. 2007. p. 13-25.

SCHWABER, K.; **Agile Project Management with Scrum**: Microsoft Press, 2004

SCOTT, Kendall. **O Processo Unificado Explicado.** São Paulo: Bookman, 2003.

SHACH, Stephen R.. **Engenharia de Software:** Os Paradigmas Clássico Orientado a Objetos. 7ª ed. São Paulo: Mcgraw-hill, 2009.

SOMMERVILLE, Ian. **Engenharia de Software.** 9ª ed. São Paulo: Pearson, 2011.

SOUZA, Francisco Flavio de ; BRAGA, R. T. V. . **Um Método de Desenvolvimento de Sistemas de Grande Porte Baseado no Processo RUP**. In: 1º Simpósio Brasileiro de Sistemas de Informação, 2004, Porto Alegre - RS. Anais do 1º SBSI, 2004. p. 31-38.

TELES, Vinícius Manhães. **Extreme Programming.** São Paulo: Novatec, 2004.

ANEXO I – Questionário aplicado

Avaliação da Proposta MDS-GRP	
Avaliador:	Data Avaliação:
( ) Coordenador ( ) Desenvolvedor	

Assinale, por favor, a opção que melhor traduz a sua opinião.

Evidência a adoção da MDS-GRP

1. A comunicação entre a equipe de desenvolvimento por meio da adoção da MDS-GRP obteve resultados melhores que anteriormente?

   ( ) Muito Satisfeito ( ) Satisfeito ( ) Insatisfeito

   Caso sua opção não for 'Muito satisfeito', favor explanar abaixo o motivo (ausência de alguma atividade, etc).

_____
_____
_____
_____
_____
_____

2. Os artefatos produzidos na MDS-GRP são suficientes para o entendimento dos módulos produzidos na sua equipe?

   ( ) Muito Satisfeito ( ) Satisfeito ( ) Insatisfeito

   Caso sua opção não for 'Muito satisfeito', favor explanar abaixo o motivo (ausência de algum artefato, etc.).

_____
_____
_____
_____
_____
_____

3. As iterações com os *stakeholders* apresentou melhores resultados em termos de registro dos requisitos levantados?

   ( ) Muito Satisfeito ( ) Satisfeito ( ) Insatisfeito

   Caso sua opção não for 'Muito satisfeito', favor explanar abaixo o motivo (natureza dos formulários usados na iteração, artefatos resultantes da fase de aprovação – temos de aceite etc.).

_____
_____
_____
_____
_____
_____

4. As informações contidas nos artefatos da fase de Concepção apresentaram de forma satisfatória a visão geral do escopo do projeto, bem como as possíveis funcionalidades iniciais?

   ( ) Muito Satisfeito ( ) Satisfeito ( ) Insatisfeito

   Caso sua opção não for 'Muito satisfeito', favor explanar abaixo o motivo (quais informações estes poderiam ser acrescentadas para o entendimento do escopo inicial; a necessidade de mais alguns artefatos como complemento e quais seriam estes e etc.).

_____
_____
_____
_____
_____

5. Você concorda que as telas prototipadas e devidamente aceitas por parte do *stakeholder* trouxe maior segurança para implementação?

   ( ) Muito Satisfeito ( ) Satisfeito ( ) Insatisfeito

   Caso sua opção não for 'Muito satisfeito', favor explanar abaixo o motivo. (real necessidade das telas prototipadas e assinadas e etc.).

_____
_____
_____
_____
_____

6. O três diagramas da fase de Construção são suficientes para o entendimento da equipe de desenvolvimento na implementação da funcionalidade levantada?

   ( ) Muito Satisfeito ( ) Satisfeito ( ) Insatisfeito

   Caso sua opção não for 'Muito satisfeito', favor explanar abaixo o motivo (ausência ou exclusão de algum artefato e etc.).

_____
_____
_____
_____

7. Os testes de unidade funcional são importantes no impacto para liberação dos componentes dos módulos em construção?

   ( ) Muito Satisfeito ( ) Satisfeito ( ) Insatisfeito

   Caso sua opção não for 'Muito satisfeito', favor explanar abaixo o motivo (algum procedimento poderia contribuir e etc.).

_____
_____
_____
_____

8. O tempo gasto na geração dos artefatos impactou significativamente no tempo usado para a construção dos módulos

( ) Muito Satisfeito ( ) Satisfeito ( ) Insatisfeito

Caso sua opção não for 'Muito satisfeito', favor explanar abaixo o motivo (necessita de métricas que comtemplem o tempo para confecção dos artefatos e etc.).

_____
_____
_____
_____
_____

9. A adoção da metodologia ajudou na melhoria da interação entre as equipes e no desenvolvimento do sistema?

( ) Muito Satisfeito ( ) Satisfeito ( ) Insatisfeito

Caso sua opção não for 'Muito satisfeito', favor explanar abaixo o motivo (reuniões semanais para estabelecer as possíveis integrações e padrões de desenvolvimento). Além disso, agradecemos se for o caso, a gentileza de explicar COMO a adoção da metodologia ajudou nesta melhoria.

_____
_____
_____
_____
_____

10. A adoção da MDS-GRP apresentou maior controle nas atividades rotineiras?

( ) Muito Satisfeito ( ) Satisfeito ( ) Insatisfeito

Caso sua opção não for 'Muito satisfeito', favor explanar abaixo o motivo. (necessidade de rotinas mais definidas etc.).

_____
_____
_____
_____
_____

Comentários/ Observações: