UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA INSTITUCIONALIZAÇÃO DO PROCESSO DE PROJETO E CONSTRUÇÃO DO PRODUTO EM UMA FÁBRICA DE SOFTWARE

MARCUS PAULO DA SILVA MELO

DM 46/2011

UFPA / ICEN / PPGCC
Campus Universitário do Guamá
Belém – Pará - Brasil
2011

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MARCUS PAULO DA SILVA MELO

UMA INSTITUCIONALIZAÇÃO DO PROCESSO DE PROJETO E CONSTRUÇÃO DO PRODUTO EM UMA FÁBRICA DE SOFTWARE

DM 46/2011

UFPA / ICEN / PPGCC
Campus Universitário do Guamá
Belém – Pará - Brasil
2011

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MARCUS PAULO DA SILVA MELO

UMA INSTITUCIONALIZAÇÃO DO PROCESSO DE PROJETO E CONSTRUÇÃO DO PRODUTO EM UMA FÁBRICA DE SOFTWARE

Dissertação submetida à Banca Examinadora do Programa de Pós-Graduação em Ciência da Computação da UFPA para a obtenção do Grau de Mestre em Ciência da Computação do Instituto de Ciências Exatas e Naturais da Universidade Federal do Pará.
Área de Concentração Engenharia de Software.
Orientador Prof. Dr. Sandro Ronaldo Bezerra Oliveira.

UFPA / ICEN / PPGCC
Campus Universitário do Guamá
Belém – Pará - Brasil
2011

Melo, Marcus Paulo da Silva
Uma Institucionalização do Processo de Projeto e Construção do Produto em uma Fábrica de Software / (Marcus Paulo da Silva Melo); orientador, Sandro Ronaldo Bezerra Oliveira. - 2011.
166 f. il. 28 cm
Dissertação (Mestrado) - Universidade Federal do Pará. Instituto de Ciências Exatas e Naturais. Programa de Pós-Graduação em Ciência da Computação. Belém, 2011.

1. Engenharia de Software. I. Oliveira, Sandro Ronaldo Bezerra, orient. II. Universidade Federal do Pará, Instituto de Ciências Exatas e Naturais, Programa de Pós-Graduação em Ciência da Computação. III. Título.
CDD 22. ed. 005.1

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MARCUS PAULO DA SILVA MELO

UMA INSTITUCIONALIZAÇÃO DO PROCESSO DE PROJETO E CONSTRUÇÃO DO PRODUTO EM UMA FÁBRICA DE SOFTWARE

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Pará como requisito para obtenção do título de Mestre em Ciência da Computação, defendida e aprovada em 31/08/2011, pela banca examinadora constituída pelos seguintes membros:

Sandro Ronaldo B. Oliveira

Prof. Dr. Sandro Ronaldo Bezerra Oliveira
Orientador – PPGCC/UFPA

Prof. Dr. Eloi Luiz Favero
Membro Interno – PPGCC/UFPA

Prof. Dr. Cláudio Alex Rocha
Membro Externo – UNAMA/PA

Visto: Sandro Ronaldo B. Oliveira

Prof. Dr. Sandro Ronaldo Bezerra Oliveira
Coordenador do PPGCC/UFPA

Agradecimentos

Primeiro a Deus que meu deu forças e ajudou a concluir este trabalho.

Agradeço aos meus Pais, meu irmão e minhas sobrinhas por todo o apoio e confiança que me deram para alcançar este objetivo.

Agradeço ao meu orientador, prof. Dr. Sandro Ronaldo Bezerra Oliveira, pelo tempo dedicado e todo o apoio dado.

Agradeço especialmente a minha namorada Rosana Kalil, que esteve ao meu lado dando todo o apoio possível para a realização deste trabalho.

Agradeço ao meu amigo e Inácio Gorayeb pela a amizade e a força dada.

A todos aqueles que direta ou indiretamente contribuíram para a realização deste trabalho.

"Embora ninguém possa voltar atrás
e fazer um novo começo,
qualquer um pode começar agora e
fazer um novo fim "

(Chico Xavier)

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Lista de Abreviaturas

BID	*Banco Interamericano de Desenvolvimento*
BPMN	*Business Process Modeling Notation*
CMMI	*Capability Maturity Model Integration*
CVS	*Concurrent Version System*
ERP	*Enterprise Resource Planning*
FINEP	*Financiadora de Projetos*
GQM	*Goal Question Metrics*
ISO/IEC	*The International Organization for Standardization/ The International Electrotechnical Commission*
MCT	Ministério da Ciência e Tecnologia
MPS.BR	Melhoria de Processos do Software Brasileiro
PCP	Processo de Projeto e Construção do Produto
PET	*Programa de Excelência Tecnológica*
SEPG	*Software Engineering Process Group*
SGF	Sistema de Gestão de Fomento
SLA	*Service Level Agreement*
SQL	*Structured Query Language*
SRF	Sistema da Receita Federal
TAG	Um rótulo que é associado a um conjunto de itens de configuração no repositório
UCP	*Use Case Points*
UML	*Unified Modeling Language*

Resumo

Pesquisas apresentam que existem altas taxas de insucessos nos projetos de software, devido a processos de software não definidos corretamente ou que não evoluem com o tempo. Por sua vez, a qualidade de software é uma das áreas da Engenharia de Software que tem como objetivo prover mecanismos para a criação e a avaliação do processo de desenvolvimento, resultando em um produto de software com qualidade para o cliente. A proposta desta dissertação é relatar uma institucionalização do processo de projeto e construção do produto em uma fábrica de software. Este trabalho foi desenvolvido e avaliado em um ambiente real de fábrica de software. Em virtude da grande ocorrência de retrabalhos no desenvolvimento do projeto e dos impactos provocados por este processo, foi observada a necessidade de melhorias na qualidade do processo de projeto e construção do produto. Para realizar a avaliação, com foco em prover as melhorias, foram realizadas duas ações para verificar a situação que o projeto se encontrava. As avaliações realizadas foram a quantitativa, através de medições, e a avaliação qualitativa, através de uma entrevista utilizando como instrumento um questionário. Como resultado, os dados foram coletados e tabulados e, em seguida, analisados como forma de aperfeiçoar e de atender as necessidades de um processo de projeto e construção do produto.

PALAVRAS-CHAVE: Processo de Desenvolvimento de Software, Projeto e Construção do Produto de Software, Modelo de Qualidade, Engenharia de Software.

Abstract

Surveys show that there are high rates of failures in software projects due to software processes not set correctly or that do not evolve with time. In turn, the software quality is one of the areas of Software Engineering that aims to provide mechanisms for the definition and evaluation of the development process, resulting in a quality software product to the customer. The purpose of this work is to report an institutionalization of the product design and construction process in a software factory. This work was developed and tested in a real software factory. Due to the high incidence of rework in the project development and the impacts caused by this process, it was observed the need for improvements in the quality of the product design and construction process. To carry out the evaluation, focusing on providing the improvements, there were two actions for the situation that the project was. The evaluations were quantitatively through measurement, and qualitative evaluation through an interview using a questionnaire as an instrument. As a result, the data were collected and tabulated and then analyzed in order to provide the improvements in the process and meet the needs of the product design and construction process.

KEYWORDS: Software Development Process, Software Product Design and Construction, Quality Models, Software Engineering.

Capítulo 1 **- Introdução**

A Engenharia de software é uma das áreas da Ciência da Computação, a qual tem como um dos objetivos desenvolver um produto de software através de um processo de desenvolvimento bem definido (Koscianski, 2007).

Na década de 70, devido aos problemas relacionados ao desenvolvimento de software, houve a chamada crise do software. Os softwares que eram produzidos, não possuíam uma qualidade aceitável e também não atendiam as necessidades dos clientes. O motivo é que as metodologias utilizadas para o desenvolvimento de um software eram deficitárias em suas técnicas, métodos e ferramentas.

Os resultados desta crise, foram produtos de software entregues fora do prazo determinado, custos estipulados para o desenvolvimento ultrapassados e baixa qualidade dos mesmos (Bartié, 2002).

Segundo Pressman(2006), a qualidade de software é realizada por uma combinação complexa de fatores que variam de acordo com a aplicação em questão e as solicitações realizadas pelos clientes. A qualidade de software está ligada à capacidade do sistema atender as necessidades do usuário.

Desta forma, este trabalho possui embasamento na melhoria de processo de software, focado no processo de Projeto e Construção do Produto (PCP). Como forma de validar este relato, foi identificado o cenário de uma organização, que possuía um processo de desenvolvimento que não satisfazia a realidade da empresa.

Através do estudo realizado na bibliográfica da Engenharia de Software, incluindo modelos de qualidade, foi possível realizar uma avaliação e melhorias no processo de Projeto e Construção do Produto desta organização.

Uma das formas de viabilizar a avaliação do processo de PCP foi através da verificação dos pontos fortes, pontos fracos e oportunidades de melhoria, que contribuem para definir a situação a qual o processo encontrava-se.

Adicionalmente, durante a execução do processo de PCP foi necessário realizar avaliações periódicas, as quais eram compostas por entrevistas com os envolvidos no processo e verificação dos ativos (ferramentas, atividades, procedimentos entre outros) juntamente como produto final gerado. O objetivo da avaliação era conhecer a situação a qual o processo se encontrava. Os resultados evidenciados neste relato são apresentados ao longo deste trabalho.

1.1 Motivações

Diante do cenário relatado na seção anterior, fica evidente que um dos principais objetivos das organizações é criação de um produto de software com qualidade, para tal, encontrou-se uma organização com um cenário propício para a aplicação da metodologia de melhoria no processo de Projeto e Construção do Produto de Software.

Segundo relato da organização, houve dificuldades em conseguir definir um processo de desenvolvimento de software, através de uma linguagem de notação de fácil entendimento. Adicionalmente, era necessário avaliar este processo através de avaliações para verificar a efetividade do mesmo.

Segundo a literatura Pressman (2006) e Sommerville (2007), existem diversos processos de desenvolvimento de software, entretanto, sempre precisam de adaptações, e nem sempre as adaptações atendem a realidade de uma organização.

1.2 Objetivo da Dissertação

De acordo com os problemas citados, os objetivo deste trabalho é a definição de um processo de Projeto e Construção de Produto de Software utilizando uma notação de fácil entendimento, que é o BPMN (*Business Process Modeling Notation*).

Outro objetivo é a realização de uma avaliação no processo de PCP, utilizando a abordagem de pontos fortes, pontos fracos e oportunidades de melhorias. Como forma de verificar a efetividade do processo de PCP, foram realizadas duas avaliações que ocorreram considerando parâmetros modelos quantitativos e qualitativos (método entrevista).

1.2.1 **Objetivos específicos**

Os objetivos específicos desta dissertação é descrito a seguir:

- Apresentar a importância dos processos de desenvolvimento de software;

- Apresentar a importância dos modelos de qualidades para um processo de software;
- Realizar a definição de um processo de Projeto e Construção do Produto que esteja de acordo aos modelos de qualidade, como o Melhoria de Processos do Software Brasileiro (MPS.BR).
- Avaliar o processo através do método quantitativo utilizando o *framework* do *Goal-Question-Metrics*(GQM), para auxiliar na avaliação do processo de PCP;
- Avaliar o processo através do método qualitativo utilizando um questionário, como ferramenta, distribuído para os participantes do processo.

1.3 Contribuições

De acordo com os objetivos descritos na seção 1.2, as seguintes contribuições são apresentadas nesta dissertação:

- Descrever uma metodologia para realizar a melhoria no processo de Projeto e Construção do Produto;
- Apresentar a abrangência do processo de Projeto e Construção do Produto com o modelo de qualidade do MPS.BR, nas áreas de Projeto e Construção do Produto;
- Descrever a modelagem de um processo de Projeto e Construção do Produto utilizando uma notação, de fácil entendimento, que é o BPMN;
- Apresentar a evolução do processo de Projeto de Construção de Produto de Software, onde os pontos fracos e oportunidades de melhorias listados foram atendidos, com o objetivo de obter um processo com maior maturidade;
- Descrever a realização de uma análise no processo de Projeto e Construção do Produto, utilizando a avaliação quantitativa através do *framework* GQM (*Goal-Question-Metrics*) nas auditorias nos ativos do projeto. Apresentando os resultados em forma de gráfico e realizando uma análise dos resultados obtidos;
- Descrever a realização da análise no processo de Projeto e Construção do Produto, utilizando a avaliação qualitativa através de uma entrevista, e como ferramenta a criação de um questionário, distribuído entre os participantes do processo de PCP, mostrando através de gráficos, fazendo a análise dos resultados;

1.4 Metodologia

Esta seção descreve como a metodologia adotada neste trabalho. Este trabalho foi estruturado nas seguintes etapas:

Levantamento inicial do cenário

- Pesquisa e estudo em artigos, documentos, livros, *sites* da área de Engenharia de Software, mostrando uma visão geral de um processo de desenvolvimento de software, modelos de qualidade, assim como suas limitações de evolução, análise e melhoria do processo de PCP;
- Estudo nos modelos, normas de qualidade para processos de software, com o intuito de realizar melhorias no processo de PCP;
- Realizar um estudo nas técnicas de métricas para avaliar um processo utilizando, o *framework* GQM;
- Realizar pesquisas na área de Qualidade de Software, que serviram como base para avaliar e prover melhorias no processo de desenvolvimento.
- Estudo na área de processo de projeto e construção do produto de software, que serviu de fundamentação para a elaboração deste trabalho.

Etapa do levantamento inicial do Processo de Projeto e Construção do Produto

- Levantamento de informações sobre o processo de PCP utilizado na Fábrica de software, com o intuito de verificar os pontos fortes, pontos fracos e oportunidade de melhorias, realizando assim uma análise no processo;
- Estudo de modelos de qualidades, com o objetivo de obter boas práticas para aplicação no processo de Projeto e Construção do Produto.

Etapa de formas de avaliação

- Estudo em notação de processos, através do *Business Process Model and Notation*. O objetivo é adotar uma notação de fácil entendimento para realizar uma modelagem no processo de PCP.
- Estudo em formas para avaliação quantitativa, com o intuito de avaliar se o processo de PCP estava funcionando de forma efetiva e se atende as necessidades de uma determinada organização;

- Estudo em formas para avaliação qualitativa, com o intuito de avaliar o processo de forma qualitativa e conhecer a satisfação dos participantes deste processo.

Etapa de melhorias

- Verificar os pontos fracos e oportunidades de melhorias, para serem contempladas, aumentando assim a maturidade do processo;
- Prover um plano de ação para realizar melhorias no processo de Projeto e Construção do Produto.

Com base na literatura especializada, Silva(2001), existem diversas formas para classificar a pesquisa realizada nesta dissertação, que pode utilizando uma pesquisa aplicada, gerando um conhecimento para a aplicação prática dirigidos à solução de problemas específicos.

Quanto a abordagem do problema, pode ser realizada através de duas pesquisas:

- **Pesquisa quantitativa**: pois considera que tudo pode ser quantificável, o que significa traduzir em números opiniões e informações para classificar e analisar;
- **Pesquisa qualitativa**: tudo o que não pode ser traduzido em números. A interpretação dos fenômenos e a atribuição de significados são básicas no processo de pesquisa qualitativa. Não requer o uso de métodos e técnicas estatísticas.

Quanto aos objetivos foram utilizados a pesquisa exploratória, pois tem o objetivo de proporcionar maior familiaridade com o problema, através de uma pesquisa bibliográfica, entrevistas com pessoas que tiveram experiências praticas com a melhoria do processo de PCP.

Foi utilizado a pesquisa descritiva, que tem como objetivo descrever as características de determinada população ou fenômeno ou estabelecer relações entre variáveis. Envolve o uso de técnicas padronizadas de coleta de dados: questionário e observação sistemática.

Quanto aos procedimentos técnicos, foram utilizadas a pesquisa bibliográfica, elaborada através da coleta de materiais publicados como livros, artigos de periódicos e materiais disponibilizados na Internet.

1.5 Organização da Dissertação

A estrutura desta dissertação segue organizada assim:

- CAPÍTULO 2 – apresenta o processo de Desenvolvimento de Software através de uma visão geral, apresentando conceitos, modelos e normas de qualidade, comparação entre a qualidade de software com a qualidade do produto e os trabalhos relacionados;
- CAPÍTULO 3 – apresenta uma visão geral do processo de Projeto e Construção do produto, mostrando sua importância. Adicionalmente serão apresentados os modelos de construção, o projeto de arquitetura, de interface e de dados;
- CAPÍTULO 4 – apresenta a contextualização do cenário do projeto, informações sobre o cliente, fornecedor e projeto, em seguida é apresentado o processo de PCP Versão 1. Após a contextualização é apresentada a evolução do processo;
- CAPÍTULO 5 – apresenta uma metodologia para realizar uma avaliação do processo de Projeto e Construção do Produto, utilizando as avaliações quantitativa e qualitativa e ao final uma análise de acordo com os resultados obtidos;
- CAPÍTULO 6 – apresenta as conclusões deste trabalho onde serão apresentadas as limitações do processo, os resultados obtidos, os trabalhos futuros e o trabalho publicado.

Capítulo 2 - **Processo de Desenvolvimento de Software: Visão Geral**

O processo de desenvolvimento de software engloba diversos estágios, desde a concepção à versão final entregue do produto, além de ter como um dos objetivos apoiar o gerenciamento e monitoramento de projetos. Para que este seja construído de forma íntegra é necessária uma organização lógica de diversas atividades de gerenciamento e técnicas envolvendo métodos, ferramentas, artefatos e recursos que possibilitam sistematizar, organizar e disciplinar o desenvolvimento e a manutenção do software. Falhas na gestão podem ocasionar impactos e riscos não previstos e consequentemente atrasos no cronograma, aumento de custos entre outros fatores que comprometem a evolução do projeto(Booch, 2006).

2.1 Conceitos Iniciais

Segundo Pressman (2006), nos últimos 20 anos o desenvolvimento de sistemas ficou mais custoso do que a aquisição de hardware dado a complexidade desses. Exemplo disso são sistemas hospitalares e aéreos, os quais possuem milhares de linhas de código e mobilizam para o desenvolvimento uma equipe técnica qualificada. Aliados a crescente demanda de desenvolvimento de software, problemas relacionados ao software crescem de maneira exponencial, ocasionando impactos na evolução dos projetos e, em casos extremos, a não utilização do produto pelo cliente(Pressman, 2006).

Um modelo de processo de software é uma representação abstrata de um processo de software. Cada modelo de processo representa um processo a partir de uma perspectiva particular, de modo que proporciona apenas informações parciais sobre o processo (Pressman, 2006). Durante o processo de desenvolvimento de um software, deve ser definido um conjunto de etapas chamado de modelo de ciclo de vida de software ou paradigmas da Engenharia de Software (Sommerville, 2007).

Processo de software refere-se a todas as atividades, bem como relacionamentos, artefatos, ferramentas, papéis etc., necessários para construir, entregar e manter um produto de

software. Já o ciclo de vida apresenta uma representação alto nível do processo de software executado (processo de software real) ou como deveria ser executado, ou seja, normalmente, ciclos de vida determinam as fases e o relacionamento entre as fases (Sommerville, 2007).

Modelo de processo de software, diferentemente do ciclo de vida, representa o processo de software executado ou como deveria ser executado em um nível de abstração mais minucioso que o ciclo de vida. Portanto, modelos de processo de software são representações que, além da representação de fases e relacionamento entre fases, descrevem as atividades executadas em cada fase e seus relacionamentos, artefatos consumidos e produzidos pelas atividades, papéis desempenhados, ferramentas utilizadas, etc. Alguns conceitos relacionados à definição de processos de software são apresentados nos trabalhos de Travassos(1994) e Falbo(1998).

• **Atividades**: as tarefas ou atividades são a forma "como as coisas são realizadas" ou ainda "o que será feito" e assim incorporam regras, procedimentos, políticas, recursos, papéis e artefatos. Todas as atividades devem ter início e fim definidos. As atividades podem ainda ser subdivididas, exemplo disso seria a decomposição da atividade gerenciamento de projeto nas subatividades: estimativas de custo, estimativas de tamanho, controle e acompanhamento;

• **Artefatos**: são também conhecidos como um tipo de recurso produzido ou consumido em uma atividade. Os artefatos podem ser utilizados como entrada para uma atividade assim como ser um resultado da execução de uma atividade. Além disso, um artefato pode estar associado a diversas atividades, as quais possibilitam a associação com diversos artefatos. Em um projeto de software, artefatos são arquivos lógicos versionáveis, pois ao sofrerem alterações são produzidas várias versões do mesmo artefato. Como exemplo, tem-se o código-fonte, manual de padrões, documento de requisitos;

• **Procedimentos**: são condutas/procedimentos estabelecidos e ordenados para realizar a execução das atividades. Determina as atividades que são realizadas. Quanto à sua natureza, procedimentos podem ser classificados em: métodos, técnicas e diretrizes. A UML *Unified Modeling Language*(Booch, 2006) pode ser utilizada como procedimento para a definição de cenários das necessidades do usuário;

• **Recursos**: são entidades estáticas necessárias para executar uma atividade. Os recursos podem ser classificados como: recursos humanos, recursos de software ou recursos de hardware. A não utilização ou indisponibilidade de recursos necessários pode causar falha na execução de uma atividade ou até mesmo impedir sua execução. Recursos relacionam-se com diversos componentes do processo de software, por exemplo, técnicas, métodos e

ferramentas. Deste modo, uma atividade pode utilizar diversos recursos e um recurso pode ser utilizado por diversas atividades ou, um agente pode utilizar diversos recursos e um recurso pode ser utilizado por diversos agentes. Artefatos, ferramentas, equipamentos, ambientes (sala de reuniões, laboratórios etc.) são considerados como tipos especializados (derivados) de recursos (Magela, 2006b).

Os processos de software podem apresentar grande complexidade e possibilitar diversas alternativas de execução de suas atividades. Desta forma, um processo de software definido permite que profissionais de engenharia de software possam trabalhar de forma ordenada, possibilitando um melhor entendimento do seu trabalho, bem como de outras atividades executadas por outros membros da mesma equipe (Bartié, 2002).

2.1.1 **Definição de Processo de Software**

Um processo é organizado em atividades as quais são de responsabilidade de um membro da equipe (colaborador) e devem gerar artefatos de saída verificáveis, podendo requerer também artefatos de entrada. Um artefato é um modelo, documento ou código produzido por uma atividade e quando disponibilizado ao cliente torna-se um entregável. Um processo deve estabelecer uma série de marcos, os quais são pontos finais de atividades do processo (Basili, 1994).

As organizações que realizam o desenvolvimento de software possuem um processo de software de acordo com sua realidade, entretanto, existe um conjunto de elementos fundamentais pré-definidos que necessita ser incorporado ao processo organizacional.

A norma ISO/IEC TR 15504 (ISO/IEC, 2006)define o conjunto destes elementos fundamentais como o processo padrão, ou seja, o processo básico que guia o estabelecimento de um processo comum na organização. Desta forma, um processo padrão define uma estrutura única a ser seguida por todas as equipes envolvidas em um projeto de software, independente das características do software a ser desenvolvido. Humphrey, em(Humphrey, 1989), define um conjunto de razões para a definição de um processo padrão:

- Redução dos problemas relacionados a treinamento, revisões e suporte a ferramentas;
- As experiências adquiridas nos projetos são incorporadas ao processo padrão e contribuem para melhorias em todos os processos definidos;
- A utilização de padrões de processo, fornecendo as bases para medições de processos e qualidade;

- Economia de tempo e esforço em definir novos processos adequados a projetos.

Esta padronização permite que os processos sejam especificados com uma terminologia unificada e promova o mapeamento dos processos e práticas recomendados pela norma ISO/IEC 12207 (ISO/IEC, 2008).

A maioria das empresas que realizam o desenvolvimento de software utiliza ou já cogitou utilizar pelo menos um dos padrões de definição de processos mais conhecidos (Koscianski, 2007), como por exemplo, a norma ISO/IEC 12207 (ISO/IEC, 2008)e suas emendas 1 e 2, a norma ISO/IEC TR 15504 (ISO/IEC, 2006),o CMMI – *Capability Matutity Model Intergration*(SEI, 2008)e o MPS.Br – Melhoria de Processo do Software Brasileiro (Softex, 2011).

Estes padrões definem um processo padrão, que por sua vez, pode ser transformado em um processo especializado, se adequado a realidade da empresa ou do produto que está sendo desenvolvido (Pressman, 2005).

Em empresas ou fábricas de software, diversos projetos podem coexistir possuindo características específicas. Sendo assim, define-se um modelo que pode ser especializado em instâncias para atender a necessidade do projeto ou da empresa que desenvolve o software. Desta forma, ser adaptável e configurável torna-se um importante requisito a ser atingido na definição de um processo padrão (Weber, 2006).

2.1.2 **Meta-Processo de Software Conceitos**

De acordo com Pressman (2006), um processo de software, assim como o seu modelo, tem uma natureza evolucionária. Essa natureza tem como causa a correção e melhoria feita durante o processo de desenvolvimento do produto.

As atividades realizadas no processo de software são denominadas de meta-atividades. Já o processo de desenvolvimento e a evolução de processos de software são chamados de meta-processo (Pressman, 2006).

Antes de iniciar qualquer atividade de um processo é necessário que o mesmo seja modelado para ser executado. Existem dois agentes humanos que podem realizar esta tarefa: o projetista, responsável pela modelagem do processo e o gerente do processo que acompanha a execução e a avaliação do processo, gerando como resultado um relatório de desempenho do processo.

Como forma de obter um melhor resultado, os modelos de ciclo de vida podem ser combinados. Em todos os ciclos existem essas fases em comum que são:

- Definição: “o quê”, esta fase é responsável pela análise do sistema, planejamento do projeto e análise de requisitos.
- Desenvolvimento: “como“, esta fase tem como objetivo o projeto de software, a codificação e a realização de testes do software.
- Manutenção: “mudanças”, esta fase tem como objetivo a correção de problemas, adaptação e evolução funcional do projeto

2.1.3 **Modelos e normas de qualidade**

A disciplina Qualidade de Software foi criada para garantir a qualidade do software através de definições e normatizações dos processos de desenvolvimento de software. Apesar da ênfase na disciplina estar no processo de desenvolvimento, seu principal objetivo é verificar que o produto final satisfaz a necessidade do cliente, de acordo com os critérios de aceitação do mesmo (Koscianski, 2007).

A importância dos modelos e normas de qualidade segundo Pfleeger (2004) baseia-se na observação dos insucessos de projetos de software devido à falta de um processo padronizado ou disciplinado, ocasionado pela ausência de mecanismos de controle de qualidade ou de gerência de produto. Em contrapartida, o sucesso dos projetos baseia-se na padronização durante o seu desenvolvimento e manutenção.

Desta forma, um processo padronizado é definido por uma única estrutura a ser seguida por todos os colaboradores envolvidos não importando as características do software que está sendo produzido (Pfleeger, 2004).

O software fez-se presente em diversas empresas de diferentes ramos, e muitas vezes sendo um diferencial competitivo. Por esse motivo é imprescindível que o mesmo seja desenvolvido dentro de padrões de qualidade, evitando que não se atendam exigências do cliente e eventuais falhas no projeto (Pressman, 2006).

Para contribuir na padronização do processo de software foram criadas organizações que avaliam a maturidade de determinada empresa. Dentre elas estão o CMM (*Capability Maturity Model*), ISO (*Institute of Organization for Standardization*) /IEC (*International Electrotechnical Commission*) 12207, e por fim o MPS.br (Modelo de Processo de Software Brasileiro).A seguir o modelo de maturidade europeu ISO/IEC será abordado.

2.1.4 **ISO**

Como citado anteriormente, as organizações responsáveis pela padronização na área de processo de software também definem e certificam empresas através do grau de maturidade das mesmas. Por exemplo, a Norma ISO/IEC 12207 e os modelos de maturidade contribuem para a definição de um processo padrão de software, o qual representa um importante ponto quando se almeja alcançar determinado nível de maturidade. Entretanto, a definição de um processo padrão não é trivial, pois, além de conhecer o funcionamento e organização da empresa, é necessário estar apto para trabalhar com padrões de desenvolvimento de software (ISO/IEC, 2008).

Entre os padrões existentes, a ISO tem como objetivo melhorar a qualidade do software. Dentro da série ISO 9000: o padrão ISO 9001 tem como foco a qualidade no desenvolvimento de sistema de software. A ISO realizou a publicação de diretrizes específicas para auxiliar na aplicação da ISO 9001 ao software: ISO 9003. A ISO 9000 é uma parte do sistema de qualidade, o qual é responsável também pelo comprometimento gerencial com qualidade, treinamento intensivo e ajuste de objetivos para melhoria contínua. Os padrões CMM e ISO 9000 enfatizam em medições. No entanto, também destacam que não basta medir para melhorar: é necessário treinar continuamente (ISO/IEC, 2008).

Segundo a norma ISO 9000 (versão 2000), a qualidade é o grau em que um conjunto de características inerentes a um produto, processo ou sistema cumpre os requisitos inicialmente estipulados para estes. No desenvolvimento de software, a qualidade do produto está diretamente relacionada à qualidade do processo de desenvolvimento, desta forma é comum que a busca por um software de maior qualidade passe necessariamente por uma melhoria no processo de desenvolvimento (ISO/IEC, 2008).

Norma ISO/IEC 12207

Em 1988 foi proposto o desenvolvimento da ISO/IEC 12207. Sua primeira versão foi publicada em agosto de 1995, sendo que a versão brasileira somente em 1998. Em 2002 e 2004 realizaram atualizações nas normas gerando as ementas 1 e 2, respectivamente. O objetivo da ISO/IEC 12207 é criar e padronizar uma estrutura comum para os processos de ciclo de vida de software assim como uma terminologia bem definida, que possa ser referenciada pela indústria de software. A ISO/IEC 12207 suporta atividades como desenvolvimento, operação, manutenção e a aquisição de um produto de software(ISO/IEC, 2008) (Machado, 2006).

A norma ISO/IEC 12207 tem sua estrutura idealizada de forma flexível, modular e adaptável o que possibilita cada organização adaptá-la as suas necessidades. Então, essa norma é fundamentada em dois princípios básicos: modularidade e responsabilidade. A modularidade pode ser definida como um baixo acoplamento e uma alta coesão no que diz respeito a processos. Enquanto responsabilidade tem como objetivo alocar um responsável por cada processo da empresa, o que facilita a aplicação da norma em projetos que envolvem diversas pessoas (ISO/IEC, 2008) (Salviano, 2006).

A ISO/IEC 12207 é aplicada em todo ciclo de vida do projeto, desde o momento de sua concepção até o momento que o projeto é descontinuado na organização, o que resultou na criação de três classes de processos: Processos Fundamentais, Processos de Apoio, Processos Organizacionais e Processos de Adaptação (ISO/IEC, 2008).

2.2 Processos de apoio

Todas as atividades que a empresa realiza, podem ser denominadas de processos fundamentais. Essas atividades podem ser definidas como, por exemplo, desenvolvimento, manutenção e operação. Sendo que os cincos processos fundamentais de ciclo de vida são (ISO/IEC, 2008):

• Aquisição – tem como seu principal objetivo definir as atividades a serem realizadas pela própria empresa ou da terceirizada que realizará o desenvolvimento de um software ou serviço, garantindo assim a satisfação do cliente. Esse processo inicia-se na identificação das necessidades e dos problemas e termina com a aceitação do produto ou serviço.

• Fornecimento – o objetivo do fornecimento é que o produto ou serviço atenda o que foi levantado nos requisitos do sistema (Lahoz, 2003).

• Desenvolvimento – o processo de desenvolvimento contém tarefas e atividades, dentre elas, levantamento dos requisitos, análise de requisitos, projeto, construção, integração, testes e por fim configuração do ambiente de forma adequada para produção do cliente.

• Operação – são atividades de apoio ao produto, como suporte operacional aos usuários. O objetivo é orientar usuários treinados no sistema a elucidar dúvidas para uma melhor utilização do mesmo (Lahoz, 2003).

• Manutenção – o processo de manutenção pode ser definido quando um software precisa de modificações no código, seja por caráter evolutivo ou caráter de manutenção. O objetivo é garantir a integridade do produto.

Segundo Machado (2006), a norma ISO/IEC 12207 (ISO/IEC, 2008) tem sido utilizada para apoiar e padronizar o processo de ciclo de vida durante o desenvolvimento de um produto. Esta norma possui como um ponto forte a alta granularidade dos processos, permitindo assim a definição de pequenos processos que serão integrados durante a execução.

2.3 Processos fundamentais

Os processos de apoio são fundamentais para auxiliar no sucesso e na qualidade do projeto desenvolvido.

Segundo a norma, a documentação do sistema é executada como última tarefa para a o desenvolvedor. Esta atividade sempre acontece depois do desenvolvimento concluído de uma determinada fase. Para validação e verificação, o processo é repetido inúmeras vezes até estar aderente ao modelo (Brogini, 2002).

Os processos organizacionais são empregados por uma organização para determinar e implementar uma estrutura constituída pelos processos de ciclo de vida e pelo pessoal envolvido no desenvolvimento do software. Eles são geralmente empregados fora do domínio de projetos e contratos específicos; entretanto, os ensinamentos desses projetos e contratos contribuem para a melhoria da organização, são eles: Processos de Gerência, Infra-estrutura, Melhoria, e Treinamento (Machado, 2006).

O processo de gerência depende diretamente do porte da empresa. Em alguns casos existirá a pessoa responsável, em outros o próprio desenvolvedor assumirá o papel, mas não a função, ou seja, aparecerá como responsável, mas devido à falta de tempo e de recursos humanos fará todo o trabalho e não praticará a gerência do projeto (Lahoz, 2003).

2.4 Processos de adaptação

O processo de adaptação define as atividades necessárias para adaptar a norma para sua aplicação na organização ou em projetos. A adaptação deve ser executada com base em alguns fatores que diferenciam uma organização ou projeto de outros, dentre os quais a estratégia de aquisição, modelos de ciclo de vida de projeto, características de sistemas e software e cultura organizacional.

A existência desse processo permite que a norma seja adaptável a qualquer projeto, organização, modelo de ciclo de vida, cultura e técnica de desenvolvimento (Machado, 2006).

A seguir o modelo de maturidade CMMI será abordado.

2.5 CMMI (*Capability Maturity Model Integration*)

O CMMI é mais um padrão que tem como objetivo melhorar o processo de produção de software, independente do ciclo de vida adotado. O instituto responsável pela elaboração é o SEI(*Software Engineering Institute*) da Universidade de Carnegie-Mellon, USA. O modelo CMM abrange o processo software tanto no aspecto técnico como no aspecto gerencial e está dividido em 5 níveis. Os cinco níveis são (SEI, 2008):

O modelo tem como objetivo estabelecer, com base em estudos, históricos e conhecimento operacional, um conjunto de "melhores práticas" que devem ser utilizadas para um fim específico (Chrissis, 2002). Os cinco níveis do CMMI estão descritos no Quadro 2-1.

Quadro 2-1 – Níveis de maturidade do CMMI

Nível	**Definição**	**Descrição**
1	Inicial	Processo sem controle e imprevisível.
2	Repetível	Processo disciplinado
3	Definido	Processo padronizado e consistente
4	Gerenciado	Processo controlado e previsível
5	Otimizado	Processo continuamente melhorado

É importante ressaltar que cada nível possui uma série de áreas de processos chaves que a organização deve cumprir antes de passar para o nível seguinte (SEI, 2008).A versão do CMMI (versão 1.2) apresenta dois modelos (SEI, 2008):

- CMMI for *Development* (CMMI-DEV) publicada em agosto de 2006. Dirige-se ao processo de desenvolvimento de produtos e serviços.
- CMMI for *Acquisition* (CMMI-ACQ) publicada em novembro de 2007. Dirige-se aos processos de aquisição e terceirização de bens e serviços.

O primeiro passo para se iniciar uma evolução contínua consiste em entender o processo em andamento. O passo seguinte consiste em formular o processo pretendido. As ações necessárias para melhorar o processo são determinadas e ordenadas por prioridade. O plano de melhoria é montado e executado. Esta série de passos é repetida (Chrissis, 2002) (SEI, 2008).

O primeiro nível do *Software-Capability Maturity Model* SW-CMM é o nível 1 o qual é caracterizado como imprevisível e ocasionalmente caótico e na maioria dos casos os

processos não estão definidos. Dessa forma o processo de software é visto como uma caixa-preta onde somente a entrada e a saída podem ser detectadas com clareza (Staples, 2007).

Geralmente as empresas com o CMM nível 1 demonstram dificuldades com cronogramas e planos irreais, pois não existe um planejamento a ser seguido, não há controle sobre os requisitos gerados, o cliente avalia apenas na entrega do produto (Staples, 2007).

Como as fases de desenvolvimento de software não são seguidas é comum a especificação de requisitos ser encaminhada para codificação sem que seja realizada a análise prévia do mesmo. A maioria das empresas encara a documentação como algo inútil (SEI, 2008).

O segundo nível do SW-CMM é denominado de repetível, ou seja, é possível repetir o sucesso de um projeto anterior nos projetos futuros similares. Esse nível também define como o objetivo o processo básico de gerência de projetos onde são estabelecidos prazos, custos, escopo e cronograma (SEI, 2008).

O nível 2, diferentemente do nível 1, é visualizado como diversas caixas pretas, ou seja, é possível verificar o andamento através de macro ciclos do projeto. O segundo grau de maturidade possibilita que a organização consiga determinar custos, prazos e escopo com maior margem de acerto. Entretanto, a gerência de projetos ainda não é considerada proativa, apesar de haver o acompanhamento do cronograma e custos de cada projeto (Staples, 2007).

Os projetos podem ter processos diferentes, entretanto é necessário que se eleja um único para que seja adotado nos demais, além da existência do controle da evolução dos requisitos do sistema permitindo uma avaliação ao final de cada macro ciclo do projeto (SEI, 2006) (Machado, 2006).

O próximo nível é o 3 ou definido, no qual a organização tem suas tarefas internas definidas e visíveis para o cliente, por exemplo. Todos os colaboradores utilizam uma versão aprovada e adaptada do processo para o desenvolvimento e manutenção de software. Vale ressaltar que mesmo que exista a padronização é permitido adaptar os processos de modo que atenda às necessidades particulares de outro projeto (Machado, 2006).

Outra característica importante é que no nível 3 existe um treinamento técnico e gerencial e a organização consegue permanecer executando o processo mesmo em momentos de crise. Os processos pertencem à organização e não aos projetos (Staples, 2007).

Para tal, existe o grupo de especialistas denominado de SEPG (*Software Engineering Process Group*), o qual é responsável pelos processos da empresa. E ainda mesmo que um

desenvolvedor abandone o projeto o impacto é relativamente menor do que nos níveis 1 e 2, já que o processo é bem definido (Machado, 2006) (Staples, 2007).

O próximo nível do CMM é o nível 4 ou gerenciado, nesse nível o processo e o produto de software são controlados quantitativamente (controle estatísticos de processos) e métricas são aplicadas ao projeto para determinar a qualidade do produto gerado. No nível 4, a empresa ou organização define metas de qualidades quantitativas e a produtividade para as atividades do processo e para os produtos gerados. Ao longo do projeto medidas de qualidade são coletadas para que os gerentes possam avaliar a evolução do desenvolvimento bem como a ocorrência de problemas no produto (SEI, 2008).

O último nível do SW-CMM é o nível 5 ou otimizado, neste nível a melhoria é continua, durante o projeto ocorrem avaliações quantitativas e implantação planejada e controlada de tecnologias e ideias inovadoras. O objetivo é melhorar os processos constantemente, já que o grau de maturidade alcançou o patamar máximo no CMM. Através dessa evolução constante, é possível avaliar fraquezas e fortalecer o processo de forma proativa, evitando riscos que possam surgir futuramente. Qualquer mudança mais significativa no processo ou de tecnologias é feita a partir de análises de custo / beneficio com base em dados quantitativos, do nível anterior (SEI, 2008).

O próximo modelo de maturidade que será abordado nesse capitulo é o modelo de processo de software Brasileiro ou MPS.BR.

2.6 Mps.Br

Qualidade, assim como para outros setores, é um fator muito relevante para o sucesso da indústria de software brasileira. Para que o Brasil tenha um setor de software competitivo, internamente e internacionalmente, é importante que os empreendedores do setor coloquem a eficiência e a eficácia dos seus processos em foco nas empresas, visando a oferta de produtos de software e serviços correlatos conforme padrões internacionais de qualidade. Alcançar competitividade através da qualidade, implica tanto na melhoria da qualidade dos produtos de software e serviços correlatos, como dos processos de produção e distribuição do mesmo(Softex, 2011).

A crise de software, que afetou o mercado internacional, influenciou as empresas brasileiras e ocasionou a criação do Programa de Melhoria de Processo de Software Brasileiro (MPS.Br). Como fatores motivadores para a organização do MPS.BR podemos citar: o aumento da qualidade do produto, a diminuição do retrabalho, maior produtividade, redução

do tempo para atender o mercado, maior competitividade, maior precisão nas estimativas, acompanhamento da satisfação do cliente (Softex, 2011).

O projeto do MPS.Br é coordenado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), tendo o apoio do Ministério da Ciência e Tecnologia(MCT), do Banco Interamericano de Desenvolvimento (BID) e da Financiadora de Projetos (FINEP). Apesar de ser um modelo definido para o cenário nacional Brasileiro, tem o objetivo de difundir-se para a América Latina, incluindo Argentina, Chile, Peru e Uruguai (Softex, 2011).

A equipe de coordenação do MPS.BR possui dois grupos de apoio: Fórum de Credenciamento e Controle (FCC) e Equipe técnica do Modelo (ETM), além da participação de universidades, centros de pesquisas e instituições governamentais (Softex, 2011)

O objetivo do FCC e ETM são, respectivamente, garantir que as instituições implementadoras e as instituições avaliadoras sejam submetidas a um processo de credenciamento e atuar sobre aspectos técnicos relacionados ao modelo de referência (MR-MPS) e o método de avaliação (MA-MPS) assim como elaborar e evoluir o modelo, atualizar os guias, preparar o material dos treinamentos e aplicar as provas, dentre outras responsabilidades (Softex, 2011).

O MPS.BR possui um guia geral que descreve o modelo de referencia (MR-MPS) e outros guias como o de avaliação e aquisição. O guia geral foi baseado na Norma Internacional ISO/IEC 12207: 1995/Amd 1:2002 2:2004, a ISO 15504 e o modelo CMMI-SE/SW. O CMMI-SE/SW também foi baseado na Norma ISO/IEC 12207 e 15504, entretanto, existem diferenças nos dois modelos, como por exemplo, a quantidade de níveis de maturidade (Softex, 2011).

O Mps.Br apresenta 7 níveis de maturidade, conforme o Quadro 2-2, cada nível de maturidade possui suas áreas de processo, onde são analisados os processos fundamentais: (aquisição, gerência de requisitos, desenvolvimento de requisitos, solução técnica, integração do produto, instalação do produto, liberação do produto), processos organizacionais (gerência de projeto, adaptação do processo para gerência de projeto, análise de decisão e resolução, gerência de riscos, avaliação e melhoria do processo organizacional, definição do processo organizacional, desempenho do processo organizacional, gerência quantitativa do projeto, análise e resolução de causas, inovação e implantação na organização) e os processos de apoio (garantia de qualidade, gerência de configuração, validação, medição, verificação, treinamento) (Softex, 2011). Os 7 níveis do MPS.BR estão descritos no Quadro 2-2.

Quadro 2-2 - Níveis de maturidade do MPS.BR

Nível	Definição
G	Parcialmente Gerenciado
F	Gerenciado
E	Parcialmente definido
D	Largamente Definido
C	Definido
B	Gerenciado Quantitativamente
A	Em Otimização

2.7 Qualidade do Processo x Qualidade do Produto

Todo produto de software produzido deve possuir um padrão de qualidade definido entre o contratante e a contratada. Para a construção são necessárias várias fases e atividades, que por sua vez influenciam na qualidade do produto final. A execução ineficiente ou a não execução de uma ou mais atividades pode gerar um produto deficiente, ou seja, fora do padrão de qualidade acordado. Sendo assim, para garantir a qualidade do produto é importante que o processo seja aplicado corretamente(Pfleeger, 2004).

Características de qualidade de um produto de software são os atributos nos quais sua qualidade pode ser descrita e avaliada. A Norma ISO/IEC 9126 (Softex, 2011)objetiva fornecer uma arquitetura para avaliação da qualidade do produto de software. Para atender tal objetivo, a norma define um modelo de qualidade aplicável a qualquer tipo de software. São definidas 6 (seis) características de qualidade(Oliveira, 2006).:

• **Funcionalidade**: são funções que atendem necessidades declaradas assim como suas propriedades específicas;

• **Confiabilidade**: é a capacidade de armazenar as informações de forma íntegra, como por exemplo, disponibilizar informações relevantes para determinado período de tempo e condições pré-estabelecidas em contrato.

• **Usabilidade**: é a forma de verificar o quanto a interface atende às demandas do usuário, e se está de acordo com o padrão acordado entre a empresa contratante e a empresa contratada, caso houver;

• **Eficiência**: é o nível de desempenho do software e a quantidade de recursos utilizados, sob condições estabelecidas;

• **Manutenbilidade**: é o esforço necessário para realizar manutenções corretivas e evolutivas no produto de software;

• **Portabilidade**: capacidade de um software ser executado em servidores ou máquinas de diferentes tipos, sistemas operacionais, processadores e até mesmo arquitetura.

A qualidade de software pode ser definida de acordo com o processo definido e de como é utilizado pela empresa. Mesmo assim, é importante salientar que um bom processo de software não garante que os produtos de softwares gerados tenham qualidade suficiente para a satisfação do cliente. Entretanto, isso demonstra que a organização tem a capacidade de produzir produtos de software com boa qualidade (Oliveira, 2006).

2.8 Trabalhos Relacionados

Durante o desenvolvimento deste trabalho foram pesquisados diversos trabalhos relacionados, de forma a gerar um embasamento teórico. Devido ao sigilo empresarial não foi possível coletar dados de outras empresas que utilizem processos de desenvolvimento de software. Porém, alguns trabalhos sobre melhoria do processo de software foram pesquisados com o objetivo de melhor guiar a institucionalização do processo descrito neste trabalho.

Yoon(2001) descreve em seu trabalho que a formalização de processo, integrando-o com módulos de processo encapsulados, constitui um método sistemático para adaptação de cada módulo do processo.

Os autores Coelho (2003) e Maia (2004) propõem em seus trabalhos uma adaptação de processo para organização que já tem um processo definido, mas também novos processos podem ser adaptados. Ambos os trabalhos relatam que a adaptação é orientada por regras associadas apenas a atividades do processo.

Outro trabalho relacionado é de Münch (2004) que aborda em seu trabalho um modelo de processos adaptável definido no princípio de sistemas de transformação. Um processo pode sofrer várias alterações até que esteja adaptado.

Segundo Guerra et al, 2006 propõem em seu trabalho a evolução de um processo de software e hardware, através de melhorias do processo de desenvolvimento. O Resultado foi a obtenção do CMMI nível 2.

Macedo et al, 2007 propõem em seu trabalho a implantação da melhoria do processo de Software no Tribunal Superior Eleitoral. O objetivo é evoluir o processo para alcançar o segundo nível do CMMI.

Silva et al propõem em seu trabalho a necessidade de realizar alterações nos processos para adequá-los às especificidades do negócio e aos objetivos gerais da organização. A partir de avaliações e experiências bem conduzidas, as organizações podem identificar e adaptar boas práticas às suas necessidades e incorporá-las em seus processos. Medições de produto e processo podem ser usadas para avaliar a efetividade das melhorias. Comparações entre resultados de projetos-piloto e dados históricos, bem como tendências gerais podem testar e demonstrar os efeitos da mudança do processo.

As comparações entre os trabalhos relacionados e a proposta deste trabalho são apresentadas no Quadro 2-3.

Quadro 2-3 – Comparação dos trabalhos relacionados

	Processo de PCP	**Yoon (2001)**	**Coelho(2003) Maia (2004)**	**Münch (2004)**	**Guerra (2006)**	**Macedo(2006)**	**Silva (2007)**
Aborda como foi definido o processo	✓	✗	✓	✓	✗	✗	✗
Aborda como foi realizado a avaliação	✓	✗	✓	✓	✓	✓	✓
Descreve a evolução do processo	✓	✓	✗	✓	✓	✓	✓
Utiliza algum modelo de qualidade	✓	✓	✓	✓	✓	✓	✓
Utiliza avaliação quantitativa para avaliar o processo	✓	✗	✗	✗	✗	✗	✗
Utiliza avaliação qualitativa para avaliar o processo	✓	✗	✗	✗	✗	✗	✗

Legenda da Tabela	
✓	Contém
✗	Não Contém

Capítulo 3 - **Processo de Projeto e Construção de Produto (PCP): Visão Geral**

A Engenharia de software trata da criação e utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, confiável e que trabalhe eficientemente em máquinas reais (Sommerville, 2007).

Na década de 1960 foi criado o termo Engenharia de Software e utilizado oficialmente na Conferência sobre Engenharia de Software da OTAN (NATO – *Conference on Software Engineering*) (Sommerville, 2007). O objetivo de sua criação é o de contornar a crise de software e dar um tratamento de forma controlada ao desenvolvimento de softwares considerados de complexidade alta (Pressman, 2006).

A complexidade de um software pode ser medida de diversas formas, como por exemplo, a técnica de Pontos de Caso de Uso, onde é possível verificar a complexidade de um determinado sistema (Pressman, 2006).

Como forma de contribuir no desenvolvimento de sistemas complexos, é importante compreender o uso de modelos abstratos, desse modo, permitir ao engenheiro realizar a especificação, o projeto, o desenvolvimento e a manutenção do software, avaliando e garantindo suas qualidades (Xavier, 1995). Empresas desenvolvedoras de software passaram a empregar esses conceitos, sobretudo para orientar suas áreas de desenvolvimento, muitas delas organizadas sob a forma de Fábrica de Software (Magela, 2006) .

Este capítulo está dividido da seguinte forma, na seção 3.1 será abordado a importância do processo de Projeto e Construção de produto no contexto do desenvolvimento de software, mostrando um histórico, a sua evolução.

Na seção seguinte será mostrado a construção do produto, onde é apresentado os modelos de construções, os ciclos de vida para a construção de um produto de software. Na seção 3.3 será abordado a definição de projeto do sistema, onde será abordado o projeto de arquitetura de software, interface e de dados.

3.1 A importância do Processo de Projeto e Construção de Produto no contexto do desenvolvimento de software

Em 1979 foi escrito um livro pelo autor Tom DeMarco chamado de "*Structured Analysisand System Specification*" (análise estruturada e especificação de sistemas) (DeMarco, 1983), onde são formalizados alguns princípios e estabelecidas algumas técnicas que introduzem a atividade de análise de sistemas na era estruturada. Nessa mesma época outros estudiosos do assunto, como Gane e Sarson (Gane, 1988), desenvolveram pesquisas e chegaram a conclusões que mais tarde somariam-se às de Tom DeMarco (DeMarco, 1983). O modelo que caracterizava a análise estruturada era o Diagrama de Fluxo de Dados (DFD). Esse diagrama passou a representar em forma de rede os processos de um sistema e o fluxo dos dados dos processos entre si e com as entidades externas ao sistema (Gane, 1988).

A melhoria da qualidade de um software, segundo Codd, Bachman e Chen (Chen, 1987), iniciava na modelagem da informação. Na década de 70 esse processo era conhecido como Engenharia da Informação e contou com um impulso bastante significativo da IBM. Essa linguagem passou a ser conhecida como SQL (*Structured Query Language* — linguagem estruturada, para consulta) para manipulação de bancos de dados (Chen, 1990).

O autor Peter Chen publicou em 1976, um artigo intitulado "*The Entity-Relationship Model, Toward a Unified View of Data*" (o modelo entidade-relacionamento, rumo a uma visão unificada dos dados — Communications of the ACM), formalizando um modelo que representa a "malha de memória" do sistema (Pressman, 2006).

Outro tipo de sistema ou a terceira espécie de sistemas, cuja implementação era derivada de um modelo de análise diferente dos modelos funcional e da informação, tratavam-se dos softwares baseados nos modelos de estado, orientados para circuitos digitais. Esse sistema é chamado de “softwares embarcados” ou sistemas em tempo real que são geralmente utilizados em máquinas como, por exemplo, aviões, elevadores, automóveis e etc.(DeMarco, 1983).

Abaixo será caracterizada a diferença entre os três modelos: o funcional, o da informação e o de estados, na perspectiva inicial da análise e do projeto:

• Funcional: parte da decomposição dos processos (funções) da organização em estudo, para serem derivadas as estruturas de dados necessárias para o estabelecimento do sincronismo entre essas funções(Pfleeger, 2004);

• Informação: inicia o processo de análise identificando-se o modelo conceitual de dados (modelo que representa a visão dos usuários sobre as informações necessárias), que é

decomposto nos conjuntos de dados do software (utilizando-se de uma técnica conhecida como normalização) e, a partir desses conjuntos de dados, são implementadas as operações que se aplicam sobre eles(Pfleeger, 2004);

• Estados: inicia o processo de análise na identificação dos possíveis estados que um determinado dispositivo pode assumir (aberto, fechado, ligado, etc.) para serem obtidos os processos que levariam esses dispositivos de um estado para outro e os eventos que fariam com que esses processos fossem disparados(Pfleeger, 2004).

Tom DeMarco(1983) agrupou essas três visões sobre sistemas com o objetivo de representar os sistemas em perspectivas, ou em três dimensões. Os defensores de cada idéia deixaram de competirem entre si, teve início um movimento de coesão de conhecimentos, que serve como base às ferramentas e metodologias atuais.

Na década de 80 até o início da década de 90 esses conceitos foram ampliados. Porém, iniciava-se outro problema que era a dificuldade na manutenção dos modelos que eram gerados nas diversas fases do desenvolvimento, principalmente nas atividades de análise e projeto do sistema pela perspectiva das funções (Sommerville, 2007).

Stevens (1988) cita a importância do reaproveitamento máximo de funções já desenvolvidas, entretanto, apesar de teoricamente fundamentado, não se mostrou muito produtivo, até mesmo quando da aplicação desses modelos em projetos considerados pequenos (300 a 400 pontos de função), pela dificuldade apresentada na localização de uma função específica (estima-se que os softwares de gestão empresarial de mercado possuam entre 40.000 a 160.000 pontos de função)(Stevens,1988 apud Sommerville, 2007).

Assim, uma forma mais eficiente de estruturação dessas funções mostrou-se necessária. Nesta época, essa forma tem sido apresentada como um conjunto de técnicas e disciplinas denominadas de orientação por objetos(Sommerville, 2007).

A programação orientada a objeto foi discutida pela primeira vez no final dos anos 60 (Pressman, 2006). As pessoas envolvidas na discussão eram aqueles que trabalhavam com a linguagem SIMULA. Nos anos 70, foi desenvolvida na Xerox PARC uma parte importante da linguagem *Smaltalk*. O conjunto total das técnicas orientadas para objetos tinham por objetivo, inicialmente, "auxiliar no gerenciamento da complexidade de grandes softwares de tempo real" (Booch, 2006).

Na década de 80 alguns autores previam que as técnicas orientadas a objetos estariam para os anos 80 o que a programação estruturada representou para a década de 70. A diferença

entre as técnicas estruturadas e as orientadas para objetos está na forma de decomposição das operações a serem executadas pelo software (Booch, 2006).

A programação estruturada fundamenta-se na decomposição das funções ou procedimentos de uma unidade hierarquicamente estruturada, ou seja, o programa. A programação orientada tem como fundamento a distribuição das funções por objetos e classes de objetos, sobre os quais essas funções podem ser aplicadas (Pressman, 2006).

A programação orientada a objetos seguiu os passos da programação estruturada, do ponto de vista das semelhanças nas pesquisas sobre técnicas e na produção de software(Pressman, 2006).

Da mesma forma como a estruturação transformou-se num paradigma, afetando os processos de projeto, análise e gerenciamento do desenvolvimento de software, os objetos têm se transformado num paradigma desses mesmos processos(Pressman, 2006).

Entretanto, é importante ressaltar que o paradigma de orientação a objetos, apesar de agregar qualidade nos softwares desenvolvidos, não atende totalmente o ciclo de vida do software, visto que os pontos principais do mesmo são os estágios de projetos e implementação, deixando as fases de requisitos e análise descobertas(Booch, 2006).

Sugere-se, então, que sejam agregados à orientação a objetos métodos específicos às fases de requisitos e análise, para que questões relacionadas ao controle de projetos de software, como requisitos não coletados adequadamente ou distorcidos durante o ciclo de desenvolvimento, sejam amenizadas e/ou sanadas de forma que se tenha o sucesso do projeto (Sommerville, 2007).

Adicionalmente ao estudo da orientação surge a Orientação a Aspecto (*aspect-oriented-programming* ou simplesmente AOP). A Orientação a Aspecto surgiu com o objetivo de separar algumas funcionalidades básicas do sistema com as regras de negócio. Como por exemplo, pode-se citar as funcionalidades de log de auditoria de uma aplicação, sem a necessidade de escrever uma linha de código nos método (Sommerville, 2007).

Para um melhor entendimento, o aspecto funciona da seguinte forma: o aspecto por sua vez, consegue ”capturar” trechos de códigos durante sua execução e realizar diversas funcionalidades. A vantagem é que o código torna-se limpo, garantindo uma melhor legibilidade (Sommerville, 2007).

O nome do comportamento do AOP que atravessa todo o sistema é chamado de *crosscutting*, porque ele atravessa a divisão de responsabilidades dos modelos de programação tradicionais (Sommerville, 2007).

AOP complementa a programação orientada a objeto por facilitar um outro tipo de modularidade que expande a implementação espalhada de uma responsabilidade dentro de uma simples unidade. Esta unidade é chamada de aspecto, daí o nome programação orientada a aspectos. Os interesses tornam-se fáceis de tratar porque estão concentrados no código de aspectos e afetaram todas as unidades desejadas (Sommerville, 2007).

Os aspectos de um sistema podem ser alterados, inseridos ou removidos em tempo de compilação, e frequentemente reusados. Por estarem em um único bloco de código a manutenção é muito mais fácil e a complexidade do sistema diminui, facilitando o entendimento do mesmo (Sommerville, 2007).

Outra tecnologia que atualmente está sendo utilizada por algumas empresas é a Arquitetura Orientada a Serviços ou *Service Oriented Architecture* (SOA) (Erl, 2009), que é uma solução que visa utilizar conceitos de computação distribuída. SOA permite projetar componentes de software que através de uma rede lógica, interagem entre si. O nome desses componentes são chamados de serviços, que nada mais são que funcionalidades desenvolvidas para que se possam ser reutilizados entre diversas organizações. Segundo Davydov essa utilização pode se tornar muito complexa (Davydov, 2007).

A sua utilização pode-se tornar complexa já que existem vários sistemas interagindo entre si, ficando difícil de entender e predizer o que pode ocorrer. Segundo Tan o grande número de componentes cria dificuldade em compreender a estrutura relacionada, dessa forma não é possível prever o comportamento de um sistema sem aparente padrão, fazendo com que ele pareça aleatório (Tan, 2005).

Após esta abordagem teórica nas técnicas de programação a próxima seção irá mostrar a construção de um produto de software.

3.2 Construção de Produto

O desenvolvimento de software tem o objetivo de desenvolver um sistema computacional, ou seja, transformar a necessidade de um utilizador ou de um mercado em um produto de software. As experiências iniciais dos programadores no desenvolvimento de software mostraram que uma abordagem informal na construção não era o bastante para se obter sucesso no processo (Sommerville, 2007).

Nos anos 70, quando os gerentes de tecnologia da informação (TI) não tinham ideia de como desenvolver um software com qualidade, usando uma metodologia específica e também

com uso de técnicas de engenharia de software, a ideia era de produzir softwares que fossem apenas eficientes. Porém, com o passar do tempo, essa visão ficou ultrapassada e, hoje em dia, fica claro que o desenvolvimento de software tem de gerar um produto que por si só seja eficaz, em tempo hábil, e com orçamento previsível (Sommerville, 2007).

Os projetos de grande porte estavam com a qualidade aquém do esperado, com o orçamento elevado e atraso no cronograma. A solução para minimizar o problema foi à utilização de uma metodologia de desenvolvimento de software para garantir a qualidade no processo de desenvolvimento e também no produto final (Weber, 2006).

O planejamento do processo de desenvolvimento de software é importantíssimo em se tratando de construção de sistemas robustos. A construção de um software, quando não se utiliza as metodologias de engenharia de software, pode gerar perdas de monetária ou, o que é pior, prejuízos incalculáveis para as organizações. Outro aspecto importante é que a manutenção do software seria muito complicada e dificilmente sustentável(Tonsig, 2008) (Weber, 2006).

A maior parte das empresas que realizam o desenvolvimento de software não faz o uso de metodologias da engenharia de software. O motivo é que o importante pra o cliente é ver o sistema pronto o mais rápido possível. Um processo de desenvolvimento de software com qualidade requer tempo que, muitas vezes, o cliente não está disposto a esperar (Weber, 2006).

O desenvolvimento de software deixou de ser sinônimo apenas de código. Sabe-se que é necessária a utilização de uma metodologia de trabalho para obter um sucesso do produto fina (Weber, 2006).

A metodologia é a maneira – forma – de se utilizar um conjunto coerente e coordenado de métodos para atingir um objetivo, de modo que se evite, tanto quanto possível, a subjetividade na execução do trabalho. Fornecendo um roteiro, um processo dinâmico e interativo para desenvolvimento estruturado de projetos, sistemas ou software, visando à qualidade e produtividade dos projetos (Sommerville, 2007).

A metodologia tem o objetivo de definir “quem” faz “o que”, “quando”, “como”, e até mesmo “onde”, para todos os que estejam envolvidos diretamente ou não com o desenvolvimento de software (Sommerville, 2007).

É importante definir também qual o papel dos técnicos, dos usuários, e o da administração da empresa no processo de desenvolvimento. Com isso, evita-se a situação a qual o conhecimento sobre o sistema é de poucos, também conhecido como “os donos do

sistema". Adicionalmente, deve instruir um conjunto de padrões preestabelecidos, de modo a ser evitar a subjetividade na abordagem, com a meta de garantir fácil integração entre os sistemas desenvolvidos. Com isso, o uso de uma metodologia possibilita (Tonsig, 2008):

• Ao gerente: gerenciar o projeto de desenvolvimento de software mantendo o rumo do projeto sobre controle para que não haja desvios de planejamentos de custos e prazos, que, se negligenciados ou mal conduzidos, podem por em risco o sucesso do projeto (Sommerville, 2007).

• Ao desenvolvedor: obter a base para produzir de maneira eficiente, software de qualidade que satisfaça os requisitos estabelecidos.

A utilização de uma metodologia de software é encarada como:

• Tolir a criatividade dos técnicos,

• Acréscimo de burocracia leia-se documentações, por muitos tidos como desnecessário a construção de software.

Uma metodologia não deve limitar a criatividade profissional, mas deve ser um instrumento que determine um planejamento sistemático, que harmonize e coordena as áreas envolvidas. O que limita a criatividade não é a metodologia, mas os requisitos de qualidade e produtividade de um projeto (Tonsig, 2008).

A Metodologia de Desenvolvimento de Software são as boas práticas do desenvolvimento de software, pois estabelece ordem nas atividades a fim de possibilitar a conclusão de tarefas e/ou objetivos e oferecer suporte ao gerenciamento permitindo a repetição do desenvolvimento em busca de problemas ou erros para serem corrigidos (Sommerville, 2007).

Um método é abordagem técnica passo a passo para realizar uma ou mais tarefas indicadas na metodologia. Ou seja, é (são) o(s) procedimento(s) necessário(s) a ser (em) adotado(s) para atingir um objetivo. Já uma técnica, pode ser compreendida como sendo um modo apropriado de se investigar sistematicamente um universo de interesse ou domínio do problema. Para tanto, utiliza-se de uma notação. Como exemplo de técnica, temos: Análise estruturada, Análise Essencial, Projeto Estruturado, Análise Orientada a Objetos (Sommerville, 2007).

A escolha de uma metodologia a ser utilizada no desenvolvimento, deve ser realizada com base na natureza do projeto e do produto a ser desenvolvido, dos métodos e ferramentas a serem utilizadas e dos controles e produtos intermediários desejados (Tonsig, 2008).

A utilização de uma metodologia, mesmo que ainda não fortemente sedimentada, no desenvolvimento de software é de extrema importância, para que o sistema construído atenda as necessidades dos interessados, com um mínimo de qualidade. Para complementar a explicação será mostrada abaixo alguns dos ciclos de vida de desenvolvimento de software existentes.

3.2.1 **Modelo Sequencial Linear**

O Ciclo de Vida Clássico ou Modelo Cascata é o modelo mais antigo e mais utilizado na engenharia de software. Este é modelado em função do ciclo da engenharia convencional requer uma abordagem sistemática, sequencial ao desenvolvimento de software, conforme mostrado na Figura 3-1. Cada uma das fases desta figura representa etapas do processo de Desenvolvimento seguidos em ordem linear (Pressman, 2006).

Figura 3-1 – Modelo Sequencial Linear.

3.2.2 **Modelo de Prototipação**

O modelo de prototipação é o processo que possibilita que o desenvolvedor crie um modelo do software que deve ser construído. Idealmente, o modelo (protótipo) serve como um protótipo mecanismo para identificar os requisitos de software apropriado para quando o cliente definiu um conjunto de objetivos gerais para o software, mas não identificou requisitos de entrada, processamento e saída com detalhes, conforme mostrado na Figura 3-2(Pressman, 2006).

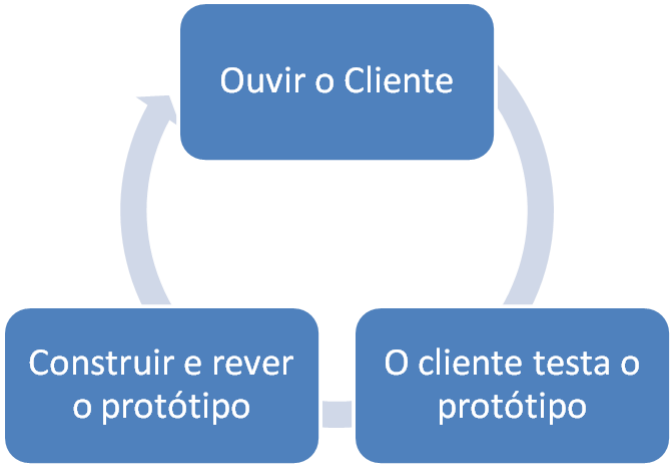

Figura 3-2 – Modelo de Prototipação

3.2.3 **Modelo Rapid Application Development (RAD)**

O modelo RAD é o modelo sequencial linear, mas que enfatiza um desenvolvimento extremamente rápido A “alta velocidade” é conseguida através de uma abordagem de construção baseada em componentes. Usado quando os requisitos são bem definidos e o escopo do sistema é restrito. Abordagem usada principalmente para aplicações de Sistemas de informação (Pressman, 2006).

Para que o processo consiga ser executado com uma maior velocidade, as tarefas são paralelizadas entre equipes, onde cada equipe fica responsável por um determinado módulo do sistema, como resultado final é possível agregar os módulos construídos e entregar em um tempo mais rápido ao cliente, conforme mostrado na Figura 3-3.

Figura 3-3 – Modelo RAD

3.2.4 **Modelo Incremental**

O modelo incremental combina elementos do Modelo Linear com a filosofia da Prototipação aplica sequências lineares numa abordagem de “saltos” à medida que o tempo progride Cada sequência linear produz um incremento do software (proc. de texto) O processo se repete até que um produto completo seja produzido Difere da Prototipação, pois a cada incremento produz uma versão operacional do software, conforme mostrado na Figura 3-4 (Sommerville, 2007).

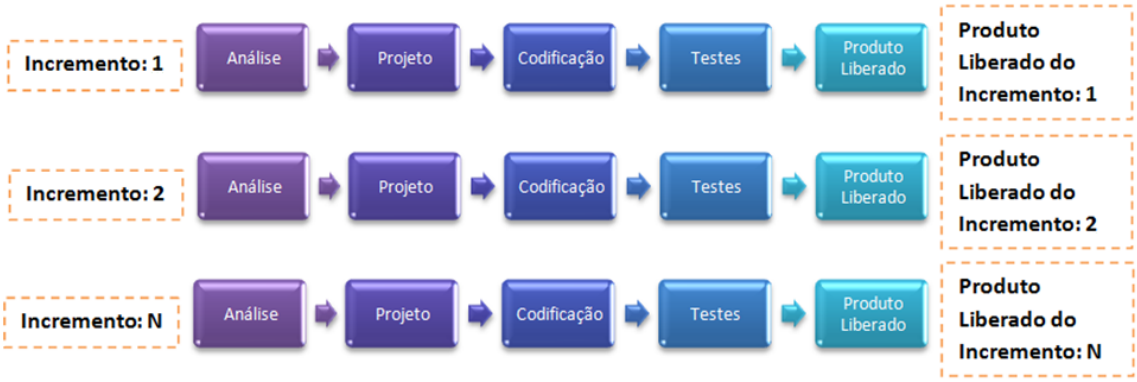

Figura 3-4 – Modelo Incremental

3.2.5 **Modelo Espiral**

O modelo espiral combina elementos do Modelo Linear com a filosofia da Prototipação aplica sequências lineares numa abordagem de "saltos" à medida que o tempo progride. Cada sequência linear produz um incremento do software (proc. de texto) O processo se repete até que um produto completo seja produzido difere da prototipação, pois a cada incremento produz uma versão operacional do software, conforme mostrado na Figura 3-5 (Sommerville, 2007).

Figura 3-5 – Modelo Espiral

3.2.6 **Montagem de Componente**

Desenvolvimento baseado em componentes ou *component-based-development* (CBD) também é conhecido como *component-based-software-engineering* (CBSE) ou simplesmente componente de software, conforme apresentado na Figura 3-6 (Sommerville, 2007).

Pressman(2006) não define o que é um componente e restringe-se a dizer que o modelo de desenvolvimento baseado em componentes utiliza paradigma de orientação a objetos baseando-se em uma classe como código reutilizável, ou seja, o componente. Em orientação a objetos uma classe encapsula dados e algoritmos e este último também pode ser usado para manipular os dados (Pressman, 2006).

O modelo de montagem de componente incorpora características de tecnologias Orientadas a Objetos. No Modelo Espiral a atividade de Engenharia começa com a identificação de classes candidatas, se a classe existe, será reutilizada, se a classe não existe, ela será desenvolvida nos moldes do paradigma de orientação a objetos. Através desta abordagem uma biblioteca de classes é criada com as classes identificadas no desenvolvimento do software. A partir deste momento toda iteração da espiral deverá verificar o conteúdo da biblioteca que pode ser reutilizado ou identificar se novas classes devem ser inseridas na biblioteca para posterior reuso (Pressman, 2006).

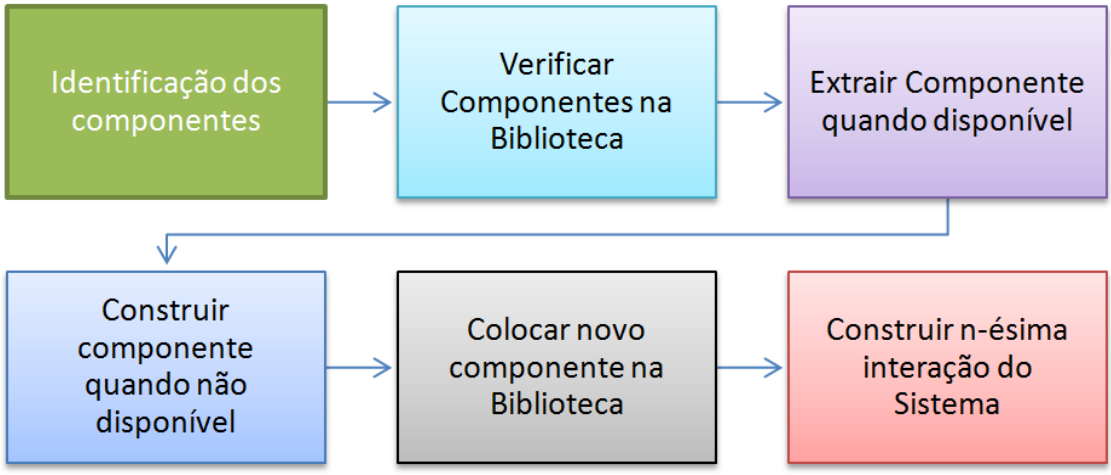

Figura 3-6 – Montagem de Componente

3.2.7 **Definição de Projeto do Sistema**

O projeto de um sistema é realizado pela equipe de Analistas de Sistemas do projeto. O objetivo é visualizar e predizer o comportamento do sistema antes de sua construção. Ao construir um sistema complexo, o desenvolvedor deve abstrair diferentes visões do sistema,

elaborar modelos através de uma notação precisa, verificar se esses modelos satisfazem os requisitos do sistema e transformar os modelos em implementação (Sommerville, 2007).

Para o projeto de um sistema são utilizadas diversas metodologias que envolvem princípios filosóficos que guiam uma gama de métodos que utilizam ferramentas e práticas diferenciadas para realizar algo.

Segundo Sommerville (2007) o projeto preliminar preocupa-se com a transformação do requisito numa arquitetura de dados de software. O Projeto detalhado concentra-se nos aprimoramentos da representação estrutural que levam representações algorítmicas e de estruturas de dados detalhados.

No contexto de um projeto preliminar e detalhado, são realizadas um conjunto de atividades, como por exemplo: o projeto de arquitetura, o projeto de banco de dados e o projeto de interface. As aplicações modernas, segundo Pressman, possuem uma atividade de projeto de interface distintas. O projeto de interface estabelece o layout e mecanismo de interação para a interação homem-máquina. Abaixo são citados aspectos relevantes para um bom projeto do sistema (Pressman, 2006).

- O software deve ser logicamente divido em componentes que executem funções e sub-funções especificas, pode chamar de software modular;
- Um projeto deve ter uma organização hierárquica que faça uso controle entre os componentes de software;
- Um projeto deve possuir métodos, procedimentos que apresentem características funcionais independentes;
- Um projeto deve conter uma representação distinta e separável dos dados e procedimentos;
- Um projeto deve levar uma interface que reduza a complexidade de conexões entre os módulos e com o ambiente externo;
- Um projeto deve ser derivado usando-se um método capaz de permitir repetição e que seja orientado pelas informações obtidas durante a análise de requisitos de software.

O processo de projeto de engenharia de software estimula o bom projeto por meio da aplicação de princípios fundamentais, metodologias sistemáticas e cuidadosa revisão. Vale ressaltar que as características citadas acima não são conseguidas casualmente. Desta forma, a Engenharia de Software aborda uma série de práticas e tecnologias, principalmente estudadas pela ciência da computação e áreas afins, enfocando seu impacto na produtividade e qualidade

de software. Como forma de auxiliar na criação de um projeto existem algumas técnicas, como (Pressman, 2006):

• **Análise Estruturada**: é uma atividade de construção de modelos. Utiliza uma notação que é própria ao método de análise estruturada com a finalidade de retratar o fluxo e o conteúdo das informações utilizadas pelo sistema, dividir o sistema em partições funcionais e comportamentais e descrever a essência daquilo que será construído (DeMarco, 1983).

• **Projeto Estruturado**: é a atividade da especificação das atividades que compõem um modelo funcional de transformação das necessidades do usuário. São provenientes das fases de análise e diagramação e de plano de implementação.

• **Programação Estruturada**: é uma forma de programação de computadores que preconiza que todos os programas possíveis podem ser reduzidos a apenas três estruturas: sequência, decisão e iteração, desenvolvida por Michael A. Jackson no seu livro "*Principles of Program Design*" de 1975 (Jackson, 1975 apud Pressman,2006) (Pressman, 2006).

• **Análise Essencial**: propõe a divisão do sistema por eventos. A rigor, o valor de um sistema está na sua capacidade de responder com eficácia a todos os estímulos a que for submetido. Assim, um sistema é construído para responder a estímulos. A cada estímulo, o sistema deve reagir produzindo uma resposta predeterminada. A expressão *Essential Analysis,* traduzida por Análise Essencial, foi proposta em 1984 por McMenamim e Palmerpara refletir a introdução dos novos conceitos que estavam sendo incorporados à Análise Estruturada clássica (Pressman, 2006).

• **Diagrama de Fluxo de Dados** (DFD): é uma ferramenta para a modelagem de sistemas. Ela fornece apenas uma visão do sistema, a visão estruturada das funções, ou seja, o fluxo dos dados. Caso se esteja desenvolvendo um sistema no qual os relacionamentos entre os dados sejam mais importantes que as funções, pode-se dar menos importância ao DFD e dedicar-se aos Diagramas de Entidade-Relacionamento (DER).

• **Modelo de Entidades e Relacionamentos** (MER): é um modelo abstrato cuja finalidade é descrever, de maneira conceitual, os dados a serem utilizados em um sistema de informações ou que pertencem a um domínio. A principal ferramenta do modelo é sua representação gráfica, o diagrama de entidade-relacionamento.

• ***Structured Analysis and Design Technique***(SADT): é uma técnica para apresentação de ideias. Esta técnica foi proposta por Douglas T. Ross em 1977, no artigo Ross, D. (1977) "*Structured Analysis (AS): A Language for Communicating Ideas*". *IEEE Transactionson Software Engineering*, vol. 3, no.1, pp. 16-34 (Sommerville, 2007).

Para o sucesso de um projeto de software é necessário realizar uma espécie de abstração do sistema de software através de modelos que o descrevem é um poderoso instrumento para o entendimento e comunicação do produto final que será desenvolvido (Souza, 1999).

A maior dificuldade nesta atividade está no equilíbrio entre simplicidade (favorecendo a comunicação) e a complexidade (favorecendo a precisão) do modelo. Para um melhor entendimento, é necessário explicar cada uma das fases do projeto do sistema, que são Projeto de Arquitetura do Sistema, Projeto de Interface do Sistema, Projeto de Dados(Pressman, 2006).

3.2.8 **Projeto de Arquitetura do Sistema**

O projeto da arquitetura do sistema conta com a participação da equipe de analistas do projeto, com o apoio de programadores e projetistas de banco de dados. É comum, também, a existência de um membro específico da equipe que assume o papel de arquiteto(Pfleeger, 2004).

O objetivo desta atividade é a definição dos mecanismos fundamentais que serão utilizados para o desenvolvimento do sistema.

A arquitetura definida deverá suportar de forma adequada os requisitos funcionais e não-funcionais levantados para o sistema. A definição dos mecanismos da arquitetura inclui decisões de projeto a respeito de como o sistema será implementado em termos de(Pfleeger, 2004):

- Linguagens de programação
- Tecnologias/plataformas utilizadas
- Métodos, componentes e ferramentas para acesso e recuperação de dados
- Distribuição e comunicação entre componentes e aplicativos
- Infraestrutura de hardware e software *Log* e tratamento de erros
- Segurança Padrões e convenções

A arquitetura deverá ser descrita a partir do Documento de Arquitetura que deve conter de acordo o Quadro 3-1, além das descrições dos mecanismos, os critérios que motivaram as decisões tomadas para o projeto do sistema. Além das descrições dos mecanismos fundamentais, a arquitetura pode ser representada através de diagramas demonstrando as estruturas estáticas e dinâmicas do sistema e, principalmente, sua distribuição física e lógica (Sommerville, 2007).

Abaixo um exemplo baseado na UML de como definir uma arquitetura de um sistema Estas representações podem ser elaboradas utilizando diagramas da UML (pacotes, classes, sequência, distribuição, componentes etc.) (Booch, 2006).

Quadro 3-1 - Passos para definição de uma arquitetura de software

Documentos de Entradas	
• Documento de Consenso do Produto • Modelo de Casos de Uso (Diagrama de Casos de Uso) • Matriz de Requisitos	
Passos	**Descrição**
1	Analisar os requisitos funcionais e não-funcionais levantados para o sistema com base na Matriz de Requisitos, no Diagrama de Casos de Uso e no Documento de Consenso do Produto
2	Projetar os mecanismos fundamentais necessários para que a arquitetura projetada suporte de forma adequada os requisitos do sistema
3	Avaliar a viabilidade de implementação dos mecanismos definidos
4	Elaborar representações gráficas da arquitetura e de seus mecanismos
5	Registrar as decisões de projeto, os mecanismos definidos e a as representações gráficas elaboradas no Documento de Arquitetura
6	Revisar o Documento de Arquitetura junto à área de Homologação em conformidade com o Plano de Homologação do Sistema
7	Revisar, em conjunto com representantes do usuário gestor e usuários finais, a arquitetura proposta para o sistema
Documentos de Saída	
• Documento de Arquitetura	

3.2.9 **Projeto de Interface do Sistema**

A definição de interface segundo o autor Moran é "a interface de usuário deve ser entendida como sendo a parte de um sistema computacional com a qual uma pessoa entra em contato físico, perceptiva e conceitualmente". A interface viabiliza a iteração usuário e sistema (Moran, 1981).

A definição do projeto de interface caracteriza uma perspectiva para a interface de usuário como tendo um componente físico, que o usuário tem um contato físico e visual, e outro conceitual, que o usuário interpreta, processa e raciocina. Moran e outros autores denominam este componente de modelo conceitual do usuário (Moran, 1981).

O usuário realiza o contato com o sistema através da interface, ou seja, viabiliza e facilitar os processos de comunicação entre o usuário e a aplicação, pode se chamar também

de interface homem-máquina. A interface é parte de um artefato que permite o usuário realizar o controle e avaliar o funcionamento deste artefato, está avaliação pode ser feita através de dispositivos sensíveis às suas ações e capazes de estimular sua percepção (Pressman, 2006).

Segundo Norman a interface de sistemas computacionais é diferente das interfaces de maquinas convencionais, já que exigem do usuário um raciocínio maior durante as atividades de interpretação e expressão das informações que o sistema processa Através da interface é que os usuários têm acesso às funções da aplicação (Norman, 1986)

O projeto de interface pode ser construído através de um processo iterativo de construção e avaliação do protótipo baseados em princípios e diretrizes empíricas. Todavia, estes princípios podem ser conflitantes em determinadas situações. Como forma de resolver os problemas é necessário basear a prática de design de interfaces em uma fundamentação teórica. Esta fundamentação irá servir como apoio para que o designer ao longo da elaboração da sua solução particular para o conjunto de problemas que a aplicação pretende resolver (Pressman, 2006).

Vários estudos sobre a área de Interação Humano–Computador (IHC) têm como principal objetivo fornecer aos desenvolvedores de sistemas e pesquisadores explicações e previsões para fenômenos de interação usuário-sistema e resultados práticos para o design da interface de usuário (Pressman, 2006).

Os estudos realizados na área de IHC, ajudam a prever antecipadamente se o sistema a ser desenvolvido satisfaz as necessidades de usabilidade, comunicabilidade e aplicabilidade dos usuários. Com isto, estudos de IHC têm o objetivo de desenvolver modelos teóricos de desempenho e cognição humanos, bem como técnicas efetivas para avaliar a usabilidade (Sommerville, 2007).

Nos estudos realizados em IHC também percebe-se a importância que além de usabilidade, as aplicações devem buscar atingir aplicabilidade e comunicabilidade(Souza, 1999), oferecendo ao usuário artefatos fáceis de usar, aplicar e comunicar.

No contexto de IHC devemos considerar quatro elementos básicos:

1. Sistema;
2. Desenvolvedores de software;
3. Usuários;
4. Ambiente de uso.

Os elementos citados acima estão envolvidos em dois processos: o primeiro é a interação entre o usuário e o sistema, o outro processo é o desenvolvimento deste sistema. Para melhoria na aplicação dos processos de desenvolvimento e de interação usuário-sistema existem diferentes disciplinas que proporcionam estudos teóricos que podem ser aplicados ao desenvolvimento, são eles:

• **Design e Desenvolvimento do hardware e software**: para que o produto saia com uma qualidade aceitável, são realizados estudos de tecnologias de dispositivos de entrada e saída; e tecnologias de software, como ambientes gráficos e virtuais.

• **Estudo da capacidade e limitação física e cognitiva dos usuários**: estudos de ergonomia para avaliar limites de esforço físico do usuário, e estudos de psicologia e ciência cognitiva sobre a capacidade humana de memorização, raciocínio e aprendizado.

• **Instrumentação teórica e prática para o design e desenvolvimento de sistemas interativos**: envolve o conhecimento teórico a respeito dos fenômenos envolvidos; modelos para o processo de desenvolvimento que descrevam as etapas necessárias e como devem ser conduzidas; diretrizes, técnicas, linguagens, formalismos e ferramentas de apoio a estas etapas (Souza, 1999).

• **Modelos de interfaces e do processo de interação usuário–sistema**: para desenvolver modelos abstratos do processo de interação compatíveis com as capacidades e limitações físicas e cognitivas dos usuários (Souza, 1999).

Usabilidade

Dentro da área de IHC uma parte importante é a usabilidade do sistema, ou seja, já que a interface é o meio de comunicação entre o usuário e o sistema está deve ser de fácil intuição para o usuário que irá utilizar a aplicação. O conceito de usabilidade diz que à qualidade da interação de sistemas com os usuários e depende de vários aspectos. Alguns destes fatores são:

• Facilidade de aprendizado do sistema: qual o tempo e o esforço necessários para que os usuários alcancem um determinado nível de desempenho;

• Facilidade de uso: tem com objetivo avaliar o esforço físico e cognitivo do usuário durante o processo de interação, medindo a velocidade de e o número de erros cometidos durante a execução de uma determinada tarefa;

• Satisfação do usuário: realizar pesquisas com o usuário para saber se ele a interface atende as demandas dele, se ele está confortável ao utilizar o sistema e coletar as sugestões para melhorar a interface;

• Produtividade: avaliar se a ferramenta trouxe uma produtividade maior do que ele já tinha antes

3.2.10 **Projeto de Dados**

O projeto de dados é a primeira (e diriam alguns, a mais importante) dentre três atividades de projeto que são realizadas durante a engenharia de software(Tonsig, 2008). O impacto na estrutura de dados sobre a estrutura do programa e sobre a complexidade procedimental faz com que o projeto de dados tenha uma profunda influência sobre a qualidade do software.

Independentemente das técnicas a serem usadas, dados bem projetados podem levar uma melhor estrutura do programa, à efetiva modularização e a reduzida complexidade procedimental(Tonsig, 2008).

O autor Wasserman propôs um conjunto de princípios que podem ser usado para se especificar e projetar dados. Na realidade, o projeto de dados frequentemente é considerado como uma parte da tarefa de análise de requisitos. Relembrando que a análise de requisitos e a atividade do projeto muitas vezes se sobrepõem, consideremos o seguinte conjunto de princípios para a especificação de dados (Wasserman, 1980 apud Sommerville, 2007):

1 - Os princípios de analise sistemática aplicados à função e ao comportamento também dever ser aplicado aos dados;

2 - Todas as estruturas de dados e as operações a serem executadas em cada uma devem ser identificadas;

3 - Um dicionário de dados deve ser estabelecido e usado para se definir tanto o projeto de programa como o de dado;

4 - Decisões de baixa prioridade referentes ao projeto de dados devem ser proteladas até mais tarde no processo de projeto;

5 - A representação da estrutura de dados deve ser conhecida somente por aqueles módulos que precisam fazer uso direto dos dados contidos na estrutura;

6 - Uma biblioteca de estrutura de dados úteis e das operações que podem ser aplicações a elas deve ser desenvolvida;

7 - Uma linguagem de programação e projeto de software deve suportar a especificação e a realização de tipos abstratos de dados;

Os princípios acima descritos formam a base para uma abordagem ao projeto de dados que pode ser integrada tanto na fase de definição como na etapa de desenvolvimento do

processo de engenharia de software. Conforme observamos em outra parte deste livro, uma definição clara das informações é essencial a uma desenvolvimento de software bem-sucedido (Sommerville, 2007).

3.2.11 **Considerações Finais**

Este capítulo abordou a importância do Processo de Projeto e Construção de um produto, onde, foi observado que um projeto bem elaborado pode trazer resultados positivos. Neste capítulo também foi demonstrado como se trata a construção de um produto de software mostrando alguns modelos de desenvolvimento, assim como os ciclos de vida do desenvolvimento de Software. Após a abordagem mais conceitual foi mostrando algumas boas práticas, para realizar a criação de um projeto de Arquitetura, interface e de dados de forma mais prática.

Capítulo 4 - **Melhoria no Processo de Projeto e Construção do Produto**

As empresas que produzem software buscam a melhoria do processo de software com o objetivo de aumentar a competitividade, baseado em técnicas da Engenharia de Software. A definição e melhoria de um processo pode ser definida de acordo com a necessidade de uma determinada organização, ou seja, seus processos de PCP podem ser instanciados para um determinado cenário (Filho, 2007).

Diante deste contexto, as empresas realizam a contratação de colaboradores que possam definir, realizar melhorias e manter o processo de software dela, além de aprimorar seus processos principais de negócio e de apoio, normalmente, tendo como base modelos e normas nacionais e/ou internacionais de qualidade (Ebert, 2006).

Esta dissertação apresenta como realizar a definição de um processo de PCP, utilizando o BPMN (Miers, 2008) como notação e também a fazer uma análise de adequação do processo, além das formas para realizar uma avaliação no processo de PCP.

Neste capítulo será apresentada na seção 4.1, a contextualização do cenário que o processo de PCP será avaliado, onde também será mostrado o processo de criação da Fábrica de Software. Na Seção 4.2 será apresentada a primeira versão do Processo de PCP, nomeada de PCP Versão 1. Em seguida, na Seção 4.3 será abordada a Avaliação para melhoria do PCP Versão 1, onde serão mostrados os pontos fortes, pontos fracos e oportunidades de melhorias (MELO, 2010). Já a seção 4.5 aborda o processo evoluído da Versão 1, chamado de Processo de PCP Versão 2, o qual é uma evolução do primeiro processo. Na Seção 4.6 será apresentada uma análise das melhorias do PCP Versão 2.

4.1 Contextualização do cenário do projeto

Com o objetivo de compreender o processo que foi utilizado na Fábrica de Software, faz-se, primeiramente, necessária a contextualização do cenário do projeto, conforme apresentado na seção 4.1. Para tal, esta seção está divida em três subseções. A primeira descreve as características e informações do cliente atendido, a segunda apresenta as

informações do fornecedor e a última subseção ambienta sobre o cenário do projeto que foi contratado.

4.1.1 **Cliente**

De acordo com o decreto-lei de número 4.451, em 9 de julho de 1942, foi fundado através de um acordo entre Brasil e os Estados Unidos. E após oito anos, foi criado o, então, Banco *BETA*. Para o seu crescimento foram disponibilizadas, através de incentivos do Governo Federal, dezenas de linhas de crédito para financiamento. As linhas de financiamento de crédito são voltadas para o desenvolvimento sustentável da Amazônia Legal (Melo, 2010).

O Banco *BETA* possui dentre seus produtos, linhas de financiamento em longo prazo com taxas a valores acessíveis em relação aos outros bancos. O objetivo do Banco é "prover o desenvolvimento da região amazônica desempenhando um papel importante tanto no âmbito da pesquisa, quanto no crédito de fomento".

O crédito de fomento é um dos principais produtos do banco. A maior parte desta renda é responsável pelo seu funcionamento, entretanto, devido a problemas encontrados no sistema legado e a necessidade de atualização nos sistemas e máquinas, fez-se necessário atualizar o parque tecnológico, visando atender novas demandas e melhorar a qualidade dos serviços prestados.

Para realizar esta atualização do parque tecnológico, o Banco *BETA* criou o Programa de Excelência Tecnológica (PET), que tem como objetivo de acompanhar a evolução dos projetos. O PET tem o objetivo organizar e atualizar estruturas de hardware e software do banco. Dentre os projetos de software, está o Sistema de Gestão de Fomento, foco deste trabalho. Para realizar esta atualização foi necessário a contratação de uma empresa de desenvolvimento de soluções bancárias.

Na próxima subseção serão apresentadas informações sobre o fornecedor contratado para prover a atualização tecnológica no banco.

4.1.2 **Fornecedor**

Em 18 de julho de 1974, através de um acordo entre a E.E. Eletrônica, o BNDES (Banco Nacional do Desenvolvimento) e a inglesa Ferranti, foi fundada a Empresa *ALFA*. A empresa *ALFA* foi a pioneira na produção de tecnologia nacional para computadores, absorvendo tecnologia do SERPRO (Serviço Federal de Processamento de Dados). A empresa

foi responsável pela criação da linguagem LTD (Linguagem e Tecnologia Digital) para seus computadores.

Nos anos 90, um grande Banco Brasileiro adquiriu as ações da empresa *ALFA*, tornando-se a integradora de soluções, como: comercializar computadores, implantar, treinar e prover assistência técnica em todo o país para os equipamentos das maiores empresas desenvolvimento mundial. Em 2003, a empresa *ALFA* passa a atuar no mercado exterior, além de se tornar a maior empresa do segmento no mercado nacional.

Em 2004, dada a necessidade de atualização do parque tecnológico, conforme explicado na subseção 4.2.1, o Banco *BETA*, realiza a contratação da empresa *ALFA*.

A empresa *ALFA* até o ano de 2007 não possuía experiência na construção de software. Entretanto, para desenvolver o maior projeto do PET, a empresa *ALFA* optou por não terceirizar a construção do sistema de Gestão de Fomento e criou o projeto de implantação de uma Fábrica de Software, objeto deste estudo.

Inicialmente, para implementar a Fábrica de Software foi necessário a contratação de 3 Gerentes de Projeto para verificar a viabilidade do projeto, o qual consistia no levantamento do escopo, assim como a criação do cronograma e do processo de desenvolvimento da Fábrica, o qual será apresentado na seção 4.2. Posteriormente à definição do processo, foi necessário realizar a contratação de mão-de-obra qualificada para iniciar a construção do produto.A contratação inicial foi de 3 (três) desenvolvedores, 1 (um) arquiteto de software e 1 (um) analista de requisitos (Melo,2010).

4.1.3 **Projeto**

O PET possuía vários sistemas a serem desenvolvidos, como os sistemas de ERP (*Enterprise Resource Planning*), o *Internet Banking*, a integração do sistema da Organização aos sistemas externos (BACEN, SERASA, Sistema da Receita Federal (SRF)), sistemas de conta corrente, entre outros. Entretanto, como foco deste trabalho será apresentado o Sistema de Gestão de Fomento (SGF), já que o processo de software tratado neste trabalho foi usado para atender esta demanda. O SGF consiste na administração de recursos financeiros de fundos externos e sua utilização por meio de contratos firmados entre o Banco e os proponentes.

Inicialmente, os Gerentes de Projeto realizaram um estudo para verificar o tamanho estimado do SGF. Para realizar a estimativa foi utilizada a métrica Pontos de Caso de Uso (UCP). O projeto foi estimado em 4.560 UCPs.

A partir da coleta de UCPs, os Gerentes de Projeto decidiram dividir o escopo do projeto em quatro grandes entregas. Nestas entregas estava contido um conjunto de funcionalidades que agregavam valor de negócio ao cliente (Melo, 2010).

A primeira entrega do sistema tinha como funcionalidade realizar o acolhimento da Proposta para financiar um projeto industrial ou agrícola de um cliente do banco. A segunda entrega tinha como objetivo realizar uma análise para verificar a viabilidade da proposta emitida pelo cliente. Esta análise é composta através de verificações técnicas com analistas especializados. Após a avaliação, esta proposta é submetida à aprovação de um comitê. Por sua vez, este comitê realiza votação da proposta para aprovação ou não.

Após a aprovação da proposta do cliente, a terceira entrega tinha o objetivo de gerar um contrato para o cliente. Na quarta entrega são realizados os cálculos referentes a liberação do dinheiro ao cliente, e o reembolso através de pagamento feitos pelo cliente.

A seguir, a seção 4.2 apresenta o processo de software usado na Fábrica de Software utilizado no início do desenvolvimento do projeto.

4.2 Processo de Projeto e Construção do Produto Versão 1

Para iniciar o desenvolvimento do SGF, os Gerentes de Projeto realizaram a definição do Processo de Desenvolvimento de Software do Produto, entretanto o foco deste trabalho se limita apenas a área de Projeto e Construção de Produto. Como forma de melhorar o entendimento do trabalho, o processo de PCP definido inicialmente pelos Gerentes do Projeto será chamado de PCP Versão 1, conforme ilustrado na Figura 4-1 (Melo, 2010).

A primeira atividade dos gerentes foi conhecer o escopo completo do projeto, bem como o seu entendimento de uma forma geral, sem um profundo detalhamento de cada funcionalidade do sistema.

Para iniciar a definição do processo de PCP na Fábrica de Software foi utilizada a notação *Business Process Modeling Notation (BPMN)* (Miers, 2008). A escolha desta notação foi uma opção dos gerentes, por se tratar de uma linguagem intuitiva e clara quanto a expressividade das notações.

A primeira etapa do processo era composta pelas atividades de Análise de Requisitos, Construção do Caso de Uso e Elaboração da Regra de Negócio, as quais consistem no primeiro contato com o cliente. O resultado era a geração de artefatos para a equipe de desenvolvimento, com o foco na codificação do produto.

Após a codificação, foram realizados testes internos pela própria equipe de desenvolvimento e, posteriormente, liberada uma versão da aplicação para a equipe de testes. A equipe de testes realizava procedimentos a procura de defeitos e/ou melhorias na aplicação. Todos os defeitos e/ou melhorias detectados eram relatados na ferramenta de *bug tracking* e enviados para a equipe de desenvolvimento realizar a devida análise e posterior correção.

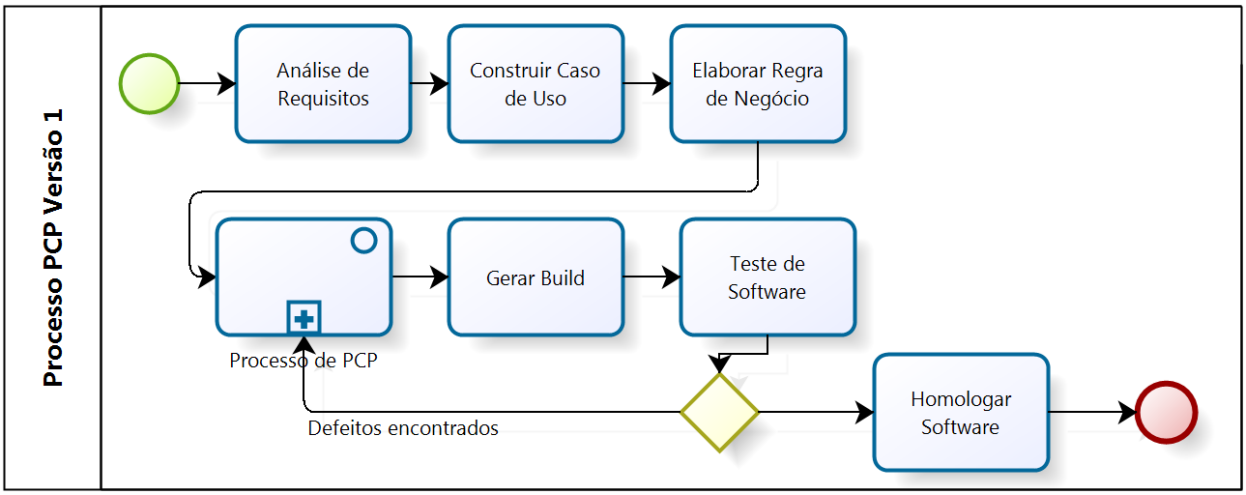

Figura 4-1 – Processo de desenvolvimento de software Versão 1

A equipe de desenvolvimento realizavam a correção e gerava uma nova versão da aplicação para ser novamente testada pela equipe de teste.

Posteriormente à fase de testes, o produto estavam pronto para ser avaliado pelo cliente. Esta atividade era chamada de Homologação. O objetivo desta atividade é receber o aceite do cliente, contendo aprovação, rejeição ou aprovação com ressalva, baseado no SLA (*Service Level Agreement* – Acordo de Nível de Serviço) definido em contrato, o qual define a criticidade dos *bugs* como:

- **Trivial**: Defeitos que não impediam o fluxo de negócio da aplicação;
- **Importante**: Defeitos encontrados que poderiam ter algum comportamento errôneo, entretanto, sem impedir o fluxo de negócio;
- **Crítico**: Defeito que impedia o correto funcionamento do sistema.
- **Mudanças**: Quando uma determinada funcionalidade não foi implementada da maneira como o cliente desejava.
- **Melhoria**: Quando uma funcionalidade precisa de algum ajuste, para facilitar sua utilização por parte do cliente.

Para que a homologação fosse aceita, o sistema não poderia ter nenhum erro crítico, no máximo 25% de defeitos importantes. Durante a fase de Homologação, o cliente registrava todos os defeitos, solicitação de mudança e melhorias encontrados. Este registro era feito em uma ferramenta de *bug tracking*. Todas as solicitações eram avaliadas pela equipe de Gerentes do Projeto e equipe técnica para verificar a viabilidade da implementação ou correção dos defeitos.

O PCP Versão 1 foi apresentado de forma geral. Para um melhor entendimento da definição do processo é necessário conhecer os detalhes da fase de projeto e construção do produto de software, foco deste trabalho, como pode ser vista na Figura 4-2.

A atividade inicial do processo PCP era Analisar e Projetar a Arquitetura do software, baseando-se no que foi definido como insumos gerados pela equipe de Requisitos. Nesta atividade eram realizadas reuniões entre os arquitetos do projeto e os engenheiros de software para modelarem a arquitetura do sistema, bem como a utilização de *frameworks* para apoiar o desenvolvimento (Melo, 2010).

Depois da arquitetura de software definida, a próxima atividade era o projeto de Banco de Dados, que tinha o objetivo de projetar o modelo das tabelas de acordo com os casos de uso gerados pela equipe de requisitos. Posteriormente, a equipe de banco de dados realizava uma validação do diagrama de Banco de Dados e do Dicionário de Dados com a equipe de Arquitetura do projeto. Os documentos de diagrama de Banco de Dados e o Dicionário tinham a finalidade de apoiar a equipe de Desenvolvimento na realização de suas atividades (Melo, 2010).

De posse do diagrama de banco de dados validado, iniciava-se a implementação do mesmo, onde eram criadas todas as tabelas, atributos, relacionamentos. Adicionalmente, a equipe de Banco de Dados iniciava o projeto de Migração dos Dados a partir do sistema legado do cliente. Os dados migrados contribuíam para realização dos testes no sistema.

A atividade de Codificar o requisito funcional era executada logo após o projeto de banco dados. A equipe de desenvolvimento utilizava os documentos gerados pela equipe de requisitos, para iniciar a codificação. Como forma de apoio à codificação, os arquitetos apresentavam a arquitetura do sistema e os *frameworks* utilizados. Após a codificação eram realizados os testes de unidade, que tinham o objetivo de testar funcionalmente unidades de código para validação do próprio Engenheiro de Software.

Após a realização de todas as atividades, era gerada uma versão do sistema para prosseguir com os testes, conforme demonstrado na Figura 4-2.

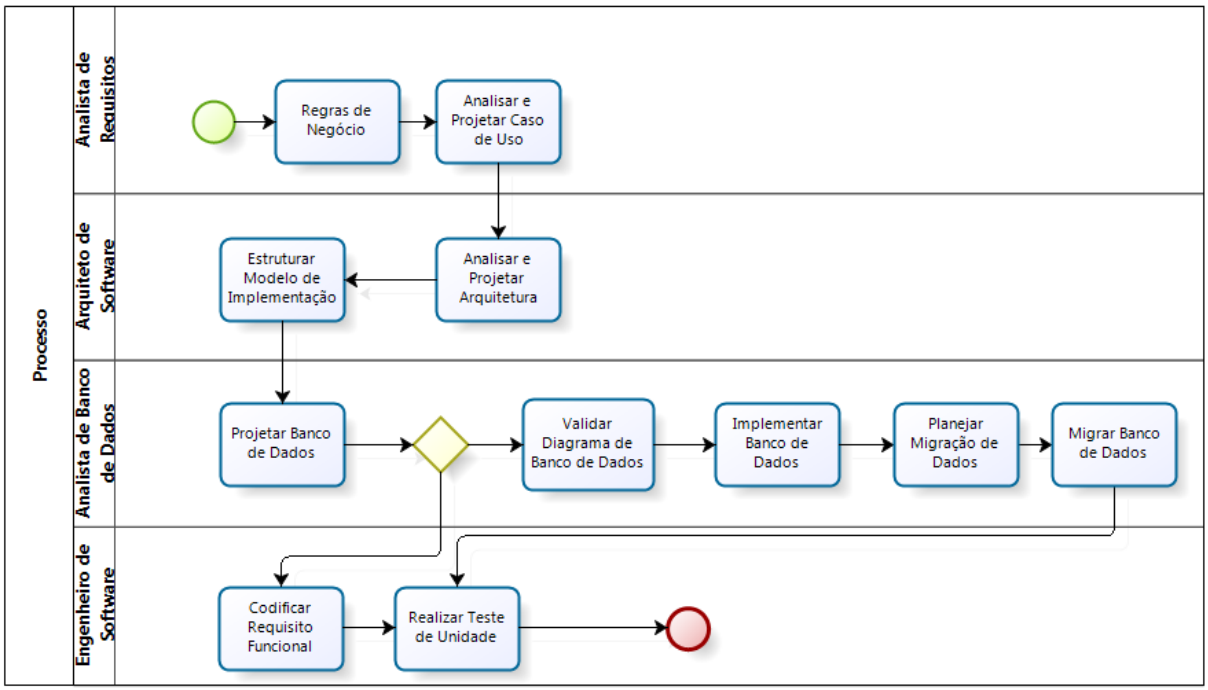

Figura 4-2 – Processo de Projeto e Construção de Produto Versão 1.

A elaboração dos artefatos gerados pela equipe de requisitos estava sendo executado normalmente, entretanto, devido ao atraso para conseguir validar esses artefatos com o cliente, houve um desvio na execução do processo de desenvolvimento. Os documentos gerados pela equipe de requisitos eram liberados para a fábrica de código, sem que ao menos tivessem passado pela validação do cliente, conforme ilustrado na Figura 4-3. Este desvio ocorria como o objetivo de atender as demandas do projeto e evitar atrasos na entrega do sistema (Melo, 2010).

Esses artefatos eram liberados para que a equipe de arquitetos realizasse uma análise dos poucos dados que se tinham levantado através de conversa com os analistas de requisito, para elaboração de uma arquitetura de software. A forma de execução das atividades eram orientadas pelos gerentes de projeto da Fábrica.

A arquitetura era definida e, então, repassada para a equipe de desenvolvimento e de banco de dados iniciar o projeto e implementação. A equipe de desenvolvimento possuía uma atividade de Teste de Unidade, que não era seguida, devido a urgência para entregar em um curto período uma nova versão do sistema para o cliente validar.

Por sua vez, a equipe de banco de dados iniciava a atividade de migração dos dados do sistema legado para o sistema que estava sendo desenvolvido. Após a construção do produto, uma versão era entregue para a equipe de testes.

Os impactos da execução deste segundo processo afetavam a qualidade do produto, já que quando uma versão era submetida para fase de homologação o cliente rejeitava alegando que não estava de acordo com o que foi solicitado.

Na equipe de desenvolvimento o impacto era grande também, cada defeito relatado em homologação, em sua grande maioria, ocasionavam retrabalho, pois era necessário descartar a implementação inicial do caso de uso pela desconformidade do negócio do cliente.

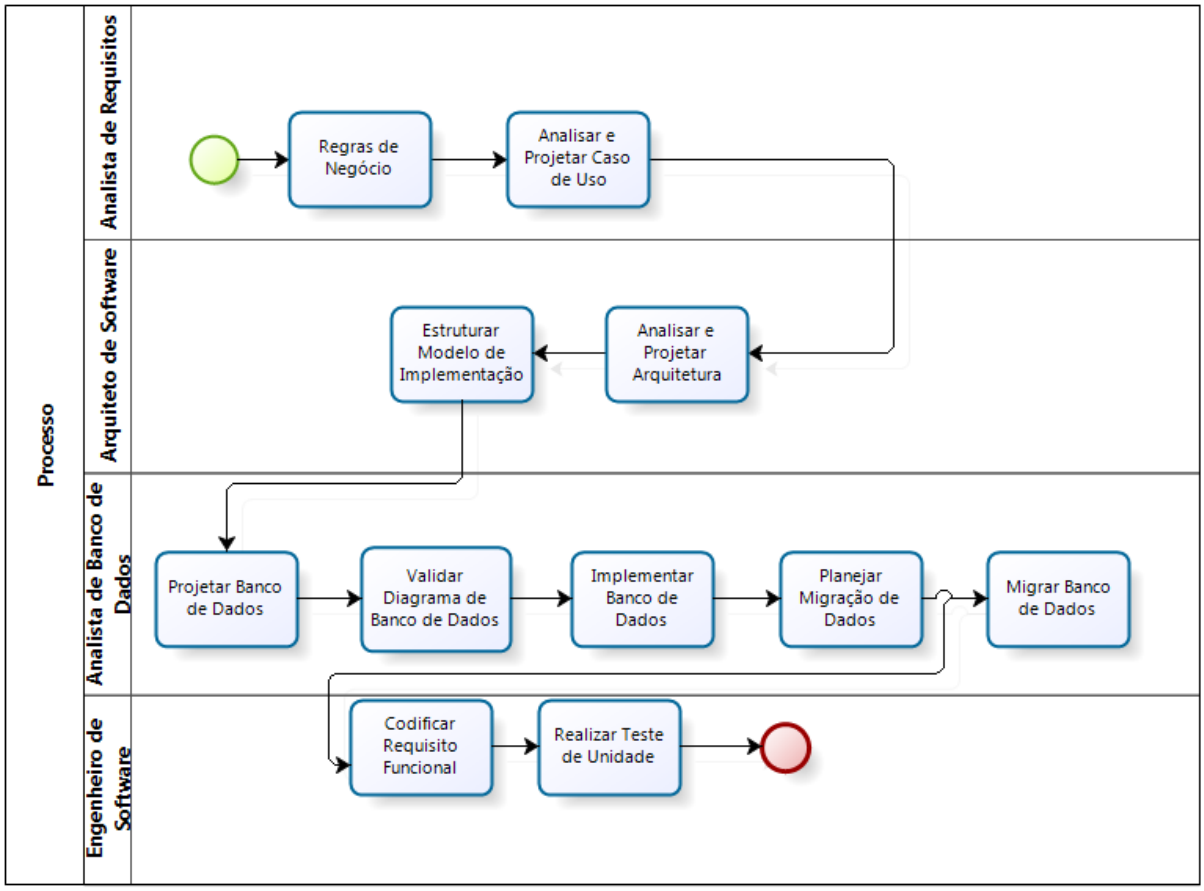

Figura 4-3 – Desvio da execução do Processo PCP Versão 1.

Após um levantamento inicial do cenário do projeto, foi possível verificar que não existia a unificação e a padronização dos processos definidos. O resultado era um sistema que não atendia a necessidade do cliente, sendo necessário assim reconstruir grande parte do que tinha sido construído.

O primeiro problema encontrado foi na atividade de criação dos documentos de Caso de Uso e de Regras de Negócio, onde estes artefatos não eram mais validados pelo cliente.

Outro problema encontrado foi o início da atividade de arquitetura do sistema, sem que os requisitos tivessem sido validados.Com isso os arquitetos definiam uma arquitetura

pouco consistente e não planejada. Qualquer mudança no sistema gerava impacto, que aumentava a possibilidade de reestruturações de código ou *refactoring* nos módulos seguintes.

O outro ponto negativo encontrado era a não criação de um plano para a realização da prova de conceito das linguagens de programação e frameworks, a qual consiste em testar as tecnologias existentes para analisar quais se adaptam à solução de um determinado problema.

Outro problema, não menos importante, é a falta de padronização das nomenclaturas das tabelas do banco de dados, assim como dos atributos contidos em cada tabela, o que gerava dúvidas de entendimento entre os membros da equipe de desenvolvimento.

Após uma análise realizada na fábrica, foi gerado um relatório que continha os pontos fortes, pontos fracos e oportunidades de melhoria do projeto. Esses pontos serão expostos e analisado na seção 4.3.

4.2.1 **Papéis utilizados no processo de PCP**

Como complemento do processo de PCP descrito anteriormente, é importante apresentar os papéis que eram envolvidos neste processo. Papéis, neste caso, também, podem ser considerados como função que uma pessoa desempenha na empresa.

Para elucidar a função de cada papel, dividiu-se os papéis em duas partes. A primeira apresenta os papéis que apoiam, conforme o Quadro 4-1 e a segunda os papéis que realmente fazem parte do processo de PCP, conforme o Quadro 4-2.

Para definição dos papéis, foi realizada a identificação das habilidades necessárias para cada papel da equipe de desenvolvimento do projeto, as quais apoiaram as provas de conhecimento e entrevistas para a seleção dos recursos humanos.

- **Papéis de apoio:** são os papéis que apoiam as atividades do PCP, mas não as realizam. São eles:

Quadro 4-1 - Papéis de apoio ao PCP versão 1

Papel	**Descrição**
Analista de Requisitos	Responsável por gerar artefatos de Casos de Uso, Regras de Negócio e modelagem UML. Tem como objetivo transferir o conhecimento do negócio para a área de implementação.
Analista de Testes	Responsável por verificar se o produto está funcionando corretamente.

Papel	Descrição
Analista Documentador	Responsável por elaborar a documentação técnica e funcional dos produtos da empresa (manuais de utilização/administração, etc.).
Gerente de Projeto	Responsável por realizar a atividade de homologação junto ao cliente.

- **Papéis do PCP**: Estão presentes diretamente no fluxo das atividades do PCP. São eles:

Quadro 4-2 - Papéis do PCP versão 1

Papel	Descrição
Analista de Dados	Responsável por atuar no projeto e análise do Modelo de Dados Lógico e Físico do sistema.
Arquiteto de Software	Responsável por especificar os padrões e metodologias que deverão ser utilizados pelo programador para implementar as regras de negócio, de forma a atender aos requisitos da empresa.
Desenvolvedor	Papel responsável por codificar e testar, de forma unitária, programas de computador, estabelecendo os processos operacionais necessários para o tratamento dos dados, baseando-se nas definições fornecidas na fase de análise de sistemas e valendo-se de métodos e técnicas adequadas aos equipamentos e aplicações a que se destinam.

4.2.2 **Artefatos do PCP**

Nesta seção serão descritos os artefatos produzidos no processo de PCP. O Quadro 4-3 apresenta os artefatos gerados pelas áreas de apoio ao PCP e o Quadro 4-4apresenta os artefatos gerados pela execução da fase do PCP propriamente dita.

Quadro 4-3 - Artefatos de apoio ao PCP versão 1

Artefato	**Descrição**
Regra de Negócio	Tem o objetivo de definir o comportamento do sistema, como as regras de negócio de uma determinada ação.
Caso de Uso	Tem como objetivo prover a interação entre um ator e o sistema ou ainda entre sistemas, ou entre hardware e sistema, mostrando as mensagens e o comportamento desta funcionalidade.
Glossário	Tem como objetivo definir todos os termos em comum utilizados no projeto. É equivalente a um dicionário, onde contém informações de conceitos, notações utilizadas, padrões.

Quadro 4-4– Artefatos do PCP versão 1

Artefato	**Descrição**
Ata de Reunião de Viabilidade do Projeto	Tem como objetivo conter os relatos de todas as manifestações dos envolvidos em uma reunião técnica para verificar a viabilidade de implementação de um requisito, de acordo com o cronograma do projeto.
Scripts de criação de estruturas de banco de dados e Dicionário de Dados	Tem como objetivo documentar e implementar o Modelo de Banco de Dados do Sistema.
Plano de Arquitetura	Tem como objetivo apresentar de forma geral a arquitetura do sistema.
Modelo de Análise e Projeto	Tem como objetivo verificar possíveis inconsistências de requisitos e casos de uso. Neste artefato também é possível verificar as dependências entre partes do sistema.

4.3 Avaliação para Melhorias do processo de PCP versão 1

Durante as reuniões realizadas pelo SEPG (*Software Engineering Process Group*), foi possível definir através de auditorias no repositório de ativos do projeto informações sobre o

andamento do processo de software definido. O objetivo era conhecer os pontos em que a organização precisa de melhoria, e a partir de então gerar um plano de ação para tratá-los (Andrade, 2005).

Esta seção apresenta os Pontos Fortes, Pontos Fracos e Oportunidades de melhorias identificadas a partir da análise do processo de PCP versão 1.

- **Pontos Fortes**
  - **Planejamento da Criação da Arquitetura do Sistema:** A equipe de arquitetura criava e atualizava o documento de Plano de Arquitetura, contemplando as informações pertinentes sobre a mesma. Este documento tem como objetivo apoiar tecnicamente as atividades de desenvolvimento;
  - **Planejamento na criação do Banco de dados:** O banco de dados foi criado após uma análise da equipe de arquitetura juntamente a equipe de análise de dados de acordo com o documento de caso de uso, o qual é elaborado pela equipe de requisitos.
- **Pontos Fracos**
  - **Necessidade de uma Unificação dos Processos**: O processo foi executado de maneira diferente do que havia sido definido e institucionalizado;
  - **Execução da Prova de Conceito**: A equipe de arquitetura não realizava a prova de conceito, a qual permite testar as tecnologias e *framework* antes de gastar todo o tempo desenvolvendo-o uma funcionalidade.
  - **Falta da realização do teste unitário no código-fonte:** Mesmo estando presente no processo, a atividade de teste unitário precisa ser institucionalizada;
  - **Falta de Padronização no Banco de Dados:** A equipe de banco de dados não possuía um padronização da nomenclatura de tabelas, atributos, procedimentos, o resultado disto era a dificuldade para os desenvolvedores entenderem o modelo relacional do banco de dados. Desta forma atrasava o desenvolvimento do produto;
  - **Reutilização:** Conforme verificado em código-fonte foi constatado que a equipe de desenvolvimento não realizava técnicas para reaproveitamento de código-fonte. A desvantagem é que o desenvolvedor despendia um tempo para fazer uma método similar já desenvolvido, acarretando em um possível atrasado na entrega de uma determinada funcionalidade;

- **Oportunidades de melhorias:**
  - **Treinamentos de arquitetura, das tecnologias utilizadas em desenvolvimento e boas práticas de programação:** Recomenda-se realizar treinamentos periódicos das tecnologias, de boas práticas de programação e da arquitetura utilizada. O objetivo é melhorar o entendimento da equipe de desenvolvimento sobre os *frameworks* e a arquitetura do sistema, aumentando assim, a autonomia da mesma.
  - **Documentar o código-fonte**: Institucionalizar para toda a equipe de desenvolvimento a necessidade de escrever comentários no código fonte, pois o objetivo é facilitar manutenções corretivas e evolutivas futuras.

De acordo com os pontos fracos e melhorias citados ao longo da avaliação do processo de PCP Versão 1, o processo foi revisado e aperfeiçoado pelo SEPG. O grupo identificou as evoluções no processo de PCP, que são discutidas na seção 4.4.

4.4 Processo de PCP Versão 2

O objetivo desta seção é a apresentar o cenário do processo de PCP Versão 2 e as melhorias em relação ao processo na versão 1.

Como suporte a estas melhorias, fez-se necessário incluir novos papéis e artefatos ao processo que são apresentados nas subseções 4.4.3 e 4.4.4, respectivamente. A seção 4.4.5 apresenta uma análise do atendimento às melhorias realizadas (Melo, 2010).

4.4.1 **Cenário de Aplicação do processo PCP Versão 2**

Na seção 4.3 foram apresentados pontos fracos e elementos que oportunizassem melhoria que evidenciam que o processo PCP Versão 1 estava requerendo aperfeiçoamento. O objetivo de atender a essas melhorias é garantir uma melhor execução do processo, mantendo os aspectos positivos e eliminando todos os pontos fracos listados. Desta forma, obtendo um processo com menos inconsistências e gerando um produto de software com qualidade ao seu final (Melo, 2010).

Para iniciar o processo de melhoria, foi necessário criar o grupo de SEPG. Este grupo é composto por no máximo 10 pessoas, envolvendo os responsáveis de cada área do processo. O objetivo do grupo é revisar o processo que está sendo executado, verificar possíveis desvios, corrigir e aplicar as mudanças. Para isto, foi necessário realizar reuniões semanais

para discutir o processo de desenvolvimento que está sendo executado. Nestas reuniões existe um grupo obrigatório que é formado pelos seguintes membros:

- Analista de Qualidade;
- Gerente de Configuração;
- Coordenador de Requisitos;
- Coordenador de Implementação;
- Coordenador de Testes;
- Coordenador de Banco de Dados;
- Coordenador de interface.

Dados os pontos fracos citados na seção 4.3, foi necessária a atualização do processo de PCP versão 1, nomeado de processo de PCP Versão 2. Como forma de iniciar o planejamento do processo e atender as melhorias, foi necessário realizar um levantamento que abrangia a estrutura organizacional da empresa, as ferramentas utilizadas, capacitação de pessoas através de treinamentos e *workshops*, e o processo de desenvolvimento (Melo,2010).

Nas reuniões do SEPG ficou alinhado que, além de atender as melhorias propostas, seria necessário realizar uma análise e abrangência ao Modelo de Melhoria de Processo do Software Brasileiro (Softex, 2011).

Para a abrangência foi utilizado o MPS.BR, baseado na Norma Internacional ISO/IEC 15504 (ISO/IEC, 2006), entretanto poderiam ser utilizados outros modelos de processos de software, como o CMMI, ISO/IEC 12207 (ISO/IEC, 2006), ISO/IEC 15504 (ISO/IEC, 2006). Foi escolhido o MPS.BR, devido a fábrica de software, pertencer ao Governo Federal, o qual incentiva a adoção deste modelo.O MPS.BR atende a necessidade de implantar os princípios de Engenharia de Software de forma adequada às empresas, compatível com as principais abordagens internacionais para definição, avaliação e melhoria de processos de software (Softex, 2011b).

4.4.2 **O Processo de Projeto e Construção de Produto Versão 2**

Nesta seção será apresentado o processo de PCP Versão 2 e suas melhorias. Este processo é ilustrado na Figura 4-4 e suas atividades estão detalhadas no Apêndice E.

O processo de PCP Versão 2 tem seu início através da especificação de requisitos que ocorre juntamente com a atividade de projetar uma solução técnica, que consiste em verificar a viabilidade na construção do requisito. Em paralelo, também ocorre a atividade da equipe de

interface, que é criar o diagrama navegacional, ou seja, determina o local no sistema onde uma determinada funcionalidade estará disponível para o usuário.

Após a criação do digrama navegacional, é criado um protótipo do sistema, seguindo o padrão de interface definido no projeto. Este protótipo é referenciado no artefato de Caso de Uso, e o mesmo servirá como base para a implementação de uma determinada funcionalidade, auxiliando assim na diminuição do tempo do desenvolvimento.

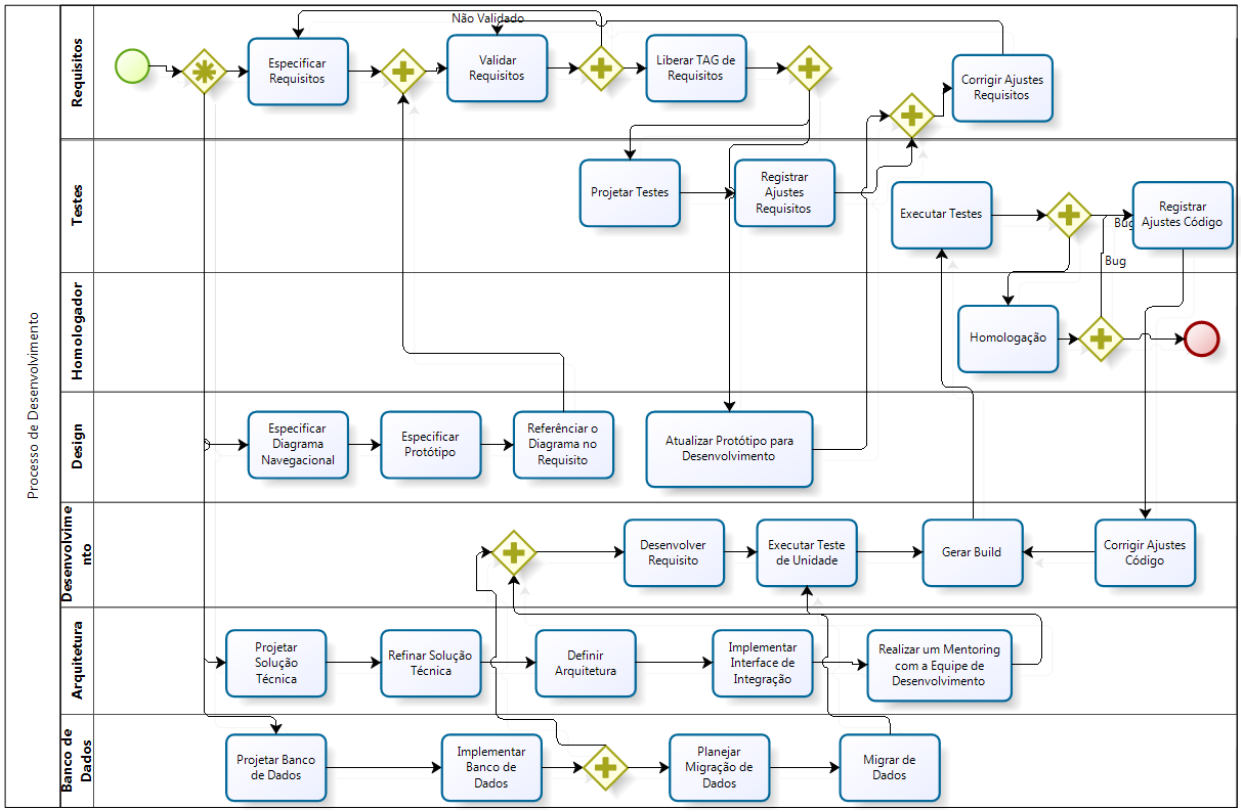

Figura 4-4 – Fluxo Detalhado da Fábrica de Software.

A atividade posterior reflete a validação dos artefatos de Caso de Uso e Regra de Negócio com o cliente, onde o protótipo é mostrado como forma de melhorar o entendimento do cliente, o qual pode aceitar e assinar um termo de aceite daquele requisito, ou então, rejeitar propondo os ajustes a serem realizados. Caso o requisito não tenha sido aceito, é realizada a correção do protótipo e apresentado novamente até obter o termo de aceite (Melo, 2010).

A atividade seguinte consiste em liberar uma *baseline*[1] ou uma TAG de artefatos validados, para que seja iniciada a atividade de testes. Por sua vez, a equipe de teste realiza a criação do Projeto de Testes, que serve como apoio durante a execução dos testes. Nesta atividade é possível que sejam encontradas inconsistências nos artefatos, essas são relatadas pela equipe de testes em uma ferramenta de *bugtracking*, que é direcionado para correção dos documentos com inconsistências e em seguida é realizada uma atualização no protótipo.

Após as correções e ajustes no requisito, o ciclo de desenvolvimento retorna para o processo de validação com o cliente. Depois da aprovação do cliente, este insumo está preparado para ser formalmente liberado, através de uma *baseline*, para o restante das equipes de desenvolvimento (Molinari, 2007).

Após a liberação da *baseline*, a equipe de Arquitetura realiza um refinamento na solução técnica, e então define e cria a arquitetura do sistema. Adicionalmente, implementa as entidades que são responsáveis pela integração com sistemas externos através de *webservices*[2] (Molinari, 2007).

A equipe de arquitetura, por sua vez, realiza um treinamento técnico com toda a equipe de desenvolvimento. O objetivo deste treinamento é o aperfeiçoamento das habilidades técnicas de cada membro da equipe sobre a linguagem de programação utilizada.

Paralelamente a este ciclo, a equipe de banco de dados realiza o projeto do banco de dados e em seguida implementa-o. A equipe de banco de dados inicia, então, o planejamento para realizar a migração de dados, a qual consiste em verificar como os dados são importados para o sistema e posteriormente migrar as informações do sistema legado[3] para o sistema que está sendo desenvolvido. As informações migradas servem de apoio para a equipe de testes e desenvolvimento, já que com dados reais é possível realizar testes mais fiéis (Andrade, 2005).

A equipe de desenvolvimento inicia as atividades de codificação do requisito elicitado. Como apoio à atividade de desenvolvimento, recomenda-se a leitura do documento de modelo de implantações, este documento contempla boas práticas de programação.

---

[1] ***Baseline*** *- Conjunto de artefatos aprovados, estáveis e consistentes. Uma baseline é usada como base no desenvolvimento dos artefatos das próximas fases e tem suas mudanças controladas por um processo formal.*

[2] ***Webservices*** - é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis.

[3] **Sistema Legado** - o termo utilizado em referência aos sistemas computacionais de uma organização que, apesar de serem bastante antigos, fornecem serviços essenciais.

Durante a codificação, a equipe de desenvolvimento cria os testes de unidades que possuem como objetivo realizar testes em determinada funcionalidade do sistema (Melo, 2010).

A próxima atividade consiste na geração de uma *build* [4] para a equipe de testes que realiza os testes funcionais, testes de controle de acesso e de performance. Qualquer defeito encontrado, é relatado na ferramenta de *bugtracking* e direcionado para análise da equipe de desenvolvimento e posterior correção.

Em seguida é disponibilizada uma nova versão da aplicação para a equipe de testes realizar o mesmo procedimento. Caso a equipe de testes não encontre nenhum defeito, é gerada uma *build* para a atividade de homologação com o intuito de receber o aceite de validação do cliente. Na atividade de Homologação, o cliente pode registrar defeitos, que retornam para serem corrigidos pela equipe de desenvolvimento, passando por um novo ciclo de testes.

Depois de todos os defeitos corrigidos, o sistema volta para a atividade de homologação até o aceite do cliente.

Durante a definição do processo de PCP versão 2, procurou-se realizar uma comparação com o modelo da versão 1 do processo para verificar a evolução ocorrida. O Quadro 4-5 permite uma análise deste atendimento, caracterizando o início da implementação do programa da qualidade dos processos organizacionais na Fábrica de Software.

[4] ***Build*** - são partes integrantes do ciclo de vida iterativo. Eles representam tentativas contínuas de demonstrar a funcionalidade desenvolvida até esta data. Cada build é colocado sob o controle da configuração nos casos em que haja necessidade de voltar a uma versão mais antiga, quando a funcionalidade adicionada causar quebras ou qualquer outro efeito que comprometa a integridade do build.

Quadro 4-5 - Comparativo do processo PCP Versão 1 com o processo de PCP Versão 2

Atividades do Processo	Forma de Execução no Processo de PCP Versão 1	Forma de Execução no Processo de PCP Versão 2	Comentários da Forma de Execução
Analisar e Projetar Casos de Uso	A atividade era iniciada mesmo que os requisitos não estivessem prontos	Esta etapa é realizada após a aprovação dos requisitos com o cliente	A diferença entre o processo PCP Versão 2e o processo de PCP Versão 1 é que na versão 2, o requisito só começa a ser desenvolvido depois de ser aprovado pelo cliente. Qualquer mudança de requisito é avaliada para saber o impacto e alocada em uma atividade posterior de acordo com o cronograma.
Analisar e Projetar Arquitetura	Após a geração do documento de requisito era definida a tecnologia sem um estudo prévio.	As atividades de prova de conceito são realizadas para definir qual tecnologia será utilizada.	No processo de PCP Versão 1 existiram alguns impactos no cronograma relacionados com a escolha das tecnologias. Como suporte para seleção de tecnologias ao projeto, foi adicionado ao processo a atividade da prova de conceito. Esta atividade trouxe como beneficio a redução de riscos do não cumprimento do cronograma.
Projetar Banco de Dados	Tabelas criadas com nomes sem padronização.	Tabelas, atributos, e chaves estrangeiras e primárias foram padronizadas.	O ganho da padronização é que ocorre a diminuição nas dúvidas de entendimento do modelo do banco de dados entre os membros da equipe de desenvolvimento.
Estruturar Modelo de Implementação	Não existia um documento contendo uma padronização do código.	Documento contendo uma padronização de implementação.	O beneficio do documento de padronização é que a manutenção do sistema poderia ser feita com poucos problemas de entendimento da codificação.
Realizar Testes de Unidade	Não realização de testes de unidade	Realização do teste de unidade	Com a realização do teste de unidade, houve uma redução de defeitos no sistema, já que os erros eram detectados no tempo do desenvolvimento e logo corrigidos.
Atividade Migração de Dados	Não era realizada	Era realizada a migração de dados	A equipe de Migração de dados cria *scripts* que tem o objetivo de migrar os dados do sistema legado para o sistema que está sendo desenvolvido.

Dando continuidade na melhoria do processo, é importante apresentar os papéis e artefatos que foram introduzidos no processo de PCP Versão 2.

4.4.3 **Papéis utilizados no processo de PCP Versão 2**

O processo Versão 1 continha a definição de 6 papéis definidos. Para o processo Versão 2, foi necessária a adição de 4 novos papéis para apoiar as melhorias propostas.

O papel apresentado no Quadro 4-6, tem como objetivo apoiar o processo de PCP, ou seja, nem sempre está presente nas atividades próprias desse processo.

Quadro 4-6– Papéis de apoio ao PCP.

Papel / função	**Descrição**	**Justificativa**
Analista de Homologação	Papel responsável por conduzir validações do Produto com o cliente, gerando o ambiente propício para homologação, bem como as estratégias a serem executadas ao longo dos trabalhos de validação do produto.	Este papel foi adicionado, pois o cliente solicitou um responsável para apresentar as funcionalidades do sistema e que acompanhasse o processo de homologação.

Os papéis apresentados no Quadro 4-7 estão presentes diretamente no fluxo das atividades do PCP. Foram criados para auxiliar na divisão das atividades do processo.

Quadro 4-7 – Papéis do PCP

Papel / função	**Descrição**	**Justificativa**
Projetista de Interface	Papel responsável por definir o padrão de interface do sistema, bem como, promover treinamentos para institucionalizar o padrão de interface da aplicação.	A criação deste papel ocorreu para a resolução de inúmeras inconsistências encontradas na interface do sistema.
Analista de Migração	Papel responsável por conduzir migrações dos dados do sistema legado para o sistema em desenvolvimento, a fim de corroborar com a produção do sistema.	A criação desde papel se deu pela necessidade do entendimento dos dados presentes no sistema legado para realizar processos de migração para o novo sistema.

Desenvolvedor PL:	Papel responsável por implementar soluções usando a tecnologia PL/SQL.	Durante a atividade de desenvolvimento, houve a necessidade de desenvolver procedimentos do sistema diretamente no banco de dados para que houvesse uma diminuição do tempo de resposta de algumas funcionalidades.

4.4.4 **Artefatos do PCP**

Esta seção tem como objetivo apresentar os 7 artefatos adicionados no processo de PCP versão 2, conforme o Quadro 4-8.

Quadro 4-8– Artefatos de apoio do PCP

Artefato	**Descrição**	**Justificativa**
Plano de Interface do Usuário	O plano de interface do usuário é utilizado para descrever os padrões de usabilidade do sistema.	Este artefato foi adicionado no processo para servir de apoio à atividade de desenvolvimento, já que os relatos do cliente sobre não padronização das interfaces foram grandes.
Documentação no código-fonte	O objetivo da documentação no código-fonte é saber o que determinado trecho de código faz no sistema. Além de contribuir para a realização de futuras manutenções no sistema.	Cada classe e método deveria ser documentado para facilitar na hora de executar uma evolução ou correção de defeitos do sistema.
Glossário	Este documento define todos os termos em comum utilizados no projeto. É equivalente a um dicionário, onde contém informações de conceitos, notações utilizadas, padrões.	Devido aos inúmeros termos técnicos da aplicação, fez-se necessária a criação deste documento como forma de homogeneizar o conhecimento do negócio entre os membros técnicos e o cliente

Checklist da viabilidade técnica e funcional	Esse documento tem o objetivo de verificar se um determinado requisito pode ser desenvolvido, respeitando os limites de custo e tempo do projeto.	A criação deste artefato tem o objetivo de apoiar a equipe de arquitetos do sistema para verificar a complexidade e viabilidade de desenvolvimento de um requisito.
Scripts de criação de estruturas de banco de dados	Este *script* é gerado pela equipe de Banco de Dados para documentar e implementar o Modelo de Banco de Dados do Sistema.	O objetivo é armazenar e automatizar a criação da estrutura do banco de dados da aplicação.

4.4.5 **Análise do atendimento as melhorias**

Nesta seção, serão apresentados os resultados obtidos através da análise, realizada pelo SEPG, dos pontos fortes, os pontos fracos e as oportunidades de melhoria do processo PCP versão 1.

- **Pontos Fortes do processo de PCP Versão 2**
  - **Unificação dos Processos**: O processo foi unificado e com isto, houve uma melhoria na produção da fábrica, já que todos os requisitos tinham que ser validados e cada atividade era executada no seu tempo.
  - **Execução da Prova de Conceito:** A equipe de arquitetos adotou a utilização da prova de conceito. Com isto foi possível verificar quais as tecnologias/*frameworks* eram aptas para resolver um determinado tipo de problema.
  - **Teste unitário do código-fonte**: Foram realizados diversos treinamentos com a equipe, bem como, a institucionalização desta atividade. Como resultado foi possível notar uma quantidade menor de erros, segundo a análises feita no sistema de registros de *bugs*.
  - **Padronização no Banco de Dados:** Como forma de melhorar o entendimento da estrutura do banco de dados, foi necessário realizar uma padronização nas tabelas e objetos do banco de dados.
  - **Código-fonte documentado**: Foram realizados diversos treinamentos com a equipe de desenvolvimento, sobre boas práticas de documentação em código-fonte. Como resultado tem-se uma melhor manutenibilidade do sistema.

O único ponto fraco que não teve seu objetivo alcançado foi o item: **Não utiliza o Reuso de código-fonte**, já que este objetivo encontra-se em fase de amadurecimento por toda equipe.

Das oportunidades de melhorias indicadas todas foram atendidas:

- Treinamentos de arquitetura, tecnologias utilizadas em desenvolvimento e boas práticas de programação, o resultado foi atingido através do documento de implementação, bem como a realização treinamentos e *mentorings* com a equipe de desenvolvimento.
- Institucionalizar a utilização de Testes de Unidade para toda a equipe de Desenvolvimento
- Documentar o código-fonte para facilitar a manutenção evolutiva e/ou corretiva.
- Documentos de Caso de Uso e Regras de negócio necessitam ser validados antes do inicio do desenvolvimento

- **Oportunidades de melhorias – Versão 2**
  - Realizar treinamentos para utilização de técnicas de reutilização de métodos, interfaces visual e procedimentos do sistema,
  - Realizar periodicamente treinamentos de negócio.

Além das melhorias propostas, o SEPG também tinha como objetivo realizar uma verificação para conhecer o grau de abrangência do processo de PCP Versão 2 com o Nível D de maturidade do MPS.BR, usado como base as boas práticas presentes no processo de Projeto e Construção de Produto (Softex, 2011), conforme o Quadro 4-9.

Quadro 4-9 - Abrangência entre o processo PCP Versão 2 e o MPS.BR seção PCP

Atividade	**MPS.BR**	**MPS.BR (Nível)**	**Sugestão**
Analisar e Projetar Casos de Uso	PCP1 - Alternativas de solução e critérios de seleção são desenvolvidos para atender aos requisitos definidos de produto e componentes de produto	Totalmente	Não se aplica
	PCP3 - O produto e/ou componente do produto é projetado e documentado	Parcialmente	É necessário atualizações constantes no documento de boas práticas de codificação.

Atividade	MPS.BR	MPS.BR (Nível)	Sugestão
Analisar e Projetar Arquitetura	PCP1 - Alternativas de solução e critérios de seleção são desenvolvidos para atender aos requisitos definidos de produto e componentes de produto	Totalmente	Não se aplica
	PCP 3- O produto e/ou componente do produto é projetado e documentado	Totalmente	Não se aplica
	PCP7 - A documentação é identificada, desenvolvida e disponibilizada de acordo com os padrões estabelecidos	Parcialmente	A documentação sobre os padrões de arquitetura do sistema precisa ser disponibilizada para o cliente.
Projetar Banco de Dados	PCP1 - Alternativas de solução e critérios de seleção são desenvolvidos para atender aos requisitos definidos de produto e componentes de produto	Totalmente	Não se aplica
	PCP3 - O produto e/ou componente do produto é projetado e documentado	Totalmente	Não se aplica
Validar Diagrama De Banco De Dados	PCP6 - Os componentes do produto são implementados e verificados de acordo com o que foi projetado	Totalmente	Não se aplica
Planejar Migração De Dados	PCP 6 - Os componentes do produto são implementados e verificados de acordo com o que foi projetado	Totalmente	Não se aplica
Estruturar Modelo De Implementação	PCP6 - Os componentes do produto são implementados e verificados de acordo com o que foi projetado	Totalmente	Não se aplica
	PCP 7 - A documentação é identificada, desenvolvida e disponibilizada de acordo com os padrões estabelecidos	Totalmente	Não se aplica
Implementar Banco De Dados	PCP 6 - Os componentes do produto são implementados e verificados de acordo com o que foi projetado	Totalmente	Não se aplica
Codificar Requisito Funcional	PCP 6 - Os componentes do produto são implementados e verificados de acordo com o que foi projetado	Parcialmente	É necessário realizar um processo de validação do que foi construído(código-fonte).

Atividade	MPS.BR	MPS.BR (Nível)	Sugestão
Realizar Testes De Unidade	PCP 6 - Os componentes do produto são implementados e verificados de acordo com o que foi projetado	Totalmente	Não se aplica
Migrar Banco De Dados	PCP 6 - Os componentes do produto são implementados e verificados de acordo com o que foi projetado	Totalmente	Não se aplica

4.5 Considerações Finais

Neste capítulo, houve a contextualização do processo de PCP Versão 1, a qual apresentou as informações do cliente, da empresa fornecedora e a necessidade de criação da Fábrica de Software. Como a fornecedora não possuía uma Fábrica de software anteriormente ao desenvolvimento do projeto, foi necessário realizar uma análise do cenário do projeto para embasar a definição do processo de PCP Versão 1.

Devido a alguns problemas relatados sobre o processo de PCP Versão 1, O SEPG realizou um levantamento e identificou pontos fortes, pontos fracos e oportunidades de melhoria no processo.

Como consequência do resultado das reuniões do SEPG, foi necessário criar a segunda versão do processo, chamada de processo de PCP Versão 2. Adicionalmente, o SEPG identificou que o processo necessitava ter uma abrangência do MPS.BR, que é um modelo de amadurecimento gradual dos processos de desenvolvimento de software.

Como complemento à avaliação realizada no processo de PCP Versão 2, a equipe do SEPG verificou a importância de avaliar o desempenho e a efetividade do processo através de questionários, entrevistas e coleta de métricas, que são apresentadas no Capítulo 5.

Capítulo 5 - **Avaliação do Processo de PCP**

As empresas que produzem software estão aprimorando seus processos de desenvolvimento, com o objetivo de gerar um produto com qualidade e aumentar a competitividade no mercado. Segundo Bartié(2002) é difícil obter um produto de software que tenha uma qualidade aceitável, quando o processo de desenvolvimento é frágil e deficiente. Portanto, não é possível estabelecer um processo de garantia da qualidade que não tenha enfoque, simultaneamente, no produto tecnológico e no processo de desenvolvimento desse software (Jeston, 2008).

Para se obter um processo de desenvolvimento de software que auxilie na produção de um produto com qualidade, faz-se necessário realizar avaliações periódicas no processo que está sendo executado. O objetivo da avaliação é saber se o processo atende as necessidades da empresa, ou seja, que gere um produto com qualidade e atenda a necessidade do cliente (Bartié, 2002).

Como forma de avaliação no Processo de Projeto e Construção do Produto Versão 2, foram utilizada dois tipo de avaliação: a quantitativa, realizada através de coleta de métricas; e a qualitativa, através de um questionário distribuído para a equipe de desenvolvimento. Os dados coletados de ambas avaliações foram tabulados e analisados com intuito de verificar o efetividade do processo.

A contribuição deste trabalho para a Engenharia de software, é identificada através da definição do processo de PCP utilizando a notação do BPMN, por ser uma linguagem de notação de fácil entendimento. Outra contribuição é que foram realizadas avaliações quantitativas e qualitativas para acompanhamento do processo, que serão apresentadas neste capítulo (Miers, 2008).

Este capítulo está estruturado da seguinte forma, a seção 5.1 aborda a metodologia da avaliação. A seção 5.2 apresenta como foi realizada e definida a avaliação quantitativa. A seção 5.3 apresenta como foi realizada e definida a avaliação qualitativa. Na seção 5.4 são apresentadas as considerações finais.

5.1 Metodologia de Avaliação

Empresas que tem sua atividade fim o desenvolvimento de software buscam a cada dia melhorar seus processos de desenvolvimento para alcançar um produto final com qualidade e que atenda a necessidade do seu cliente. Alcançar a qualidade de um produto de software nem sempre é uma atividade simples de se conseguir, muitas vezes a empresa é obrigada a testar diversos processos de softwares até chegar no que mais se adapte a realidade dela e de seus projetos(Bartié, 2002).

Testar processos de software pode ter um custo elevado para a empresa, entretanto, se o processo não for avaliado periodicamente, pode acarretar em grandes prejuízos para a empresa e o seu cliente (Basili, 1994).

Como forma de avaliar o processo de PCP Versão 2, foram utilizados dois tipos de avaliações: a quantitativa e a qualitativa; que serão explicadas com mais detalhes a seguir. A finalidade destas avaliações é identificar riscos ou gargalos, avaliar mudanças e avaliar a qualidade de artefatos produzidos (Weber, 2006).

O primeiro tipo de avaliação utilizada neste trabalho foi a avaliação quantitativa, onde optou-se pela escolha do GQM (*Goal – Questions - Metrics*), no qual, a partir de objetivos definidos pela empresa, é possível chegar a resultados que auxiliam a medição da qualidade do que está sendo produzido (Basili, 1994).

Segundo Basili(1994) o método proposto contém três níveis: Conceitual (Objetivo); Operacional (Questões); e Quantitativo (Métricas); a saber:

- **Objetivos** – *Goal* – é utilizado como o objetivo principal, ou seja, onde e o que a empresa deseja alcançar;
- **Questões** – *Question* – para cada objetivo deve-se criar questões pertinentes, que ajudam a revelar se as metas foram alcançadas;
- **Métricas** – *Metrics* – cada questão deve ser associado a um conjunto de métricas.

O GQM é um método ou um paradigma orientado a objetivos e tem como meta medir a qualidade dos processos e produtos gerados. O GQM também ajuda na identificação de métricas úteis e relevantes, assim como um apoio à análise e à interpretação dos dados coletados. Como resultado permite uma avaliação das conclusões obtidas (Basili, 1994b).

O GQM pode ser utilizado para apoiar na medição de qualquer tipo de produto ou processo, não importando qual o escopo do projeto, ou seja, desde a sua caracterização até o

controle e aperfeiçoamento. A abordagem GQM provê um *framework* que envolve três passos (Basili, 1994b):

• Listar os principais objetivos do processo de medição, através de reuniões com os gerentes de projeto e o coordenador da equipe de desenvolvimento;

• Derivar de cada objetivo as perguntas que devem ser respondidas para determinar se os objetivos foram atingidos;

• Decidir o que precisa ser medido para ser capaz de responder as perguntas adequadamente (definição das métricas).

Para realizar a execução da avaliação quantitativa através do GQM, foi necessário seguir as etapas abaixo:

• **Planejamento**: envolve a seleção da aplicação a ser mensurada, definição, caracterização e planejamento do projeto;

• **Definição**: fase onde os objetivos, questões, métricas e hipóteses são definidos e documentados;

• **Coleta de dados**: atende às métricas definidas;

• **Interpretação**: fase onde os dados coletados são analisados para identificar as respostas às questões definidas.

A segunda avaliação utilizada, foi realizada através de questionários distribuídos para a equipe de desenvolvimento. O objetivo da elaboração de um questionário ocorreu através da necessidade de obter dados diretamente com as pessoas que participaram do processo de PCP Versão 2. Para o preenchimento do questionário foram necessárias reuniões com os gerentes do projeto e os coordenadores de cada área. Estas reuniões tinham como objetivo de conhecer cada atividade do processo que estava sendo realizada nas diversas áreas da Fábrica de Software (Basili, 1994b).

A elaboração do questionário ocorreu através de alguns passos listados abaixo:

- Levantamento de informações sobre as atividades executadas na Fábrica de Software;
- Definição dos Critérios para avaliação;
- Definição das perguntas do questionário;
- Definição das respostas do questionário;
- Distribuir um questionário de teste, para 3 pessoas como forma de validação;

- Distribuir o questionário para a equipe de desenvolvimento;
- Coletar os questionários;
- Realizar a tabulação do dados;
- Analisar os resultados obtidos;
- Gerar um relatório sobre a análise realizada.

O detalhamento das avaliações quantitativas e qualitativas serão abordados neste capítulo nas seções5.2 e 5.3, respectivamente.

5.2 Avaliação Quantitativa

A avaliação quantitativa ocorreu através de auditórias realizadas no repositório de ativo (ferramenta de CVS) do projeto, onde é possível visualizar o histórico de alterações de determinada atividade, como por exemplo: quem, quando e o que foi realizado.

A coleta de dados ocorreu no período de 06 de outubro de 2010 a 12 de fevereiro de 2011, como forma de verificar as evoluções do processo de PCP Versão 2, e foi dividida em 3(três) iterações. Sendo que a auditoria foi realizada através de amostragem, incluindo todos os membros da equipe de desenvolvimento do projeto, no total de 14 (quatorze) pessoas. As informações detalhadas sobre a coleta encontra-se no Apêndice A.

Como forma de aplicar o conhecimento discutido nos capítulos anteriores, as seções seguintes permitem uma analise das medições propostas para verificar se o processo de PCP está atendendo o cenário da fábrica de software, relatado no capítulo 4.

A partir da meta definida no Quadro 5-1 e usando o método GQM (Basili, 1994b), foi estabelecido um conjunto de questões para análise do processo, juntamente com as métricas que servirão para receber os dados experimentais e formular respostas para estas questões(Lahoz, 2003):

Quadro 5-1 - Meta definida para o GQM

Atributo	**Valor para o Experimento**
Analisar (Objeto do Estudo)	Atividades, procedimentos e resultados realizados/obtidos pela execução do processo de Projeto e Construção do Produto.
Com o propósito de (Objetivo)	Avaliar, aperfeiçoar e comparar
Com respeito a(o).	À disciplina e boas práticas processo de Projeto e Construção do Produto. Ao processo de Projeto e Construção do Produto inicial praticado na organização.
Do ponto de vista (Perspectiva)	Gerentes de Projetos, Coordenadores de Desenvolvimento de Software, Desenvolvedores.
No contexto da	Organização e ambiente de desenvolvimento em que ocorre o estudo de caso realizado.

Como forma de padronização do resultado obtido da coleta, todas as métricas possuem indicadores para facilitar a leitura e interpretação dos resultados, esses indicadores encontram-se no detalhamento de cada uma das métricas. Todos os indicadores listados nesta pesquisa, foram definidos pelos gerentes do projeto.

Para definição dos objetivos, foi necessário realizar reuniões com os gerentes do projeto e os coordenadores de todas as áreas da Fábrica de Software, como resultado foram definidos 3 objetivos, que são listados abaixo:

- **Objetivo 1**: Definir a produtividade da equipe de desenvolvimento utilizando as práticas definidas no processo;
- **Objetivo 2**: Definir a qualidade do produto que está sendo produzido;
- **Objetivo 3**: Verificar a gestão das estimativas dos recursos.

O Quadro 5-2, apresenta as questões que foram elaboradas para atender cada um dos objetivos da empresa. Após a elaboração das questões, foi necessário definir as métricas baseada em cada uma das questões.

Quadro 5-2–Tabela contendo as questões e métricas para serem utilizadas com GQM

Objetivo 1	
Questões	**Métricas**
Q1: Qual o tempo para realização da atividade	M1: Quantificar a velocidade da equipe de desenvolvimento.
	M2: Diferença entre o tempo estimado e o real definido pela equipe.
Q2: Qual o envolvimento do membro da equipe na atividade	M3: Definir a disponibilidade da equipe para a atividade.
Q3: Qual a complexidade da atividade.	M4: Avaliar a complexidade das atividades.
	M5: Grau de conhecimento da equipe nas regras que envolvem a atividade.
Objetivo 2	
Questões	**Métricas**
Q4: Qual o nível de qualidade do produto interno gerado de acordo com a visão dos membros da equipe?	M6: Quantificar o número de *bugs* de acordo com sua complexidade?
	M7: Analisar o grau de aderência das atividades da equipe em relação aos critérios de auditoria?
Q5: Qual o grau de adequação do produto desenvolvido ao requisito?	M8:Quantificar o percentual de aderência do produto gerado em relação ao critério de verificação dos requisitos.
Q6 :Qual a abrangência do Teste de Unidade?	M9: Quantificar o percentual de cobertura dos testes de unidade em relação às transações (fluxo principal e alternativo)?

Q7: Qual o nível de aderência do produto ao padrão de interface?	M10: Quantificar o percentual de aderência dos itens de verificação do padrão de interface em relação ao produto gerado?
Objetivo 3	
Questões	**Métricas**
Q8: Qual o grau de variância da equipe nas estimativas do projeto?	M11: Qual a diferença entre o tempo estimado pela Gerência e o tempo estimado pela equipe para realização das atividades?

Durante as reuniões realizadas entre o SEPG, a Gerência do Projeto e os coordenadores, foi possível verificar as necessidades daquela organização. O detalhamento de todas as métricas encontram-se do Quadro 5-3 até o Quadro 5-13. Cada quadro apresenta informações que auxiliaram na coleta de cada uma das métricas.

Quadro 5-3–Detalhamento da Métrica 1

Quantificar a velocidade da equipe de desenvolvimento.	
O que?	Quantificar quanto tempo de trabalho leva o Engenheiro de Software para realizar as tarefas de sua responsabilidade
Quando?	Atividades repassadas pelos coordenadores, de acordo com o cronograma.
Como?	A coleta é realizada no começo e no término de cada atividade. Será aplicado a fórmula, e assim extraindo os resultados para uma análise
Onde?	Na Fábrica de Software
Por que?	Os coordenadores e Gerentes precisam verificar qual a velocidade da equipe para melhor estimar as atividades no cronograma.
Indicadores	• **Rápido:** 1 a 3 horas • **Normal**: 4 a 7 horas • **Lento**: Acima de 8 horas
Resultado esperado	Que o indicador Lento seja abaixo de 10%.

Quadro 5-4– Detalhamento da Métrica 2

Quantificar a velocidade da equipe de desenvolvimento	
O que?	Definir a diferença entre o tempo estimado para a realização de uma atividade com o tempo real da execução da mesma.
Quando?	Atividades repassadas pelos coordenadores, de acordo com o cronograma.
Como?	Existem dois momentos para realização da coleta. O primeiro momento é quando o Engenheiro de Software estima o tempo para o desenvolvimento da tarefa. O segundo momento é a coleta de quanto tempo a atividade foi realizada. A análise será feita em cima dos dados coletados, mostrando a diferença entre o tempo estimado para o tempo real.
Onde?	Na Fábrica de Software
Por quê?	Os coordenadores e Gerentes precisam verificar se o que está sendo estimado está equivalente com o que foi planejado.
Indicadores	• **Acima de 2 horas:** Subestimada • **Abaixo de 2 horas**: Superestimada • **Entre o intervalo de 2 hora positivas e negativas:** Normal
Resultado esperado	Que o indicador Acima de 2 horas seja inferior a 30%.

Quadro 5-5– Detalhamento da Métrica 3

Definir a disponibilidade da equipe para a atividade	
O que?	Definir quando a equipe ou parte da equipe está disponível ou ociosa para realizar uma nova atividade.
Quando?	Atividades repassadas pelos coordenadores, de acordo com o cronograma.
Como?	A coleta será baseada em um questionário, sendo possível definir de forma aproximada o tempo que a equipe fica disponível.
Onde?	Na Fábrica de Software
Por quê?	Os coordenadores e Gerentes precisam verificar quanto tempo um colaborador possui para desenvolver suas atividades dentro da fábrica, de acordo com o cronograma. O motivo desta verificação é avaliar se o tempo que é perdido com atividade extra trabalho pode prejudicar o andamento do cronograma.
Indicadores	• **Igual ou acima de 34 horas semanais**: Normal • **Abaixo de 34 horas semanais**: Atenção
Resultado esperado	Que o indicador Acima ou igual a 34 horas seja maior que 75%.

Quadro 5-6– Detalhamento da Métrica 4

Avaliar a complexidade das atividades	
O que?	Avaliar o grau de complexidade das atividades executadas pela equipe
Quando?	Atividades repassadas pelos coordenadores, de acordo com o cronograma.
Como?	A coleta será realizada através de um questionário. Este questionário é composto por questões do tipo: Qual a quantidade de atributos o requisito possui, ou ainda qual a quantidade de dependência deste requisito com outros requisitos. Neste questionário cada resposta terá um peso. A soma total de pontos será possível determinar a complexidade do requisito.
Onde?	Na Fábrica de Software
Por quê?	Os coordenadores e Gerentes precisam verificar a complexidade das atividades. Que dependendo da complexidade é necessário haver um detalhamento maior, os insumos contendo mais informações para facilitar o desenvolvimento do mesmo.
Resultado esperado	Que o indicador: ”Normal” seja maior que 50%.

Quadro 5-7– Detalhamento da Métrica 5

Grau de conhecimento do Engenheiro de Software, para uma determinada atividade	
O que?	Medir o grau de conhecimento do Engenheiro de Software, para uma determinada atividade
Quando?	Atividades repassadas pelos coordenadores, de acordo com o cronograma.
Como?	A coleta será realizada através de um questionário, contendo informações de negócio do projeto. Dessa forma poderá ser mensurado o quanto aquele Engenheiro de Software conhece as regras do produto que está sendo desenvolvido.
Onde?	Na Fábrica de Software
Por quê?	A empresa precisa saber da equipe de desenvolvimento qual o grau de entendimento do negócio.
Indicadores	• **Inferior a 60%**: Atenção • **Igual ou acima de 60%**: Normal
Resultado esperado	Que o indicador: ”Igual ou acima de 60%” seja maior que 65%.

Quadro 5-8– Detalhamento da Métrica 6

Quantificar o número de defeitos de acordo com sua complexidade	
O que?	Agrupar a quantidade de defeitos encontrados de acordo com sua complexidade.
Quando?	Atividades relatadas pela equipe de teste no sistema de registro de problemas (*bug tracking*)
Como?	A coleta será realizada através da ferramenta de relato de defeitos. Após a coleta os dados serão agrupados.
Onde?	Na Fábrica de Software
Por quê?	A empresa precisa verificar que o produto que está sendo produzido está de acordo com o nível de qualidade definido em contrato.
Indicadores	• **Até 1 defeito importante e Nenhum crítico e qualquer quantidade de defeito trivial:** Normal • **Acima de 1 defeito importante, nenhum crítico e qualquer quantidade de defeito trivial:** Atenção • **A partir de 1 erro crítico e qualquer quantidade de defeito trivial: Impeditivo**
Resultado esperado	Que o indicador: ”Impeditivo seja menor que 20%.

Quadro 5-9 – Detalhamento da Métrica 7

Analisar o grau de aderência das atividades da equipe em relação aos critérios de auditoria	
O que?	Analisar o grau de aderência das atividades da equipe de acordo com os critérios que foram definidos para verificar a qualidade do que está sendo gerado. O objetivo é descobrir se todas as atividades e tarefas do processo estão sendo executadas de forma correta.
Quando?	Constantemente realizadas, avaliando tudo que é produzido no sistema de controle de versão de arquivo, onde ficam gravadas as alterações que um colaborador fez.
Como?	Através de um questionário que será respondido pelo engenheiro de software é possível definir o quanto ele está aderente ao processo.
Onde?	Na Fábrica de Software
Por quê?	A área de qualidade precisa saber se o processo está sendo seguido de forma correta.
Indicador	• **Acima de 75% das perguntas respondidas**: Contempla • **Entre 50% e 74% das perguntas respondidas:** Contempla em parte • **Abaixo de 50% das perguntas respondidas**: Não Contempla
Resultado esperado	Que o indicador: ”Contempla “seja acima de 80%.

Quadro 5-10– Detalhamento da Métrica 8

Quantificar o percentual de aderência do produto gerado em relação ao critério de verificação dos requisitos	
O que?	Quantificar o percentual de aderência do produto gerado em relação ao critério que foram definidos para a verificação dos requisitos.
Quando?	Verificar a cada final de ciclo, se que foi desenvolvido está de acordo com o que foi especificado.
Como?	Será realizada uma verificação no requisito e no código-fonte para identificar qual o grau de aderência. Esse grau de aderência será obtido através de pontuações, essas pontuações servem para verificar o grau de aderência entre o produto gerado e os requisitos. O responsável realiza a verificação do fluxo principal e atribui uma pontuação. Verifica a quantidade de fluxos alternativos e atribui uma pontuação.

Onde?	Na Fábrica de Software
Por quê?	Garantir que o requisito esteja totalmente de acordo com o que foi desenvolvido.
Indicador	• **Acima de 75% das perguntas respondidas**: Normal • **Entre 21% e 74% das perguntas respondidas:** Aceitável • **Abaixo de 20% das perguntas respondidas**: Crítico
Resultado esperado	Que o indicador: ”Normal “seja acima de 85%.

Quadro 5-11– Detalhamento da Métrica 9

Quantificar o percentual de cobertura dos testes de unidade em relação às transações (fluxo principal e alternativo).	
O que?	Verificar o percentual de cobertura dos testes de unidades de acordo com o requisito, levando-se em conta o fluxo principal e os fluxos alternativos.
Quando?	Verificar a cada final de ciclo, se a equipe de desenvolvimento está implementando os testes de unidade corretamente, de acordo com os fluxos definidos nos requisitos.
Como?	Um especialista em desenvolvimento irá realizar uma auditoria nos testes de unidades, verificando a aderência de acordo com o requisito. Essa aderência será de acordo com a cobertura de testes do fluxo principal e do fluxo alternativo.
Onde?	Na Fábrica de Software
Por quê?	Garantir que todos os testes de unidades sejam feitos de acordo com os fluxos principais e alternativos do requisito.
Indicador	• **Acima de 75% das perguntas respondidas:** Normal • **Entre 50% e 74% das perguntas respondidas:** Aceitável • **Abaixo de 50% das perguntas respondidas:** Abaixo do esperado
Resultado esperado	Que o indicador: ”Normal “seja acima de 85%.

Quadro 5-12– Detalhamento da Métrica 10

Quantificar o percentual de aderência dos itens de verificação do padrão de interface em relação ao produto gerado	
O que?	Verificar o percentual de aderência dos itens do padrão de interface junto ao produto gerado
Quando?	Verificar a cada final de ciclo, se o que a equipe de desenvolvimento está implementando, está de acordo com o padrão de interface solicitado pelo cliente.
Como?	Um especialista em desenvolvimento irá realizar uma auditoria entre o documento de padrão de interface e o produto gerado, verificando a aderência entre ambos. A aderência será realizada de acordo com o documento de padrão de interface.
Onde?	Na Fábrica de Software
Por quê?	Garantir que o produto gerado esteja de acordo com o padrão de interface definido.
Indicador	• **Acima de 75% das perguntas respondidas:** Normal • **Entre 50% e 74% das perguntas respondidas**: Aceitável • **Abaixo de 50% das perguntas respondidas:** Abaixo do esperado
Resultado esperado	Que o indicador: ”Normal “seja acima de 85%.

Quadro 5-13– Detalhamento da Métrica 11

Qual a diferença entre o tempo estimado pela Gerência e o tempo estimado pela equipe para realização das atividades	
O que?	Verificar a diferença entre o tempo estimado pela Gerência e o tempo estimado pela equipe que irá realizar a atividade.
Quando?	Atividades repassadas pelos coordenadores, de acordo com o cronograma.
Como?	A coleta será feita a partir da estimativa da equipe de desenvolvimento e da estimativa da gerência. Desta forma é possível saber se existe diferença entre a estimativa entre os dois grupos.

Onde?	Na Fábrica de Software
Por quê?	Os coordenadores precisam verificar se o que está sendo estimado está equivalente com o que foi planejado.
Indicador	• **Acima de 2 horas:** Superestimada • **Entre o intervalo de 2 hora positivas e negativas:** Normal • Abaixo de 2 horas**: Subestimada**
Resultado esperado	Que o indicador "Superestimado" seja inferior a 30%.

5.2.1 Coleta e Análise das Métricas

A análise da métrica M1, conforme ilustrado na Figura 5-1, ocorreu a através de auditorias realizada no repositório de ativo, onde foi verificado que 45% das atividades são executadas entre 1 a 3 horas, 43% das atividades são executadas em um período de 4 a 7 horas e 12% das atividades são realizadas com o tempo acima de 8 horas. O resultado esperado pela gerência não foi alcançado, vide Quadro 5-3.

Na primeira análise é possível inferir que as atividades são consideradas com o tempo de desenvolvimento lento, possivelmente isso se deve porque os requisitos não foram bem escritos e a equipe de desenvolvimento possui um baixo conhecimento das tecnologias utilizadas. Para as atividades que tem o tempo considerado normal, infere-se que os requisitos estão bem definidos, entretanto, falta a realização de treinamentos técnicos com a finalidade de melhorar o conhecimento sobre a arquitetura do sistema.

Com o objetivo de nivelar o conhecimento técnico da equipe de desenvolvimento foi recomendado realizar treinamentos da arquitetura e dos *frameworks*, treinamentos de negócio, assim como verificar a qualidade dos requisitos que estão sendo produzido.

Na segunda coleta realizada, verificou-se que 40% das atividades tiveram o tempo de desenvolvimento rápido, 53% das atividades tiveram o tempo de desenvolvimento considerado como normal e 7% das atividades tiveram o tempo de desenvolvimento lento. O resultado esperado pela gerência foi alcançado.

Infere-se que os treinamentos técnicos e de negócios realizados, ajudaram a equipe de desenvolvimento a produzirem em um tempo menor.

Na última coleta realizada, verificou-se que 19% das atividades tiveram o tempo de desenvolvimento rápido, 77% das atividades tiveram o tempo de desenvolvimento considerado como normal e 4% das atividades tiveram o tempo de desenvolvimento lento. O resultado esperado pela gerência foi alcançado novamente.

Infere-se que a continuidade dos treinamentos técnicos e de negócio, auxiliaram a equipe de desenvolvimento a produzir determinadas funcionalidades em um tempo menor.

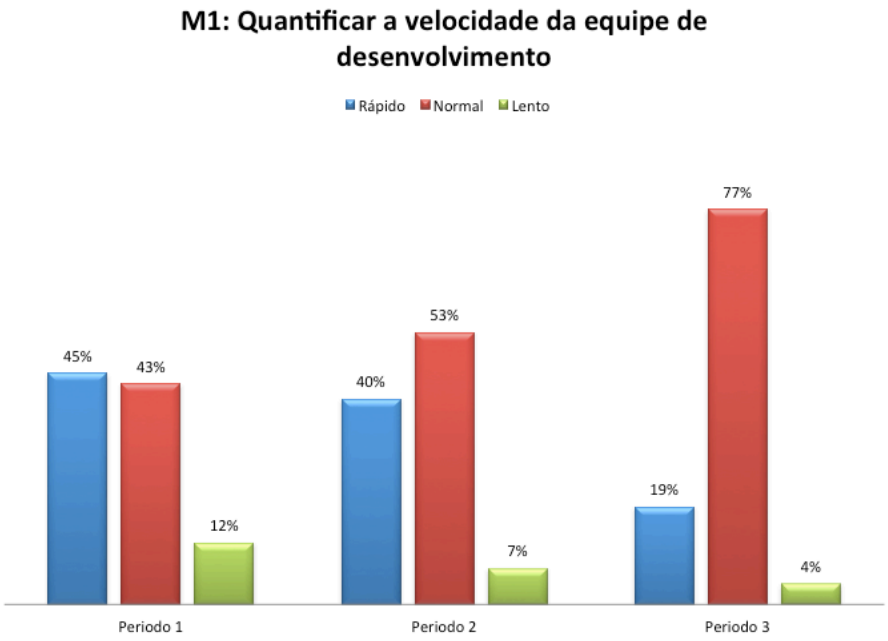

Figura 5-1 – Resultado da métrica 1

A próxima métrica a ser avaliada é a M2, ilustrado na Figura 5-2, onde, conforme os dados coletados, verificou-se que 63% das atividades são estimadas “Acima de 2 horas“, 22% das atividades tem sua estimativa “Abaixo de 2 horas“ e, o restante, 15% das atividades são estimadas “Entre 2 horas positivas e 2 horas negativas“. O resultado esperado pela gerência não foi alcançado, vide Quadro 5-4.

Em uma conclusão parcial, infere-se que as atividades que tem o indicador “Acima de 2 horas“, possuem algum tipo de inconsistência no requisito e adicionalmente a falta de entendimento de negócio, por parte da equipe de desenvolvimento. O indicador “abaixo de 2 horas“ e “Entre 2 horas positivas e 2 horas negativas“, infere que os requisitos não possuem inconsistências e que parte da equipe de desenvolvimento tem um bom entendimento do negócio.

Para que o objetivo seja alcançado, foi recomendado realizar *workshops* de negócio, ou seja, de regras e fluxos de trabalho do cliente que precisam ser sistematizados para melhorar o entendimento da equipe de desenvolvimento.

Na segunda coleta de dados verificou-se os seguintes resultados: 41% das estimativas estavam “Acima de 2 horas“, 35% estava com a estimativa “Abaixo de 2 horas“ e 24% estimaram as atividades “Entre 2 horas positivas e 2 horas negativas“. O resultado esperado pela gerência não foi alcançado.

Recomenda-se a realização de treinamentos de estimativa, com uma carga horária mínima de 8 horas, já que as estimativas fornecem dados que permite prever a quantidade de pessoas e o tempo necessário para realizar cada tarefa do cronograma, assim como os custos do projeto. Não sendo aconselhável elaborar cronograma e orçamento sem o uso de estimativas.

Após a realização do treinamento de estimativa, foi realizada a terceira coleta, onde verificou-se que: 21% das atividades foram estimadas "Acima de 2 horas", 34% das atividades foram estimadas em "Abaixo de 2 horas" e 45% das atividades foram estimadas "Entre 2 horas positivas e 2 horas negativas". O resultado esperado pela gerência foi alcançado.

Infere-se que ao realizar treinamentos sobre as regras e fluxos de trabalho do cliente, percebe-se um melhor entendimento do sistema pela equipe de desenvolvimento. Além disso, os workshops de estimativas ajudam a definir o tempo necessário para realizar cada tarefa, dentro do prazo definido no cronograma.

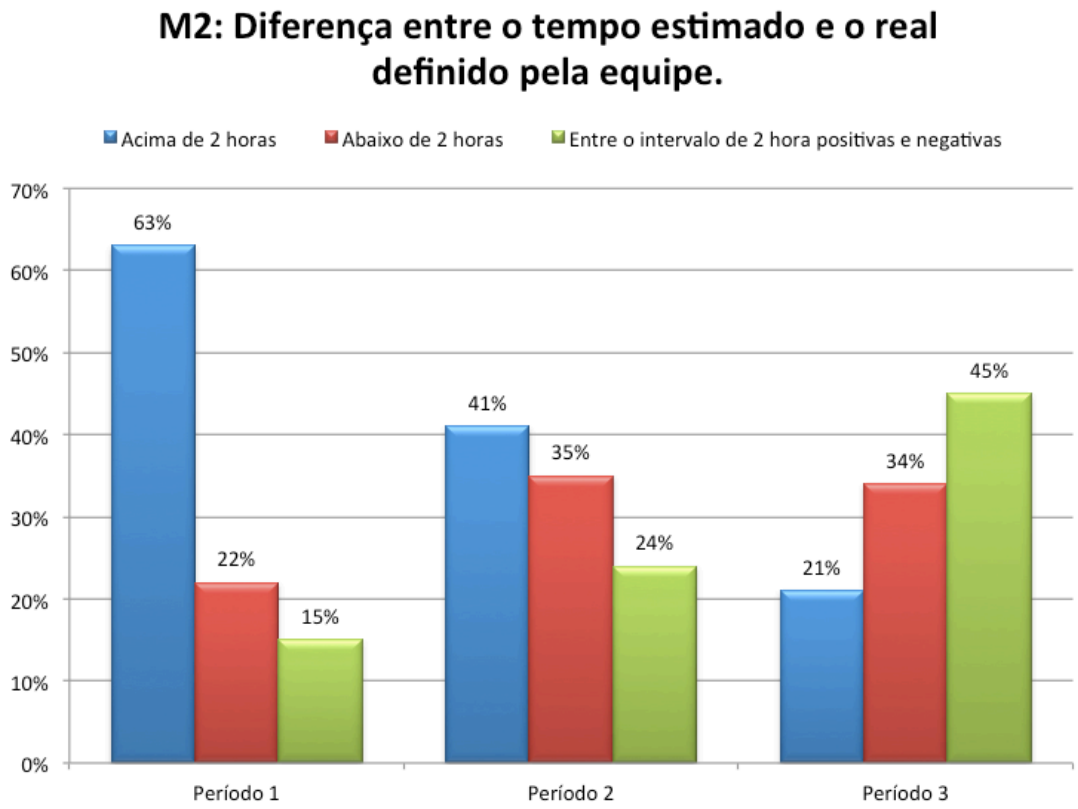

Figura 5-2 – Resultado da métrica 2

A próxima métrica a ser avaliada é a M3, ilustrado na Figura 5-3, onde se verificou que na primeira coleta realizada: 82% dos colaboradores têm atividades extras ao trabalho,

que o fazem não cumprir as 34 horas semanais, 18% apenas cumprem o indicador “Igual ou acima de 34 horas semanais”. O resultado esperado pela gerência não foi alcançado, vide Figura 5-3.

A recomendação é que os coordenadores informem a suas equipes que o tempo definido para ser cumprido semanalmente no cronograma é de 34 horas. Caso o colaborador não cumprisse esta quantidade de horas, o prazo definido para o cronograma estaria sujeito a não ser respeitado.

Na segunda coleta, houve uma diminuição no indicador “Abaixo de 34 horas semanais” que foi de 82% para 71% e o indicador “Igual ou acima de 34 horas semanais” registrou um aumento de 18% para 29%. O resultado esperado não foi atingido.

Recomenda-se que os gerentes do projeto conversem com toda equipe, explicando que o não cumprimento de 34 horas semanais, causará impacto no cronograma, já que o mesmo foi definido desta forma.

Na terceira coleta realizada, o cenário foi pouco alterado, já que o indicador “Abaixo de 34 horas semanais“ reduziu de 71% para 67% e o indicador ”Igual ou acima de 34 horas semanais“ aumentou de 29% para 33%. Este resultado não correspondeu à expectativa da gerência.

A medida adotada pela gerência foi a institucionalização de um controle de horas dos colaboradores, obtendo assim, um controle efetivo das horas trabalhas.

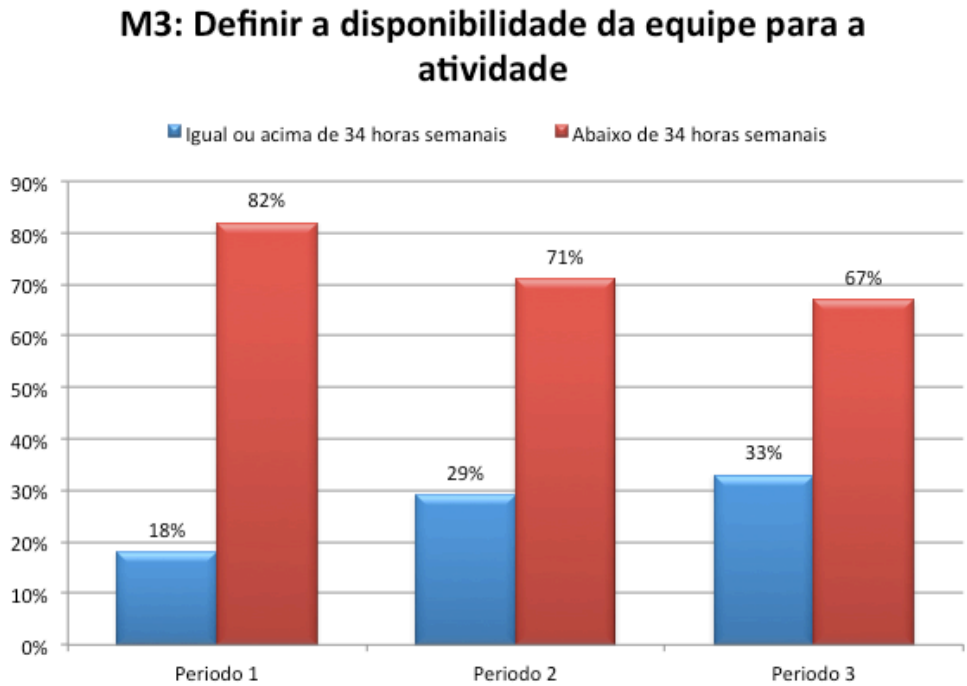

Figura 5-3 – Resultado da métrica 3

A próxima métrica a ser analisada é a M4, ilustrado na Figura 5-4, o objetivo definir qual o grau de complexidade das atividades que estavam sendo executadas pela equipe de desenvolvimento, com a finalidade de minimizar possíveis riscos no cronograma.

A primeira coleta apresentou que o indicador “Superestimado” registrou a marca de 81%, o indicador “Subestimado” marcou 12% e 7% para as atividades consideradas normais. O resultado esperado pela gerência não foi alcançado, vide Quadro 5-5.

Para alcançar o resultado esperado, recomendou-se revisar todos os insumos gerados pela equipe de requisitos e definir se o detalhamento dos mesmos precisam ser alterado ou não. A finalidade é diminuir a ambiguidade dos requisitos, a fim de reduzir o tempo despendido para o entendimento do desenvolvedor.

Na segunda medição o indicador “Superestimado” reduziu de 81% para 54%, o indicador “Subestimado” aumentou de 12% para 22% e o indicador “Normal” aumentou de 7% para 24%. Depois da revisão que houve nos requisitos foi perceptível a melhora nos resultados, porém ainda longe do esperado pela empresa.

Recomenda-se a realização de *workshops* sobre as regras e fluxos de trabalho do cliente, auxiliando assim a equipe de desenvolvimento para um melhor entendimento do sistema.

Na última medição o indicador “Superestimado” registrou uma redução de 54% para 35%, o indicador “Subestimada” reduziu de 22% para 14% e o indicador “Normal” elevou de 24% para 51%. O resultado esperado foi alcançado.

Infere-se que a revisão dos insumos gerados pela equipe de requisitos, contribuiu para a criação de artefatos com baixa ambiguidade, desta forma diminuindo o tempo gasto para o entendimento do desenvolvedor, conforme a Figura 5-4.

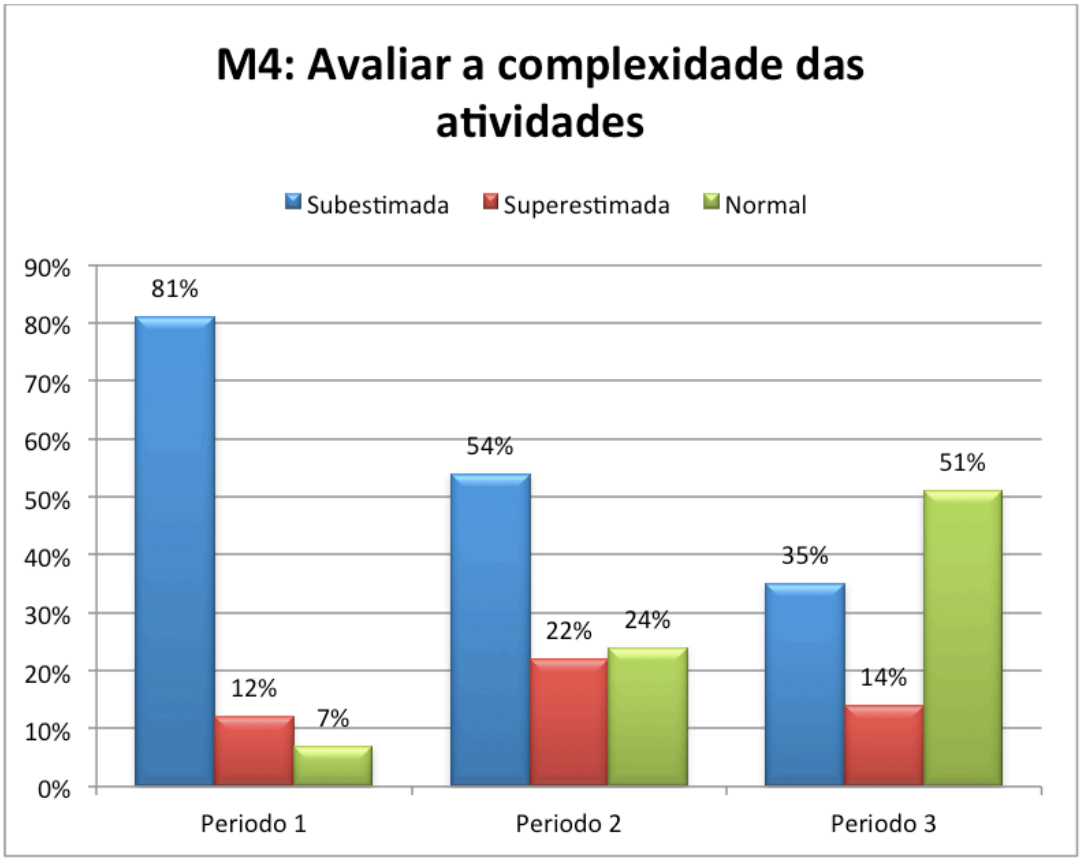

Figura 5-4 – Resultado da métrica 4

A próxima métrica a ser analisada é a métrica M5, ilustrada na Figura 5-5. Durante as reuniões entre a gerência, os coordenadores e os analistas de qualidade surgiram dúvidas a respeito do grau de entendimento do negócio pela equipe de Desenvolvimento. O objetivo era verificar se através de perguntas básicas era possível definir o grau de conhecimento do negócio que equipe de desenvolvimento possuía.

Na primeira coleta realizada foi possível verificar que o indicador "Inferior a 60%" registrou 55% enquanto que o indicador "Igual ou acima de 60%" registrou 45%. O resultado esperado pela gerência não foi alcançado, vide Quadro 5-7.

Com base nos resultados acima, é possível inferir que ao realizar treinamentos sobre as regras e fluxos de trabalho do cliente, os mesmos auxiliam a equipe de desenvolvimento para um melhor entendimento do sistema. Desta forma, recomenda-se a realização deste *workshop*.

Na segunda coleta realizada para esta métrica verificou-se que o indicador "Inferior a 60%" reduziu de 55% para 39% e o indicador "Igual ou acima de 60%" registrou um aumento de 45% para 61%. O valor ainda não era satisfatório para a gerência.

Verificou-se que os resultados parciais mostram que a equipe de desenvolvimento precisava de um treinamento de negócio mais detalhado. Preferencialmente, com carga horária de 40 horas, auxiliando assim em um melhor entendimento do sistema.

Na terceira coleta realizada, verificou-se que o indicador "Inferior a 60%" reduziu de 39% para 28% e o indicador "Igual ou acima de 60%" aumentou de 61% para 72%. Desta forma, o resultado esperado pela gerência foi alcançado.

A conclusão os treinamentos realizados ajudaram a equipe de desenvolvimento a ter um melhor entendimento do negocio.

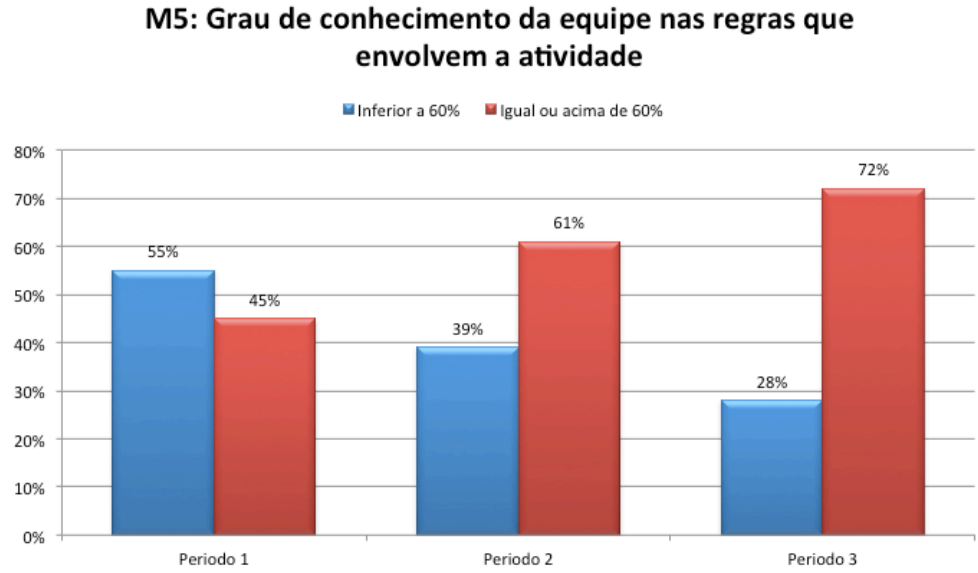

Figura 5-5 – Resultado da métrica 5

Assim o Objetivo 1 foi atendido parcialmente, já que a métrica M3, não teve o seu resultado alcançado.

Já o Objetivo 2 foi analisado através de cada uma de suas métricas, a primeira métrica do objetivo 2 é a métrica M6, ilustrada na Figura 5-6.

Durante as reuniões entre a gerência, os coordenadores e os analistas de qualidade surgiram questionamentos a respeito do nível de qualidade do produto que estava sendo desenvolvido. Segundo relatos registrados na ferramenta de *bug tracking*, era necessário uma inspeção do código que estava sendo produzido, já que o cliente reclamava de vários defeitos.

Para um melhor entendimento, um defeito é considerado trivial quando esse problema não afeta o uso do sistema. O defeito considerado como importante é quando existe alguma modificação no comportamento do sistema, entretanto o usuário consegue finalizar uma operação. O defeito crítico é quando o usuário não consegue utilizar o sistema.

Na primeira coleta realizada o indicador “Normal“ registrou 25%, o indicador “Atenção“ registrou 19% e o indicador “Impeditivo“ registrou 56%. O resultado não estava de acordo com o esperado pela gerência, vide

Quadro 5-8.

Para alcançar o resultado esperado, recomendou-se, que os desenvolvedores leiam os requisitos com mais atenção, a fim de gerar um código-fonte equivalente com que foi escrito. Adicionalmente, realizar testes de unidade com o objetivo de diminuir problemas de implementação.

Recomenda-se, que a equipe de arquitetura ministre treinamentos de teste de unidade para a equipe de desenvolvimento. O objetivo é o aumento da habilidade na construção de testes unitários abrangendo todos os fluxos de um requisito.

Na segunda coleta realizada o indicador “Normal“ aumentou de 25% para 41%, o indicador “Atenção“ passou de 19% para 23% e o indicador “Impeditivo“ reduziu de 56% para 36%. Verificou-se um ganho na qualidade do produto, entretanto, este resultado ainda não estava de acordo com o esperado.

Foi intensificado o treinamento técnico explicando sobre os *frameworks* utilizados e treinamentos sobre as regras e fluxos de trabalho do cliente, que auxiliaram a equipe de desenvolvimento para um melhor entendimento do sistema.

Na terceira coleta pode-se verificar um aumento significativo no indicador “Normal” que aumentou de 41% para 71%, o indicador “Atenção“ teve uma redução de 23% para 19% e o indicador “Impeditivo” passou de 36% para 10%. O resultado esperado pela gerência foi alcançado.

Com este resultado é possível verificar que os treinamentos técnicos e sobre as regras e fluxos de trabalho do cliente contribuíram para alcançar a meta com sucesso. Como resultado tem-se a melhoria da compreensão da solução técnica e do negócio do sistema.

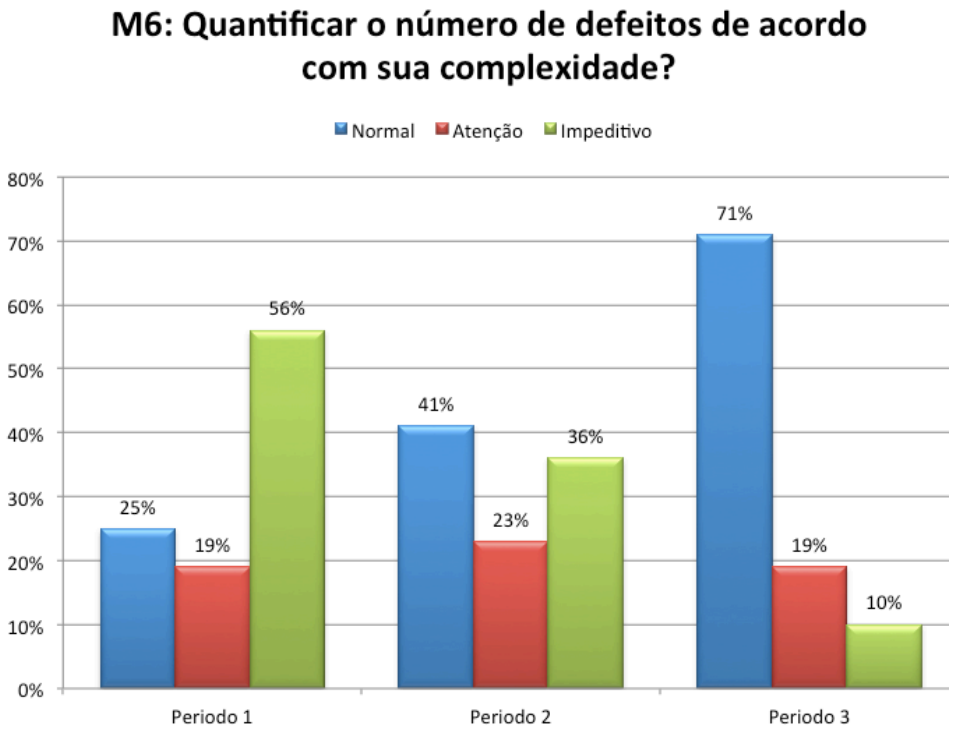

Figura 5-6 – Resultado da métrica 6

A próxima métrica a ser analisada é a M7, ilustrado na Figura 5-7. O objetivo desta métrica é verificar se o processo que está sendo executado está de acordo com o que foi definido e institucionalizado.

Na primeira coleta realizada, verificou-se que o indicador “Contempla“ registrou a marca de 37%, o indicador “Contempla em partes“ registrou 48% e o indicador “Não contempla“ registrou 15%. Este resultado não estava de acordo com o resultado esperado pela gerencia do projeto, vide Quadro 5-9.

Infere-se, de acordo com os resultados obtidos, que o indicador “Contempla em Parte“ e “Não contempla” foi apresentado com um alto percentual, pois possivelmente nem todos os desenvolvedores realizavam a atividade de testes de unidade e realizavam o comentário no código. As atividade de testes de unidade auxiliam na redução de defeitos no software e os comentários auxiliam na manutenção do sistema.

Recomenda-se que a equipe de arquitetura ministre um treinamento para a equipe de desenvolvimento sobre as boas práticas de programação, apresentando exemplos de como escrever comentários no código-fonte, além de dicas para otimizar a programação, contribuindo assim para facilitar uma manutenção no sistema.

Na segunda coleta realizada, verificou-se que o indicador “Contempla” aumentou de 37% para 79%, o indicador ”Contempla em partes” registrou uma redução de 48% para 16%,

enquanto que o indicador “Não contempla“ registrou uma redução de 15% para 5%. O resultado estava conforme o esperado pela gerência.

Infere-se que, de acordo com o resultado, os treinamentos ministrados pela equipe de arquitetura, fixaram boas práticas para a equipe de desenvolvimento, a qual gerou um código-fonte mais legível para uma futura manutenção.

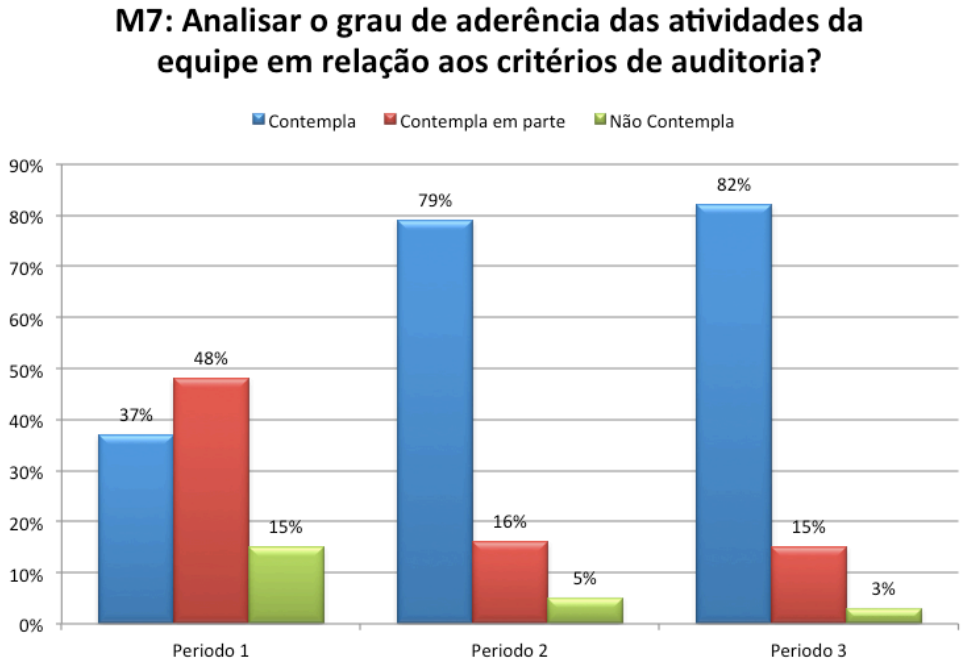

Figura 5-7 – Resultado da métrica 7

A próxima métrica a ser analisada é M8, ilustrado na Figura 5-8. O objetivo desta métrica é verifica se o que está sendo produzido em software corresponde ao que foi elicitado em requisitos.

Na primeira coleta realizada verificou-se que o indicador “Normal“ registrou 46%, o indicador “Aceitável“ registrou 31% e indicador crítico pontuou 23%. Este resultado não estava de acordo com o esperado pela gerência e equipe de coordenação, vide Quadro 5-10.

A partir do resultado, infere-se que o indicador “Aceitável“ e o indicador “Crítico“ registraram esta pontuação, possivelmente, devido a equipe de desenvolvimento não estar efetuando uma leitura com interpretação adequada. Isto pode ocorrer, por exemplo, pela falta de conhecimento do *template* utilizado.

Recomenda-se que a equipe de requisito faça uma revisão dos insumos gerados e posteriormente, realize uma apresentação para a equipe de desenvolvimento, sobre como ler corretamente os Documentos de Caso de Uso e Regra de Negócio, baseado nos *templates* destes documentos.

Após a inspeção dos requisitos, foi realizada uma nova coleta, onde o indicador "Normal" teve um aumento de 46% para 74%, o indicador "Aceitável" teve uma redução de 31% para 17%, enquanto que o indicador "Crítico" teve uma redução de 23% para 9%. Este resultado não estava de acordo com o esperado pela gerência e equipe de coordenação.

É possível inferir que, de acordo com o percentual dos indicadores "Aceitável" e "Crítico", a equipe de desenvolvimento estava com dificuldades para realizar a leitura dos artefatos gerados pela equipe de requisitos.

Recomenda-se realizar uma atualização nos *templates* dos artefatos gerados pela equipe de requisitos. O objetivo desta atualização é reduzir as ambiguidades destes documentos, a fim de facilitar o entendimento dos desenvolvedores, para que as inconsistências na codificação sejam reduzidas. Adicionalmente, sugere-se que a equipe de arquitetura realize uma inspeção no código-fonte com o objetivo de verificar se os padrões de desenvolvimento estão sendo atendidos.

Na terceira coleta foi possível verificar que o indicador "Normal" aumentou de 74% para 92%, o indicador "Aceitável" reduziu de 17% para 6% e o indicador "Crítico" reduziu também de 9% para 2%. O resultado obtido estava de acordo com o esperado pela gerência.

Infere-se que a atualização nos *templates* dos documentos gerados pela equipe de requisitos, assim como as inspeções realizadas pela equipe de arquitetura no código-fonte, auxiliaram para que o resultado desta métrica fosse positivo.

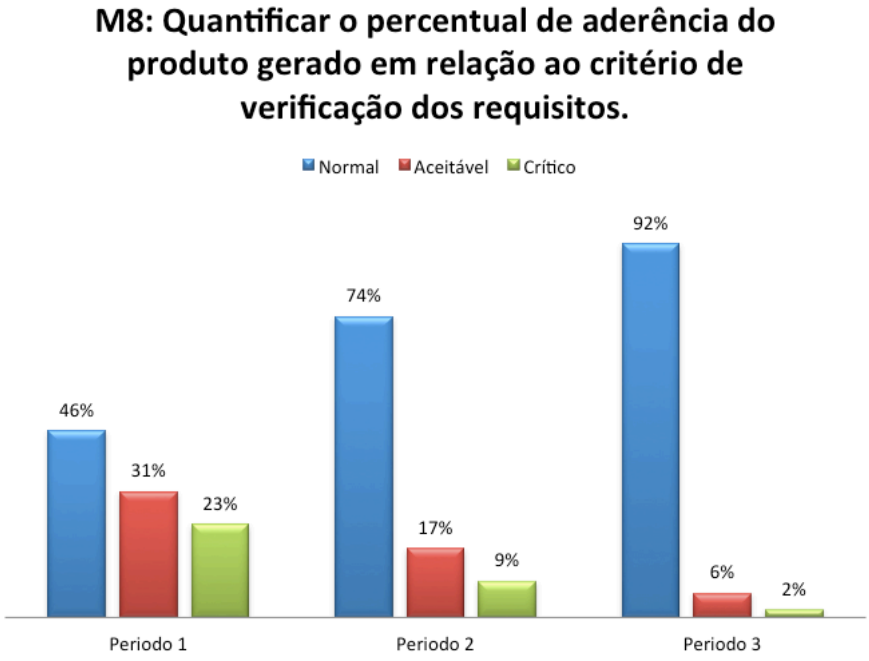

Figura 5-8 – Resultado da métrica 8

A próxima métrica a ser analisada é a M9, ilustrado na Figura 5-9. O objetivo desta métrica é verificar se a equipe de desenvolvimento está implementando os testes de unidades de acordo com o que foi definido nos fluxos principais e alternativos do documento de requisitos. A abrangência do teste de unidade, de acordo com o código fonte, garante que as regras de negócio estejam de acordo com o que foi especificado.

Na primeira coleta realizada pode se verificar que o indicador “Normal“ registrou 34%, o indicador “Aceitável“ registrou 27% e indicador “Crítico” pontuou 39%. Este resultado não estava de acordo com o esperado pela equipe de coordenação, vide Quadro 5-11.

De acordo com os resultados coletados, infere-se que o indicador “Crítico“ registrou um percentual alto, possivelmente devido aos problemas de interpretação nos documentos gerados pela equipe de requisito e também a não execução da atividade de teste de unidade.

Recomenda-se que a equipe de arquitetura ministre treinamentos sobre boas práticas de programação para a equipe de desenvolvimento, a fim de que a equipe de desenvolvimento consiga produzir de forma mais rápida e um código mais fácil de manutenção.

Na segunda coleta foi realizada verificou-se que o indicador “Normal“ registrou um aumento de 34% para 82%, o indicador “Aceitável“ teve uma redução de 27% para 15%, enquanto que o indicador “Crítico” reduziu de 39% para 3%. O resultado obtido não era o esperado pela equipe de coordenadores e gerentes.

Após a realização da auditoria, foi verificado que a equipe de desenvolvimento estava executando corretamente a atividade de criação de teste de unidade. Adicionalmente, de acordo com as melhorias realizadas na métrica M8, os *templates* dos artefatos foram alterados, visando uma melhor legibilidade e menos ambuiguidade.

Na terceira coleta realizada verificou-se que o indicador “Normal“ teve um aumento de 82% para 86%, o indicador “Aceitável” reduziu de 15% para 11% e o indicador “Crítico” permaneceu em 3%. O resultado esperado foi alcançado.

Infere-se que o resultado esperado foi alcançado devido a mudança nos *templates* dos insumos gerados pela equipe de requisitos. Essas mudanças facilitaram um melhor entendimento dos artefatos, gerando falhas de interpretação.

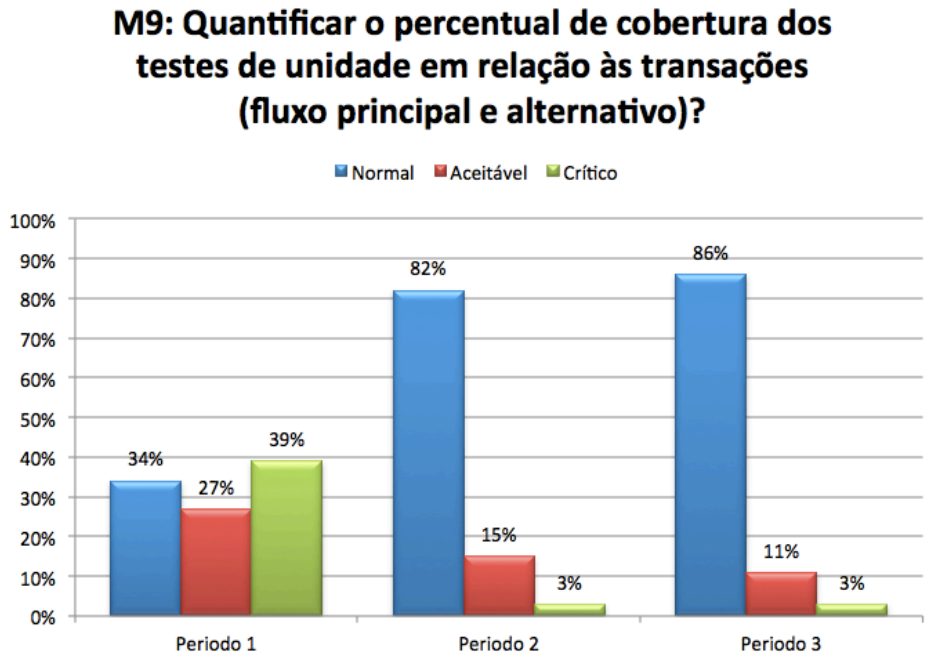

Figura 5-9 – Resultado da métrica 9

A próxima métrica a ser analisada é M10, ilustrada na Figura 5-10. O objetivo desta métrica é verificar se o que a equipe de desenvolvimento está implementando está de acordo com o padrão de interface definido.

Na primeira coleta realizada pode se verificar que o indicador "Normal" registrou 39%, o indicador "Aceitável" registrou 27% e indicador crítico pontuou 34%. Este resultado não estava de acordo com o esperado pela equipe de coordenação, vide Quadro 5-12.

De acordo com os resultados obtidos, o indicador "Crítico" registrou o percentual elevado, devido a realização de poucos treinamentos de padrões de interface ministrados.

Recomenda-se que a equipe de interface ministre *workshops* apresentando o padrão de interface para a equipe de desenvolvimento, a fim de demonstrar as boas práticas de implementação dos componentes de interface, para que as funcionalidades estejam de acordo com o padrão de interface institucionalizado.

Na segunda coleta realizada verificou-se que o indicador "Normal" registrou um aumento de 39% para 71%, o indicador "Aceitável" teve um aumento de 27% para 21%, enquanto que o indicador "Crítico" reduziu de 34% para 8%. O resultado esperado pela gerência, não foi alcançado.

Infere-se que a equipe de desenvolvimento não estava desenvolvendo corretamente os testes de unidade. Apesar da significativa melhora nos resultados coletados, foi verificado que

só o *workshop* não era suficiente para alcançar o resultado esperado, então acordou-se realizar um treinamento de 24 horas em laboratório com toda a equipe de desenvolvimento.

Na terceira coleta realizada verificou-se que indicador "Normal" teve um aumento de 71% para 92%, o indicador "Aceitável" reduziu de 21% para 4% e o indicador "Crítico" reduziu de 8% para 4%. O resultado esperado foi alcançado.

Infere-se que, de acordo com a coleta dos dados da terceira pesquisa, o treinamento de 24 horas foi suficiente para alcançar o resultado esperado, já que os desenvolvedores estavam desenvolvendo as funcionalidades baseadas no padrão de interface.

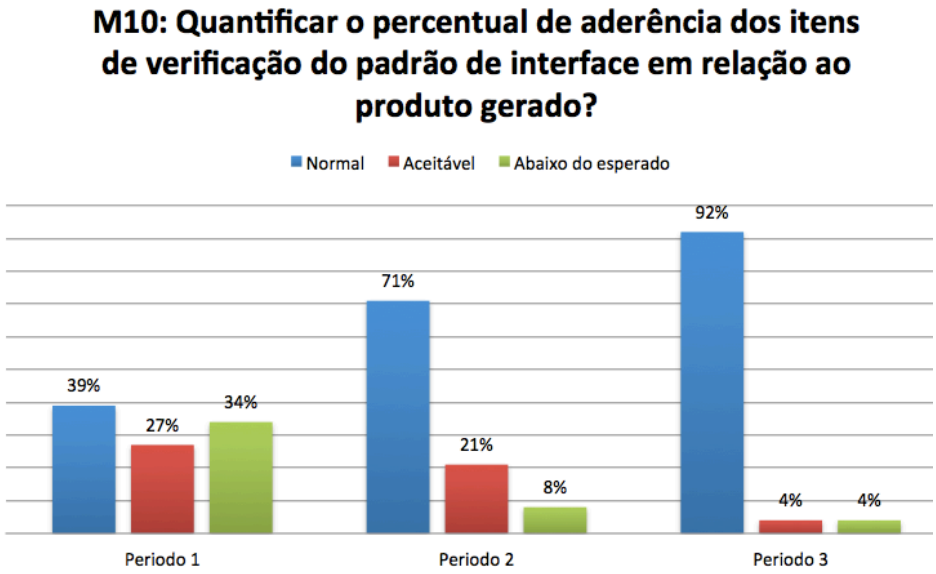

Figura 5-10 – Resultado da métrica 10

Após a análise das métricas acima, verificou-se que o Objetivo 2 foi totalmente atendido, já que todas as metas definidas pela gerência do projeto foram alcançadas.

O objetivo 3 possui apenas uma métrica que é a M11, ilustrado Figura 5-11. O objetivo desta métrica, solicitada pelos os coordenadores, era realizar uma verificação se o que é estimado pela gerência está de acordo com o que foi especificado pelo desenvolvedor. O motivo é que segundo os coordenadores, possivelmente os gerentes subestimavam o tempo de execução das atividades do desenvolvimento.

Na primeira coleta foi verificado que o indicador "Superestimado" registrou o percentual de 62%, o indicador "Normal" registrou 23% e o indicador "Subestimada" marcou 15%. O resultado esperado pelos coordenadores não foi alcançado, vide Quadro 5-13.

Infere-se que as estimativas realizada pelos gerentes, das atividades não estão acontecendo de maneira correta, por que os gerentes não utilizam métricas de estimativas confiáveis.

Recomenda-se a realização de treinamentos de estimativa, com uma carga horária mínima de 8 horas, já que as estimativas fornecem dados que permitem, com margem de erro, prever a quantidade de pessoas e o tempo necessário para realizar cada tarefa do cronograma, assim como os custos do projeto. Não é aconselhável elaborar cronograma e orçamento sem o uso de estimativas.

Na segunda coleta dos resultados, verificou-se que o indicador “Superestimada” registrou uma redução de 62% para 36%, o indicador “Normal” registrou um aumento de 23% para 57% e o indicador “Subestimada” teve uma redução de 15% para 7%. O resultado esperado não foi alcançado.

Infere-se que de acordo com os treinamentos sobre estimativas, os tempos estimados pelos gerentes e pelos desenvolvedores estavam se aproximando ao realizar a estimativa de uma determinada tarefa.

Recomenda-se a realização de treinamentos de estimativa, com a carga horária de 24 horas, já que as estimativas fornecem dados que permite prever a quantidade de pessoas e o tempo necessário para realizar cada tarefa do cronograma, assim como os custos do projeto.

Adicionalmente, recomenda-se realizar treinamento sobre os fluxos de negócio do sistema, com o objetivo dos gerentes e desenvolvedores conseguirem um melhor entendimento do sistema. O resultado é que, com o melhor entendimento do sistema, a probabilidade do tempo de execução das atividades estarem dentro das estimativas realizadas é maior.

A terceira coleta evidenciou que o indicador “Superestimada” reduziu de 36% para 16%, o indicador “Normal” registrou um aumento de 57% para 81% e o indicador “Subestimada” aumentou de 7% para 3%. O resultado esperado foi alcançado.

Infere-se que os treinamentos sobre os fluxos de negócio do sistema auxiliaram em um melhor entendimento do sistema. Além do treinamento de estimativas, que também contribuiu para que o resultado fosse um resultado aproximado do tempo previsto pela equipe de desenvolvimento.

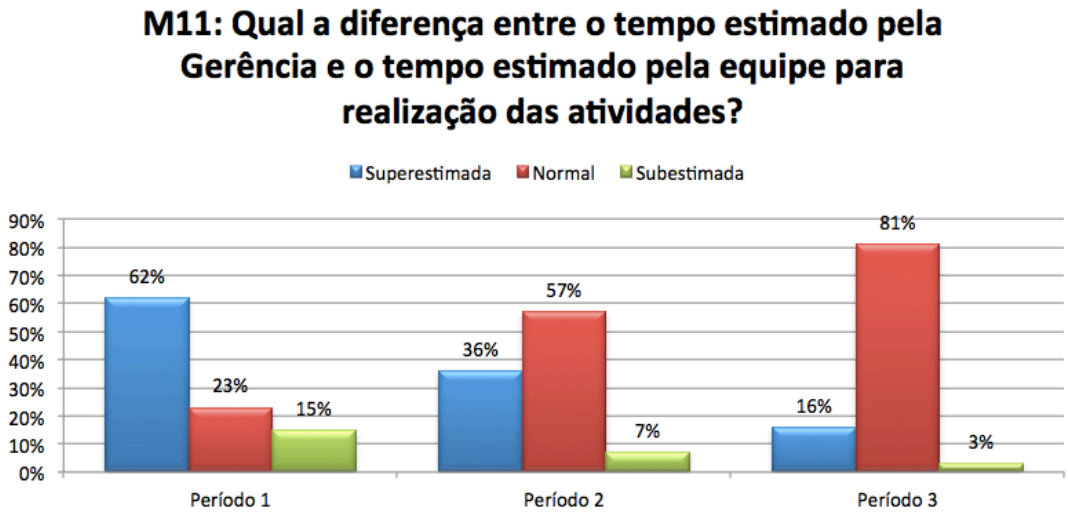

Figura 5-11 – Resultado da métrica 11

Após a análise das métricas acima, verificou-se que o Objetivo 3 foi totalmente atendido, já que a métrica definida pela gerência do projeto, foi alcançada.

Assim infere-se que, de acordo com os dados da Quadro 5-14, houve melhorias no processo de PCP Versão 2 e consequentemente a possibilidade de um produto de software com mais qualidade está sendo gerado. Entretanto, é importante que o processo de PCP Versão 2 seja constantemente avaliado para garantir o nível de qualidade desejável do produto.

Quadro 5-14 - Resultado final de acordo com os objetivos

Objetivos	Situação	Recomendação
1	Atende parcialmente	Espera-se que o objetivo da Métrica M3, seja alcançada, após a implantação do controle de horário.
2	Atende completamente	Não se aplica
3	Atende completamente	Não se aplica

5.3 Avaliação Qualitativa

A avaliação qualitativa ocorreu através de uma entrevista, utilizando como ferramenta um questionário. Esta avaliação foi realizada, como forma de complemento a avaliação quantitativa, já que as informações coletadas na avaliação qualitativa não eram suficientes.

A coleta de dados ocorreu no período de 22 de fevereiro de 2011 a 1 de março de 2011 como forma de verificar as evoluções do processo de PCP Versão 2. A entrevista foi realizada

com os membros da equipe de desenvolvimento, no total de 14 (quatorze) pessoas. As informações sobre o questionário, encontram-se no Apêndice C.

Como forma de aplicar o conhecimento discutido nas seções anteriores, as seções seguintes permitem uma analise qualitativa das medições propostas para atender o cenário da fábrica de software relatado no capítulo 4.

5.3.1 **Definição**

A metodologia utilizada para realizar a avaliação qualitativa, se deu através de entrevista usando como instrumento um questionário. Para criação do questionário foram necessárias duas reuniões com a equipe de gerentes do projeto e o coordenador da área de desenvolvimento. O objetivo era realizar o levantamento de outras necessidades a qual a avaliação quantitativa não abrangia.

Para melhor organização, este questionário foi divido em 3 (três) seções:

- Perfil dos participantes;
- Experiência dos participantes;
- Perguntas referentes as atividades do processo de desenvolvimento da fábrica de software.

A primeira e a segunda seção do questionário são importantes para definir o perfil e a experiência de cada participante, já que para validar os resultados obtidos com o questionário aplicado é necessário que o participante possua conhecimentos em codificação, tenha no mínimo um conhecimento superficial sobre a Engenharia de Software e Qualidade de Software, assim como, já tenha participado de pelo menos um projeto de médio porte. Esses requisitos foram definidos com o objetivo de verificar o conhecimento dos participantes sobre o assunto.

Como forma de avaliação do processo de PCP Versão 2, a terceira seção do questionário apresenta todas as atividades do processo de desenvolvimento da fábrica. A finalidade de avaliar cada atividade é verificar se o processo de PCP Versão 2 precisa de alguma melhoria, de acordo com a visão dos entrevistados.

A resposta dos participantes no questionário era baseado em três critérios de avaliações, que são:

- **Presença**: se uma determinada atividade está presente e é executada no Processo de Desenvolvimento de Software apresentado;

- **Utilidade**: se uma determinada atividade tem utilidade no processo de Desenvolvimento de Software;
- **Adequação**: se uma determinada atividade necessita de alguma adequação para o processo de Desenvolvimento de Software.

Para facilitar o entendimento dos resultados, foi necessário o agrupamento de cada um dos critérios citados, conforme o Quadro 5-15:

Quadro 5-15 – Critérios para agrupamento do questionário

Presença	Utilidade	Adequação do nível de detalhamento
1. Não é oferecido pelo processo e não gostaria de tivesse disponível.	1. Não é útil.	1. O detalhamento deve ser aumentado.
2. Não oferecido, mas gostaria que tivesse disponível.	2. Provavelmente é útil, mas ainda não apliquei.	2. O detalhamento não precisa ser modificado.
3. Oferecido, parcialmente.	3. É útil e já apliquei em diferentes implementações.	3. O detalhamento deve ser diminuído.
4. Oferecido.		

Após a definição dos critérios do questionário, foi realizada uma inspeção, com o objetivo de encontrar defeitos. Assim foi distribuído para revisão a três pessoas que tinham algum conhecimento em desenvolvimento, entretanto, as mesmas não participavam da atividade de codificação na fábrica. Posteriormente, foi realizada a coleta dos resultados do questionários para avaliar possíveis correções.

Após as correções o questionário estava pronto para ser distribuído para a equipe de desenvolvimento da fábrica. Antes da distribuição do questionário, a equipe de desenvolvimento, composta por 14(quatorze) pessoas, foi convidada a participar de uma explicação, através de um apresentação do questionário, o seu objetivo, o termo de confidencialidade e apresentação do processo de PCP Versão 2.

Os questionários foram distribuídos para os arquitetos do software, analistas de banco de dados, analista de interface e desenvolvedores. O tempo estimado para responder todo o questionário foi de 25 (vinte e cinco) minutos.

Todos os questionários foram recolhidos e seus dados foram tabulados em uma planilha para realizar a análise. Esses dados encontram-se no Apêndice D.

5.3.2 **Análise**

A partir dos dados coletados no questionário é possível verificar o perfil dos participantes. Verificou-se que os participantes em sua maioria são graduados em Informática/

Ciência da Computação e o restante é graduado na área de Engenharia, conforme apresentado na Figura 5-12.

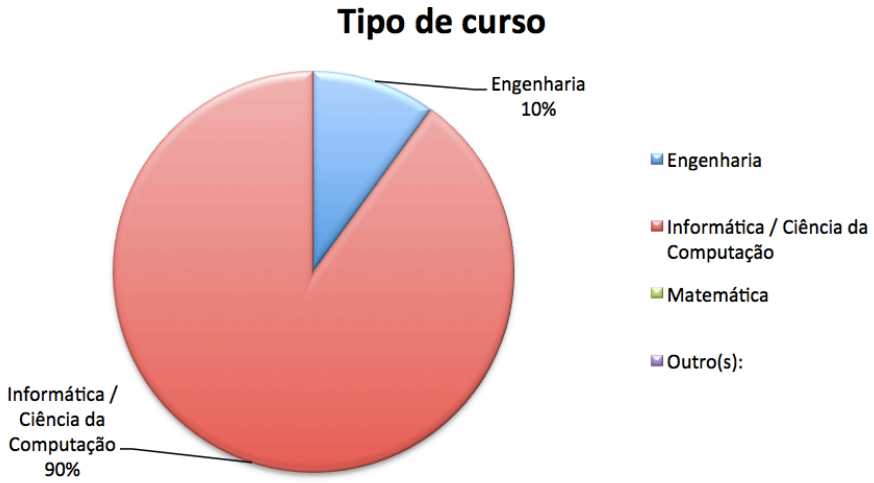

Figura 5-12 – Resultado da pergunta: Tipo de Curso

Conforme ilustrado na Figura 5-13, a metade dos participantes responderam que possuem pós graduação, enquanto a outra metade possui apenas graduação.

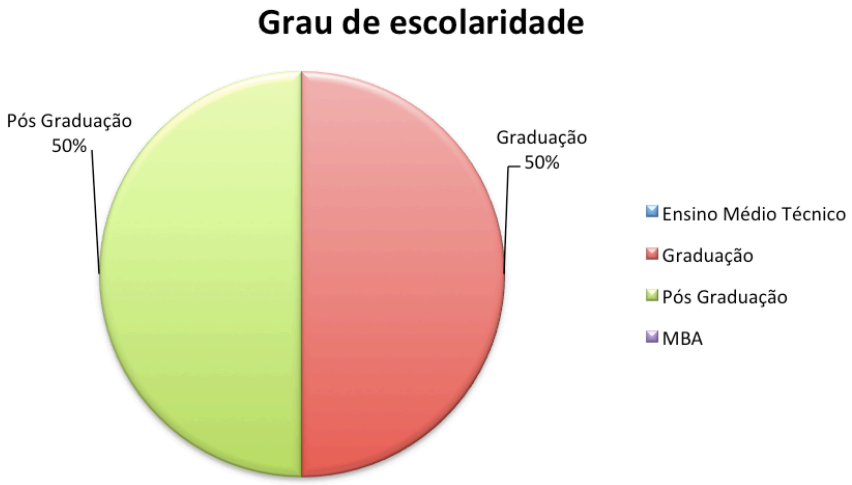

Figura 5-13 – Resultado da pergunta: Grau de escolaridade

De todos os entrevistados, a maioria respondeu que tem um alto conhecimento sobre desenvolvimento de software, seguido por bom, excelente e médio. Infere-se que a maioria da equipe já possui alguma experiência em desenvolvimento de produto de software, conforme a Figura 5-14.

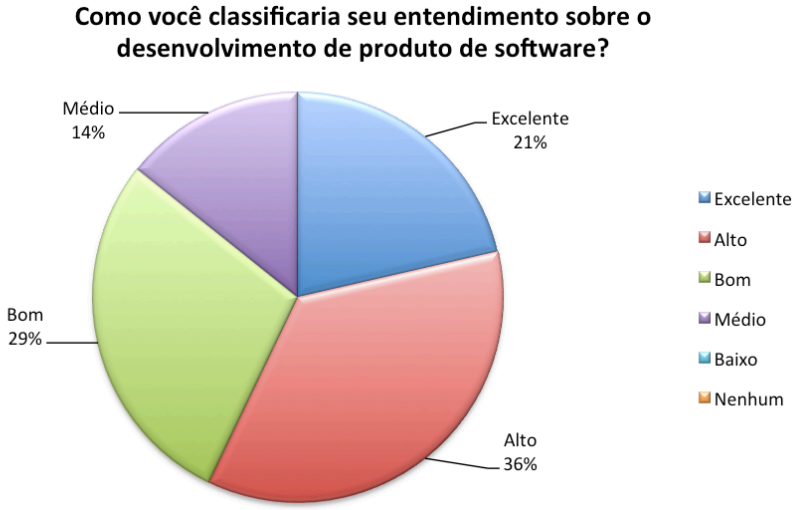

Figura 5-14 – Resultado da pergunta: Como você classificaria seu entendimento sobre o desenvolvimento de produto de software.

De acordo com a pesquisa, verificou-se que a maioria dos participantes já exerceu ou exerce atividades de codificação e o restante já exerceu ou exerce o cargo de analista de sistemas. Infere-se que a maioria dos participantes possui experiência anterior em codificação, área em que trabalham atualmente na empresa, conforme a Figura 5-15.

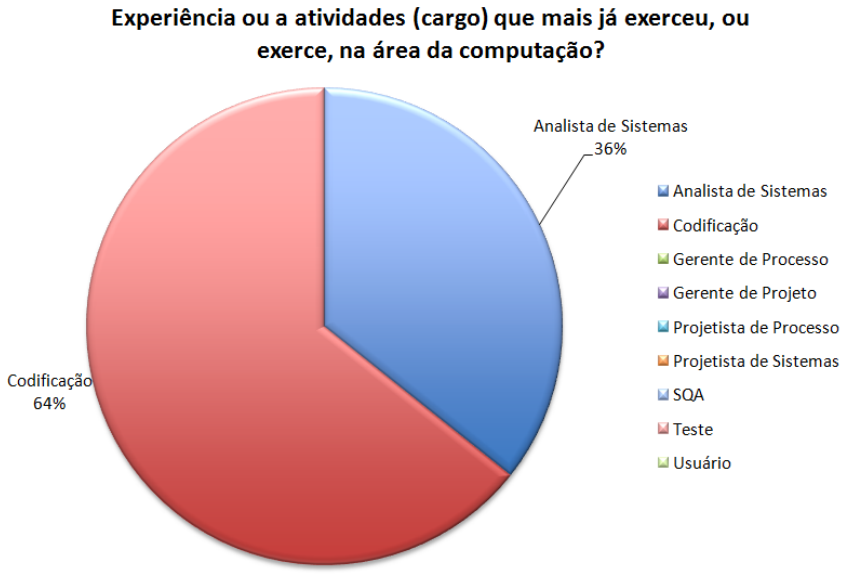

Figura 5-15 – Resultado da pergunta: Experiência ou a atividades (cargo) que mais já exerceu, ou exerce, na área da computação

Dos participantes entrevistados, verificou-se que a maioria já trabalhou em sistemas de grande porte, seguido por sistemas de médio porte. Infere-se que a maioria dos participantes já realizou atividades de codificação em projetos de grande porte, conforme a Figura 5-16.

Figura 5-16 – Resultado da pergunta: Já participou do desenvolvimento de que tipo de sistemas de computação?

Segundo as respostas da maioria dos participantes, verificou-se que já atuaram na área de codificação, e os demais atuaram na área de projetos de sistemas. Infere-se que esses participantes possuem conhecimento na atividade de implementação.

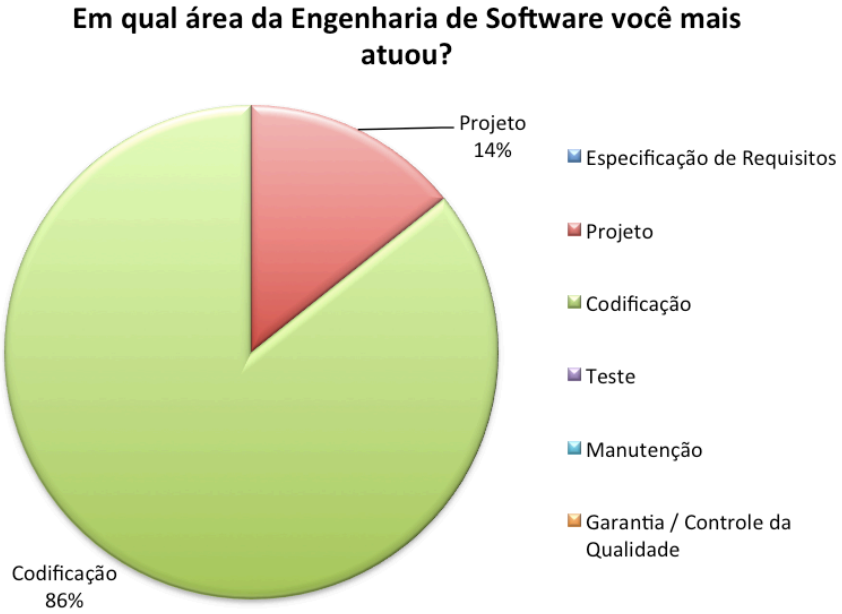

Figura 5-17 – Resultado da pergunta: Em qual área da Engenharia de Software você mais atuou

A maioria dos desenvolvedores relataram que o seu conhecimento em desenvolvimento de software é alto, seguido pelos indicadores bom e excelente. Verificando-se assim, que já tiveram experiências anteriores e que conhecem a atividade de codificação de produtos de software, segundo a Figura 5-18.

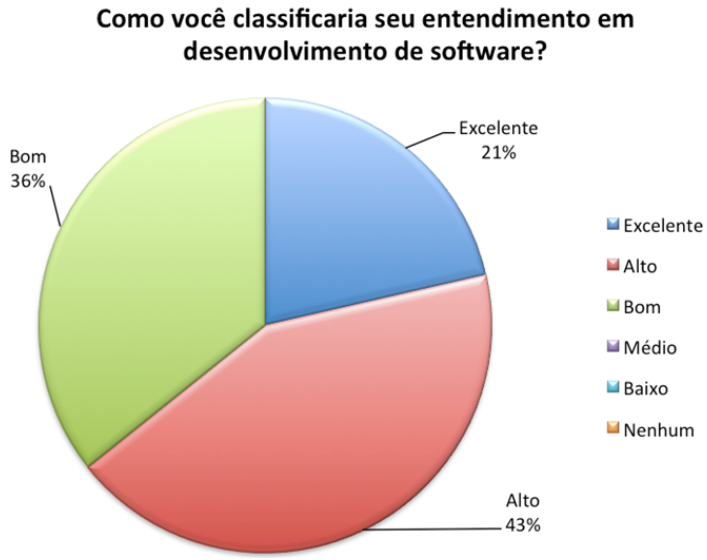

Figura 5-18 – Resultado da pergunta: como você classificaria seu entendimento em desenvolvimento de software?

Quanto ao conhecimento sobre qualidade de software, a maioria dos participantes já participou de pelo menos um projeto que possuía um processo de software bem definido. Os demais utilizaram pelo menos 3 (três) processos de software. Infere-se que os participantes tem uma experiência razoável em processos de software, conforme a Figura 5-19.

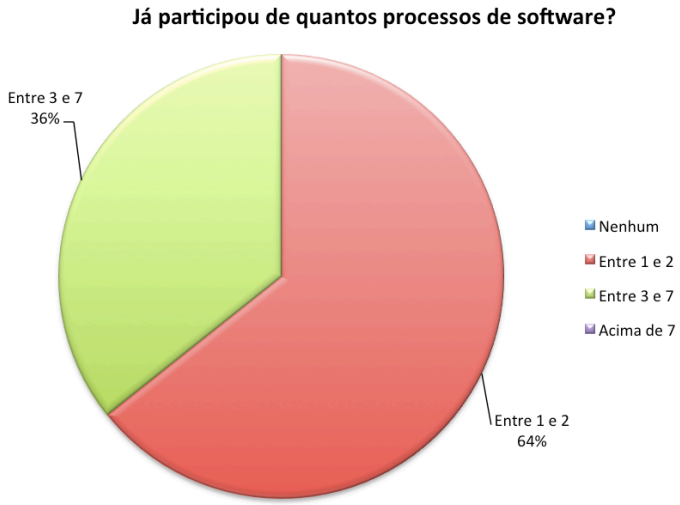

Figura 5-19 – Resultado da pergunta: Já participou de quantos processos de software

A maioria dos participantes relatou que nunca definiu processo de software, enquanto que 21% relatou que já definiu pelo menos 1 e o restante já realizou a definição de pelo menos 3 processos de software. Infere-se que os participantes que nunca definiram um processo de software, possivelmente, não participaram de grupos como do SEPG. Para o restante, pode-se considerar que houve a participação no grupo de SEPG, já que o perfil dos participantes não apresentou experiência na área de qualidade conforme apresentado na Figura 5-20.

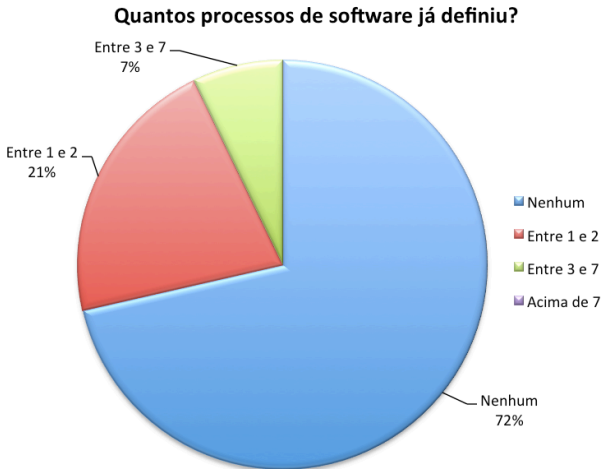

Figura 5-20 – Resultado da pergunta: Quantos processos de software já definiu.

A maioria dos participantes relatou que já utilizou pelo menos uma vez modelo/normas de qualidade, enquanto que 29% relatou que nunca utilizou e o restante utilizou de três a sete vezes modelos/normas da qualidade. Infere-se que ainda são poucas as empresas que trabalham com modelos de qualidade. Conforme apresentado na Figura 5-21.

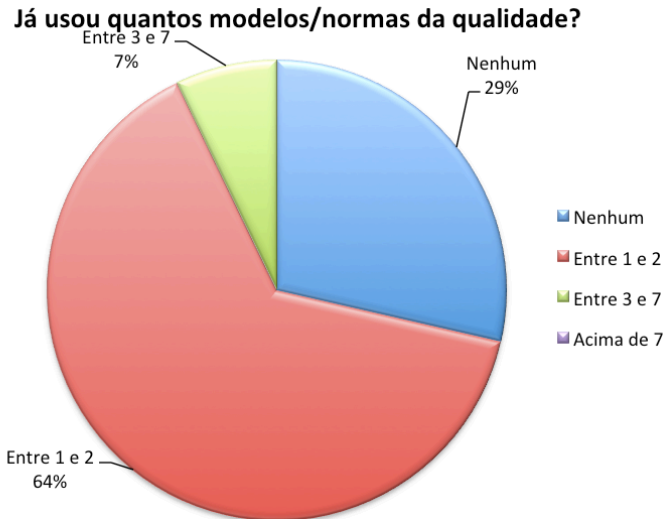

Figura 5-21 – Resultado da pergunta: Já usou quantos modelos/normas da qualidade.

Conforme os critérios definidos para o questionário, de acordo com a Figura 5-22, foi constatado que 66% das atividades do processo eram executadas normalmente, que 25% das atividades eram oferecidas de forma parcial e 8% relataram que não eram oferecidas.

Infere-se que as atividades que eram oferecidas de forma parcial, não eram executadas devido ao cronograma de curto prazo, pois existiam atividades de maior prioridade, como por exemplo: *Mentoring*, planejamento de banco de dados e referências ao diagrama navegacional no requisito.

Conclui-se que, aos que responderam que não era oferecido ou oferecido parcialmente, ainda não participaram de nenhum *mentoring*, ou da criação do planejamento do banco de dados e ainda receberam nenhum requisito que contenha o diagrama navegacional referenciado.

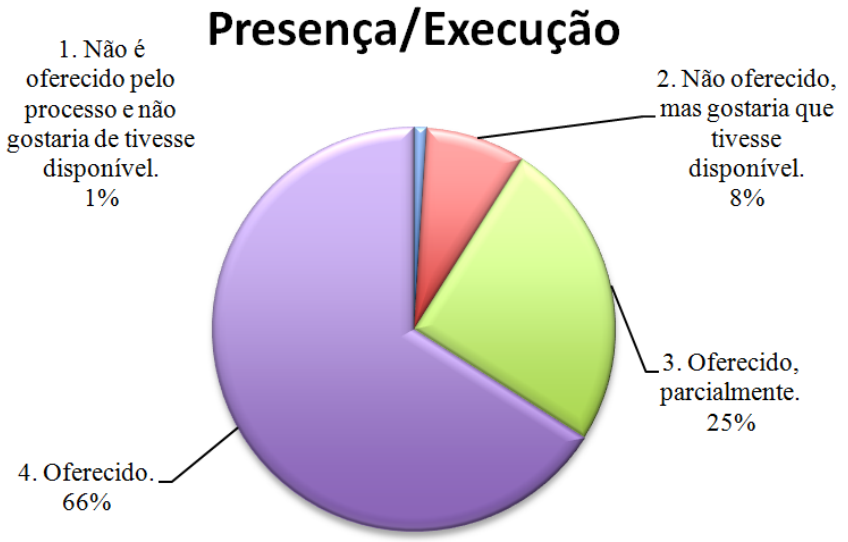

Figura 5-22 – Resultado da pergunta: Presença/Execução

Verifica-se na Figura 5-23, que 80% das atividades possuem utilidade ao desenvolvimento do projeto. O restante são atividades consideradas úteis, entretanto, ainda não foram utilizadas pelos participantes.

Pode-se inferir que, a maioria dos desenvolvedores ainda não participou de certas atividades no processo, como por exemplo, Validar Requisitos, Liberar TAG de Requisitos, Projetar Testes e Especificar Diagrama Navegacional, entretanto, acham útil para o processo, mesmo sem terem aplicado. Ainda não houve participação da equipe de desenvolvimento nestas atividades devido serem exclusivamente executadas em outras áreas da fábrica de software, ou seja, fora da área de desenvolvimento.

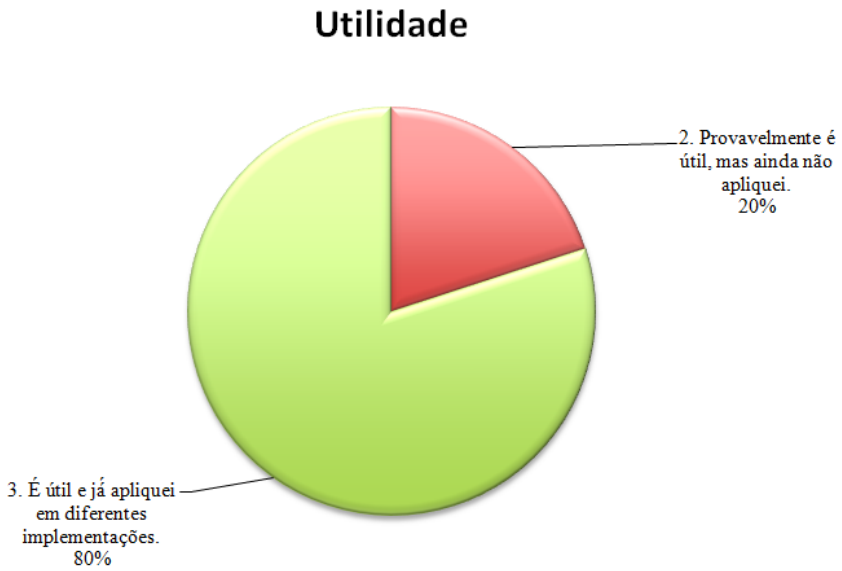

Figura 5-23 – Resultado da pergunta: Utilidade

A Figura 5-24 apresenta que 56% das atividades não precisam ter o seu detalhamento modificado. Isto demonstra que as atividades do processo não precisam de qualquer alteração. Outra parte dos participantes, 40% das respostas, responderam que o detalhamento das atividades precisa de melhoria, como por exemplo: Especificar Requisitos, Validar Requisitos, Liberar TAG de Requisitos, Corrigir Ajustes no Requisito, Projetar Testes, Especificar Protótipo, Gerar Build. Pode-se inferir, que os colaboradores não conhecem detalhadamente como essas atividades são executadas, o motivo pode ser que algumas destas atividades ocorrem em áreas que apoiam o desenvolvimento.

Apenas 4% dos participantes informaram que algumas atividades precisam ter o seu detalhamento diminuído. Estas atividades são, por exemplo, Registrar Ajustes Requisitos, Registrar Ajustes Código, Executar Testes de Unidade, Implementar Interface de Integração e Realizar um *Mentoring* com a Equipe. Pode se inferir que por não necessitarem de uma grande granularidade de detalhamento, esses participantes conhecem com mais detalhes cada uma das atividades citadas, considerando assim que a granularidade atualmente utilizada é demasiada.

Figura 5-24 – Resultado da pergunta: Adequação do nível de detalhamento

Após a conclusão desta avaliação, foi possível realizar uma análise através dos dados coletados e transformá-los em tabelas de percentuais, que encontram-se no Apêndice D. Adicionalmente, foram gerados gráficos para auxiliar análise a e verificar quais os ajustes que precisavam ser realizado no processo de PCP Versão 2.

É importante ressaltar que, através da avaliação qualitativa, não se colhe dados quantificáveis, mas sim particularidades e interpretações individuais, podendo ser úteis na evolução e melhoria do processo.

O resultado obtido apresenta que a maioria dos participantes da pesquisa informaram que as atividades estão definidas e institucionalizadas para toda a fábrica de software. Adicionalmente, a maioria dos entrevistados respondeu que as atividades do processo tem utilidade assim como o detalhamento das atividades não precisa ser modificado.

Infere-se que o processo de PCP Versão 2, após sua evolução, atende de forma positiva as necessidades para a execução do processo de PCP.

5.4 Considerações Finais

Para que um produto de software possa ter qualidade, é preciso uma inspeção constante para adaptação constante, ou seja, mesmo após a evolução do processo de PCP Versão2, é possível detectar novas oportunidades de melhorias.

Como forma de analisar as mudanças, foi sugerido a realização de avaliações qualitativas e quantitativas. O objetivo destas avaliações era coletar dados que servissem de insumo para as propostas de evolução do processo de PCP Versão 2.

As avaliações Quantitativas e Qualitativas oferecem perspectivas diferentes, mas não opostas. De fato, representam abordagens que podem ser utilizadas em conjunto, de acordo com a necessidade em questão, obtendo assim informações mais consistentes do que se fossem realizadas isoladamente.

Os dados coletados nestas avaliações confirmam a necessidade de novas melhorias no processo de PCP Versão 2.

Capítulo 6 –Conclusões e Trabalhos Futuros

O objetivo desta dissertação de mestrado foi apresentar o relato da institucionalização do processo de Projeto e Construção do Produto em uma fábrica de software. Neste trabalho foi apresentada uma das formas para realizar a criação de um processo de PCP utilizando a linguagem de notação do BMPN. Adicionalmente, foi necessário realizar uma avaliação no processo utilizando como metodologia a abordagem de Pontos Fortes, Pontos Fracos e Oportunidades de Melhoria.

Diante deste contexto, verificou-se que o processo de PCP utilizado não atendia mais a realidade da organização, então fez-se necessário realizar uma evolução no processo de PCP Versão 1, que foi chamado de processo de PCP Versão 2.

Após a evolução do processo de PCP realizou-se uma avaliação, a qual ocorreu de duas formas: quantitativa e qualitativa. A avaliação quantitativa foi realizada através de auditoria no repositório de ativos organização utilizando o *framework* GQM, já a avaliação qualitativa ocorreu através de questionários que foram distribuídos aos participantes da equipe de desenvolvimento.

Este capítulo está divido em 5 (cinco) partes, a seção 6.1 apresenta o sumário do trabalho, na seção 6.2 é apresentada a análise dos resultados obtidos, na seção 6.3 são apresentados possíveis trabalhos futuros, na seção 6.4 são relatadas as limitações da pesquisa.

6.1 Sumário do Trabalho

Esta dissertação apresentou o relato do estudo na melhoria do processo de Projeto e Construção do Produto. Como forma de validar este relato, foi realizado um estudo de caso em uma organização. Este trabalho iniciou-se através de um estudo na literatura, o qual foi apresentado no Capítulo 2 através de informações sobre o processo de desenvolvimento de software. No Capítulo 3 foram abordadas informações sobre a importância do processo de Projeto e Construção de produto.

Adicionalmente, o Capítulo 4 apresentou o contexto o qual este relato foi aplicado. Inicialmente, foi realizada a definição do processo de Projeto e Construção do Produto Versão 1 utilizando uma linguagem de notação de fácil compreensão. Posteriormente, devido a alguns problemas relatados, foi necessário avaliar o processo de PCP Versão 1, através da metodologia de Pontos Fortes, Pontos Fracos e Oportunidade de Melhoria, onde foi constatada a necessidade de evolução no processo de PCP, pois algumas atividades não estavam sendo executadas ocasionando desvios no processo.

Posteriormente, foi realizada uma análise no resultado, o qual a equipe do SEPG decidiu pela evolução do processo, chamando-o de processo de PCP Versão 2. Como forma de prover melhorias no processo, o SEPG decidiu realizar duas avaliações: a quantitativa e qualitativa, de acordo com o interesse da organização, conforme os resultados apresentados no Capitulo 5.

O resultado apresentado no Capítulo 5, é que, as evoluções que houve no processo de PCP, foram importantes para obter um processo mais maduro e de acordo com as boas práticas do MPS.BR. Desta forma, o resultado final é obtenção de um resultado positivo de melhoria no processo, atendendo assim a necessidade da empresa.

6.2 Análise dos Resultados

Todos os resultados obtidos neste trabalho foram divulgados para os gerentes e os coordenadores. O passo seguinte foi a publicação no site institucional da organização, apresentando os resultados de cada uma das avaliações realizadas, que são descritos a seguir:

- Definição do processo Projeto e Construção do Produto Versão 1 – foi apresentado como realizar a definição de um processo de PCP utilizando a linguagem de notação BPMN.

- Abrangência com um modelo de qualidade – foi apresentado como realizar um estudo de abrangência do processo de PCP com o modelo de qualidade do MPS.BR, com o objetivo de aumentar o grau de maturidade do processo.

- Evolução do processo de PCP Versão 2 – foi apresentada a evolução de forma detalhada, através da notação BPMN do segundo processo de PCP. Esta evolução ocorreu incialmente com o levantamento dos pontos fracos e oportunidades de melhoria listadas no processo de PCP Versão 1.

- Análise do processo através da avaliação quantitativa – foi apresentado detalhadamente como realizar uma avaliação de um processo de PCP utilizando o *framework* GQM, como forma de validar a evolução do processo de PCP.
- Análise do processo através da avaliação qualitativa – foi apresentado detalhadamente como realizar uma avaliação através de entrevista utilizando como método um questionário distribuído entre pessoas que participaram do processo, como forma de verificar se o processo de PCP estava funcionando corretamente.
- Unificação dos Processos: teve como benefício para a Fábrica de Software, a redução do re-trabalho das atividades do processo, gerando menos impacto. O motivo da redução foi que os requisitos só eram liberados após a aprovação do cliente, isso não ocorria no processo de PCP Versão 1;
- Utilização de Camadas na Arquitetura: a utilização de camadas oferece suporte à flexibilidade e portabilidade, o que resulta em uma manutenção menos trabalhosa, considerando-se o tempo estipulado para execução da tarefa e o prazo em que a mesma foi finalizada, segundo dados coletados com os engenheiros de software e arquitetos do projeto;
- Utilização do Protótipo de Interface: a utilização de um protótipo de interface, foi um aspecto positivo, já que os desenvolvedores realizavam o desenvolvimento baseados em uma interface gráfica, já validada pelo cliente e dentro de um padrão visual de interface, conforme descrito na métrica 10.
- Prova de Conceito: a utilização da prova de conceito deu suporte à escolha da tecnologia a ser utilizada. De acordo com a análise dos relatórios, esta prática reduz o grau de incompatibilidade ou rejeição da tecnologia escolhida, em comparação ao processo PCP Versão 1, onde esta tarefa não era realizada. A adoção da prova de conceito ao processo trouxe um ganho de produtividade, pois a prova de conceito diminui a possibilidade da utilização de uma tecnologia que não tenha permitido uma visão positiva sobre o seu potencial em um ambiente corporativo, conforme relatado no ponto forte do processo de PCP Versão 2.
- Padronização no Banco de Dados - a padronização da nomenclatura melhorou o entendimento do banco de dados pela equipe de desenvolvimento. O resultado foi a melhor organização de tabelas, atributos e outros objetos do bancos e a divisão por módulos funcionais, caracterizando o glossário de termos oriundos das regras de

negócio com maior clareza, conforme relatado no ponto forte do processo de PCP Versão 2.

- Aumento da produtividade - o aumento da produtividade pode ser observado, na métrica 1, métrica 2 através da realização de treinamentos de negócio e treinamentos técnicos.
- Redução nos erros de codificação - conforme a métrica 9, os treinamentos de teste de unidade ministrados pela equipe de arquitetura, contribuíram para gerar um código com menos erros de codificação, já que o desenvolvedor testava cada unidade de código desenvolvida;
- A publicação de um artigo em uma revista cientifica, Melo, M. P., Oliveira, S. R. B. **Uma Institucionalização do Processo de Projeto e Construção do Produto em uma Fábrica de Software**. Revista Traços do Centro de Ciências Exatas e Tecnologia da UNAMA (Universidade da Amazônia), ISSN: 1516-0025, v. 12, n. 26 – Belém Pará. 2010

6.3 Trabalhos futuros

Esta seção tem o objetivo de apresentar possíveis trabalhos futuros que podem ser realizados de forma evolutiva ou novas contribuições para a área de Qualidade de Software da Engenharia de Software, assim como no processo de Projeto e Construção de Produto.

- Adaptar o processo de PCP - atender outros processos de desenvolvimento de software. O objetivo é que em cenário similares pode ser utilizado o mesmo processo, realizando algumas adaptações;
- Adaptar este processo para utilização de metodologias ágeis – algumas empresas adotam metodologias ágeis, desta forma, realizar uma adaptação para atender a necessidade desta empresa. A vantagem é de se utilizar as metodologias ágeis realizando algumas adaptações no processo, para que sejam utilizadas as mesmas metodologias que foram utilizadas neste trabalho. Inclusive, adotar a avaliação do processo utilizando o método qualitativo durante uma atividade da metodologia ágil que é retrospectiva;
- Institucionalizar as métricas utilizadas neste trabalho e criar novas métricas de acordo com a necessidade da organização. A importância é medir o nível de

maturidade, a satisfação da equipe e a situação real do andamento do processo de construção do produto;

- Realizar um trabalho para aderência completa ao MPS.BR na área de PCP e ITP (Integração do Produto). As melhorias podem ocorrer após cada evolução do processo, objetivando alcançar níveis mais altos de maturidade.

6.4 Limitações do processo

Durante a realização das avaliações do processo foi possível observar algumas limitações, são elas:

- A impossibilidade de utilizar este processo de Projeto e Construção de Produto para uma outra realidade de fábrica de software, já que o processo foi definido de acordo com a necessidade e o tamanho da empresa, ou seja, para ser implementado em outra organização é necessário realizar ajustes para atender as especificidades daquela organização.
- Adaptar este processo de PCP para utilizar ativos de outros modelos de qualidade como o CMMI ou ISSO/IEC 12207, para realizar uma comparação da abrangência e efetividade do processo de PCP em cada modelo de qualidade.
- Outra limitação encontrada foi no número de participantes das auditorias e dos questionários: 14 (quatorze) pessoas. É recomendado aplicar em um conjunto maior de participantes, como forma de validar mais consistente o processo de PCP proposto.

Referências Bibliográficas

Andrade, J. M. S. **Avaliação de Processos de Software em Ambientes de Desenvolvimento de Software Orientados à Organização**. Dissertação de Mestrado. COPPE/ UFRJ. Rio de Janeiro. 2005.

Bartié, A.. **Garantia da qualidade de software**. 1.ed. Rio de Janeiro: Elsevier. 291p. 2002.

Basili, V. R, G. **Caldiera and H. D. Rombach. Goal Question Metric Paradigm**. In John C. Marciniak, editor, Encyclopedia of Software Engineering, volume 1. John Wiley & Sons, 1994b.

Basili, V. R., Caldiera, G. & Rombach, H. D. **The Experience Factory**. In: Marciniak. 469-476p. 1994.

Booch, G.; Rumbaugh J.; Jacobson I. **UML Guia do Usuário**; Ed. Campus. 2006

Borges, L.S.M., Falbo, R.A. **Gerência de Conhecimento sobre Processos de Software**. VIII Workshop de Qualidade de Software, XV Simpósio Brasileiro de Engenharia de Software, pp. 27-38, Rio de Janeiro, Brasil, Outubro 2007

Brogini, G.G., Sanches, R. **GQS-AE – Uma Abordagem Evolucionista para a Garantia da Qualidade de Software**. 2002.

Chen, P. Gerenciando Banco de Dados - **A Abordagem Entidade-Relacionamento para Projeto Lógico**. Editora MCGraw-Hill, 1990.

Chrissis, M. B., Konrad, M., e SHRUM, S. **CMMI: Guidelines for Process Integration and Product Improvement**, 2nd edition. Bostom: Addison-Wesley Professional. 2002

Coelho, C.: MAPS: **Um Modelo de Adaptação de Processos de Software**, Dissertação de Mestrado, Centro de Informática, Universidade Federal de Pernambuco, Brasil 2003.

Davydov, Mark. **Soa Adventures**. IBM Developer Works. http://www.ibm.com/developerworks/webservices/library/ws-soaadventure2/ index.html. Acessado em Agosto de 2007.

De Marco, T.. **Análise Estruturada e Especificação de Sistemas**. Editora Campus, 1983.

Ebert, C., Dumke, R. **Software Measurement Establish - Extract - Evaluate – Execute**. Springer-Verlag, Berlin, Germany. 2006.

Erl T., **SOA Design Patterns** (The Prentice Hall Service-Oriented Computing Series, Prentice Hall PTR; 1 edition 2009

Falbo, R. A. **Integração de Conhecimento em um Ambiente de Desenvolvimento de Software**, COPPE/UFRJ, Rio de Janeiro, v. Tese de Doutorado. 1998.

Filho, R. C. S.; Rocha, A. R., Travassos, G. H., **Uma Abordagem para Avaliação de Propostas de Melhoria em Processos de Software**. Anais do VI Simpósio Brasileiro de Qualidade de Software, SBQS. Porto de Galinhas, Brasil. 2007.

Gane, C.. **Desenvolvimento Rápido de Sistemas** . Livros Técnicos e Científicos Editora,1988

Guerra E., Travassos G., Gleidson Santos(COPPE/UFRJ), Mafra S. , Barreto A. Rocha A. S. **Melhoria de Processos no Desenvolvimento de Software e Hardware – O caso *Maxtrack***. V Simpósio Brasileiro de Qualidade de Software, p 326. 2006

Humphrey, W., Kitson, D., Kasse, T., **The Stateof Software Engineeringpractice: A preliminaryreport**, In: Proceedingsofthe 11th InternationalConferenceon Software Engineering; pp.277-288, 1989

Jeston, J, Nelis, Johan. **Business Process Management: Practical Guidelines to Successful Implementations**, Butterworth-Heinemann (2nd Edition). 2008.

Koscianski, S. A., Santos M.. **Qualidade de software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento do software**. 2.ed. São Paulo: Novatec, 2007. 395p

Lahoz, C. H. N., Santanna, N. **Os Padrões ISO/IEC 12207 e 15504 e a Modelagem de Processos da Qualidade de Software**. In: III Worshop dos Cursos de Computação Aplicada do INPE, 2003, São José dos Campos. Anais do III WORKCAP, 2003. p. 43-48.

Macedo C. Lima. S. H. C., Natali C. A. **Implantação da melhoria de Processo de Software no Tribunal Superior**. V Simpósio Brasileiro de Qualidade de Software, p 351. 2006

Machado, C. F.. **Definindo Processos do Ciclo de Vida de Software Usando a Norma NBR ISO/IEC 12207 e Suas Ementas 1 e 2**. Lavras: UFLA/FAEPE. 2006.

Magela, R. **Engenharia de Software Aplicada – Fundamentos. 1**. Ed. Rio de Janeiro: Alta Books, 2006.

Magela, R. **Engenharia de Software Aplicada – Princípios. 1**. Ed. Rio de Janeiro: Alta Books, 2006b.

Maia, A., Freitas, A., Nunes, D.: Um Modelo para Auxiliar a Adaptação de Processos de Software. In: IV Congresso Brasileiro de Computação, 2004, Itajaí. Anais do IV Congresso Brasileiro de Computação. Itajaí: Univali p.155 – 160 (2004)

Melo, M. P., Oliveira, S. R. B. **Uma Institucionalização do Processo de Projeto e Construção do Produto em uma Fábrica de Software**. Revista Traços do Centro de Ciências Exatas e Tecnologia da UNAMA (Universidade da Amazônia), ISSN: 1516-0025, v. 12, n. 26 – Belém Pará. 2010

Miers, D., BPMN Modeling and Reference Guide, Future Strategies Inc., Lighthouse Pt, FL, 2008

Molinari, L. **Gerência de Configuração - Técnicas e Práticas no Desenvolvimento do Software**. Florianópolis: Visual Books, v. 85-7502-210-5. 2007.

Moran, T. "The Command Language Grammars: a represetantion for the user interface of interactive computer systems. International Journal of Man- Machine Studies, 15, 3-50 1981.

Münch, J.: Transformation-basedCreationofCustom-tailored Software ProcessModels. FraunhoferInstitute for Experimental Software Engineering 2004.

Norman, D. "Cognitive Artifacts". In Carroll (ed.) Designing Interaction: Psychology ate the Human-Computer Interface. pp.17-38, 1986.

Oliveira, S. R. B. **Processo de Software: Princípios, Ambientes e Mecanismos de Execução**. 2006. 263 f. Exame de Qualificação de Doutorado – Universidade Federal de Pernambuco, Recife. 2006.

Pfleeger, S. L. **Engenharia de Software: Teoria e Prática**, Prentice Hall do Brasil, 2ª Edição, São Paulo, 2004

Pressman, R. S. **Software Engineering A Practitioner Approach**. 6 ed. New York:. McGraw-Hill, 2005
Pressman, Roger S. **Engenharia de software**. 6.ed. São Paulo: McGraw-Hill do Brasil, 2006. 720p

Salviano, C.F. **Melhoria e Avaliação de Processo de Software com o Modelo ISO/IEC 15504-5**:2006, Lavras: UFLA/FAEPE, 2006.

SEI – Software Engineering Institute (2006) **Capability Maturity Model Integration for Development – CMMI-Dev**. Versão 1.2.

SEI – Software Engineering Institute. **CMMI for Development**, Version 1.2. Pittsburgh: Carnegie Mellon University, 2006. Disponível em: http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tr008.pdf. Acesso em abril de 2008.

Silva, E. L.; Menezes, E. M. (2001) “Metodologia da Pesquisa e Elaboração da Dissertação”. 3. ed. rev. Atual – Laboratório de Ensino a Distância da UFSC. Florianópolis, Brasil. 2001.

Silva R. C., Rocha A. R., Travassos G. H. **Uma Abordagem para Avaliação de Propostas de Melhoria em Processos de Software.**VI Simpósio Brasileiro de Qualidade de Software (SBQS 2007). 2007.

Softex – Associação para Promoção da Excelência do Software Brasileiro (2011). **MPS.BR - Guia Geral.** Julho 2011.

Softex – Associação para Promoção da Excelência do Software Brasileiro. **MPS.BR - Guia de Implementação – Parte 4**Julho 2011b.

Sommerville, I. (2007). **Engenharia de Software.** Pearson Addison Wesley, 8a edição, 2007.

Souza, C. S.; Leite, J. C.; Prates, R.O. & Barbosa, S.D.J. - **Projeto de Interfaces de Usuário: Perspectivas Cognitiva e Semiótica**, Anais da Jornada de Atualização em Informática, XIX Congresso da Sociedade Brasileira de Computação, Rio de Janeiro, julho de 1999

Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P., E Murphy, R. **An exploratory study of why organizations do not adopt CMMI**. The Journal of Systems and Software. 2007.

Tan, Joseph;, WEN, H.Joseph; AWAD, Neveen. **Health care and services delivery systems as complex adaptative systems**. Communications of the ACM. Volume 48, Número 5 (2005), Páginas 36-44.

The International Organization for Standardization/ International ElectrotechnicalComission. **ISO/IEC 15504-5: Information Technology - Process Assessment**. Geneve: ISO, 2006.

The International Organization for Standardization/ The International ElectrotechnicalComission**ISO/IEC 12207 Systems and software engineering– Software life cycle processes**. Geneve. 2008.

Tonsig, S. L. **Engenharia de Software - Análise e Projeto de Sistemas**. 2. Ed. Rio de Janeiro: Ciência Moderna, 2008.

Travassos, G.H.; Gurov, D.; Amaral, E.A.G.; **Introdução à Engenharia de Software Experimental**; Relatório Técnico RT-ES-590/02; COPPE/UFRJ; 2002

Weber, K.C.; Nascimento, C.J. Do; Marinho, D. Da S. (2006). Programa Brasileiro da Qualidade e Produtividade em Software: **Treze Anos Acompanhando e Disseminando a Cultura da Qualidade**, In: Revista PRÓQUALITI - Qualidade na Produção de Software, vol. 2, no. 1, Lavras: UFLA, 2006

Xavier, C.M.S., Portilho, C. **Projetando com Qualidade a Tecnologia de Sistemas de Informação**. Livros Técnicos e Científicos Editora, 1995

Yoon, I. et al. : **Tailoring and Verifying Software Process**. Proceedings of the Eighth Asia-Pacific Software Engineering Conference (APSEC'01). December, 2001. Page 202. Macao, China (2001)

Apêndice A - Métricas Definidas

Para que seja necessário avaliar de forma quantitativa o Processo de Projeto e Construção de Produto, fez-se necessário a utilização de métricas de acordo com o GQM.

A.1. Definição detalhada das métricas

De acordo com o Objetivo 1"Definir a produtividade da equipe de desenvolvimento utilizando as práticas definidas no processo" foram definidas 5 métricas como parâmetro para saber se o objetivo foi atendido. A primeira métrica tem o objetivo de Quantificar a velocidade em que a equipe de desenvolvimento leva para finalizar uma determinada atividade, conforme mostrado no Quadro 6-1.

Quadro 6-1–M1: Quantificar a velocidade da equipe de desenvolvimento.

Métrica	**M1: Quantificar a velocidade da equipe de desenvolvimento.**
Descrição	Quantificar quanto tempo de trabalho leva o Engenheiro de Software para realizar as tarefas de sua responsabilidade. A quantificação será feita através da medição do tempo de trabalho de engenheiro realizando o desenvolvimento de requisitos com o mesmo grau de complexidade.
Composição da métrica	QVED = Quantificar a Velocidade da Equipe de Desenvolvimento. TI = Tempo Inicial, quando a atividade começa. TF = Tempo Final, quando a atividade termina.
Equação cálculo	QVED = TF – TI
Unidade de medida	Hora
Valor base	1.00
Procedimento coleta	A coleta é realizada no começo e no término de cada atividade, durante o período citado. Será aplicado a fórmula, e assim extraindo os resultados para uma análise.
Forma apresentação	Gráfico de coluna
Indicador	• 1 a 3 Horas – Rápido • 4 a 7 Horas – Normal • Acima de 8 Horas - Lento
Procedimento análise	O resultado da coleta é atribuído à fórmula. QVED = TF – TI, onde o tempo final de execução da atividade é substituído no TF e o tempo inicial no TI. O valor do QVED será o tempo em que a tarefa foi finalizada.

A segunda métrica tem o objetivo de quantificar qual o tempo que a equipe de desenvolvimento gasta para executar a codificação de um requisito, como mostrado no Quadro 6-2.

Quadro 6-2– M2: Diferença entre o tempo estimado e o real definido pela equipe.

Métrica	**M2: Diferença entre o tempo estimado e o real definido pela equipe.**
Descrição	Definir a diferença entre o tempo estimado para a realização de uma atividade com o tempo real da execução da mesma.
Composição da métrica	DTER – Diferença do Tempo Estimado e o Real TE – Tempo Estimado para execução de uma atividade QVED – Fórmula usada na métrica 1 para saber o tempo real que a tarefa foi desenvolvida
Equação cálculo	DTER = QVED – TE
Unidade de medida	Hora
Valor base	1.00
Procedimento coleta	Existem dois momentos para realização da coleta. O primeiro momento é quando o Engenheiro de Software estima o tempo para o desenvolvimento da tarefa. O segundo momento é a coleta de quanto tempo a atividade foi realizada. A análise será feita em cima dos dados coletados, mostrando a diferença entre o tempo estimado para o tempo real.
Forma apresentação	Gráfico de coluna
Indicador	• Acima de 2 horas – Tarefa Superestimada • Abaixo de 2 horas – Tarefa Subestimada • Entre 2 horas (para mais ou para menos) - Normal
Procedimento análise	Como análise, os valores são substituídos na fórmula DTER = QVED – TE, onde o QVED é o valor real em que a tarefa foi finalizada e o TE é o tempo estimado para execução desta atividade. Analisando o resultado, caso o valor seja positivo, chega-se a conclusão que a tarefa foi subestimada pelo desenvolvedor. O resultado sendo positivo, a tarefa foi finalizada de acordo com o tempo previsto. Caso o valor seja negativo é que a tarefa foi superestimada pelo desenvolvedor.
Resultado	Infere-se que se a tarefa foi realizada acima de 2 horas do tempo previsto, esta tarefa pode ter sido subestimada. Caso a tarefa tenha sido concluída com 2 horas a menos a tarefa pode ter sido superestimada, desta forma,. Caso estes treinamentos não sejam realizados, possivelmente poderão haver erros no planejamento do cronograma. Caso a atividade tenho sido executado entre o tempo de 2 horas pra mais ou 2 horas para menos, está atividade foi executada no tempo normal. Para este caso, recomenda-se manter o processo de estimativa que está sendo seguido.

A terceira métrica tem o objetivo de verificar qual a disponibilidade da equipe para uma determinada atividade, conforme apresentado no Quadro 6-3.

Quadro 6-3–M3: Definir a disponibilidade da equipe para a atividade.

Métrica	**M3: Definir a disponibilidade da equipe para a atividade**
Descrição	Definir quando a equipe ou parte da equipe está disponível ou ociosa para realizar uma nova atividade.
Composição da métrica	DDEA = Definir a disponibilidade da equipe para a atividade ROQ = Resultado Obtido através do questionário. O questionário possui dentre suas perguntas uma lista de atividades, como por exemplo, ir ao médico e outras atividades extratrabalho.
Equação cálculo	DDEA = 40h – ROQ
Unidade de medida	Homem/hora
Valor base	1.00
Procedimento coleta	A coleta será em cima do questionário, sendo possível definir de forma aproximada o tempo que a equipe fica disponível.
Forma apresentação	Gráfico de coluna
Indicador	Acima de 34 horas trabalhadas – Normal Abaixo de 34 horas trabalhadas – Atenção
Procedimento análise	O cálculo será realizado através do resultado obtido através de um questionário semanal, onde o desenvolvedor indica, qual será o tempo que o mesmo não estará exercendo a atividade de codificação e assuntos relacionados ao trabalho. Como exemplo, o resultado será medido da seguinte forma, se o desenvolvedor diz que tem uma consulta no médico e o tempo estimado para ficar fora do ambiente de trabalho é de 4 horas. Como análise, os valores são substituídos na fórmula DDEA = 40h – 4h, o valor será de 36 horas para realização das atividades durante a semana. O resultado é importante para alocar uma atividade onde o mesmo consiga cumprir de acordo com o tempo disponível. Por outro lado se o tempo está abaixo 34 horas (tempo definido pela gerência) é importante que os gerentes conversem com seu subordinado informando que as atividades extratrabalho estão comprometendo o andamento do projeto.
Resultado	Infere-se que se o colaborador está a disposição para a realização das atividades igual ou acima de 34 horas, então significa que este colaborador está em situação normal. Caso o colaborador esteja disponível em um tempo abaixo de 34 horas semanais, , caso contrário,

A quarta métrica tem o objetivo de avaliar a complexidade das atividades, como mostrado no Quadro 6-4.

Quadro 6-4–M4: Avaliar a complexidade das atividades.

Métrica	**M4: Avaliar a complexidade das atividades.**
Descrição	Avaliar o grau de complexidade das atividades executadas pela equipe.
Composição da métrica	ACA = Avaliar a Complexidade das Atividades CR = Complexidade do Requisito, de acordo com o questionário.
Equação cálculo	ACA = (CR * 100)/T
Unidade de medida	Não se Aplica
Valor base	1%
Procedimento coleta	A coleta será realizada através de um questionário. Este questionário é composto por questões do tipo: Qual a quantidade de atributos o requisito possui, ou ainda qual a quantidade de dependência deste requisito com outros requisitos. Neste questionário cada resposta terá um peso. A soma total de pontos será possível determinar a complexidade do requisito.
Forma apresentação	Gráfico de coluna
Indicador	
Procedimento análise	Para avaliar o grau de complexidade do requisito, é disponibilizado um questionário onde o engenheiro de software responde a alguns critérios. Por exemplo, um requisito possui dependências com outros requisitos; qual a quantidade de fluxos alternativos que aquele requisito possui; qual a complexidade no que diz respeito a quantidade de atributos; qual a complexidade relacionada a fórmulas; entre outras questões. Como análise esta complexidade ajuda a entender se o tempo alocado para aquela atividade condiz com o que foi estimado.

A quinta métrica tem o objetivo de verificar qual o grau de conhecimento da equipe em uma determinada atividade, como apresentado no Quadro 6-5.

Quadro 6-5–M5: Grau de conhecimento da equipe nas regras que envolvem a atividade.

Métrica	**M5: Grau de conhecimento da equipe nas regras que envolvem a atividade.**
Descrição	Medir o grau de conhecimento do Engenheiro de Software, para uma determinada atividade.
Composição da métrica	GCERA = Grau de Conhecimento da Equipe nas Regras que envolvem a atividade ROQ = Resultados Obtidos do Questionário. T= Total de respostas do questionário.
Equação cálculo	GCERA = (ROQ * 100)/T
Unidade de medida	Não se Aplica
Valor base	1%
Procedimento coleta	A coleta será realizada através de um questionário, contendo

	informações de negócio do projeto. Dessa forma poderá ser mensurado o quanto aquele Engenheiro de Software conhece as regras do produto que está sendo desenvolvido.
Forma apresentação	Uma tabela apresentando o valor total das respostas obtidas no questionário.
Procedimento análise	Para se obter um grau de conhecimento da equipe no que se diz a respeito às regras que envolvem a atividade, um questionário é respondido por cada membro da equipe. Após a finalização do questionário, os valores coletados são substituídos na fórmula GCERA = (ROQ * 100)/T. Segundo a definição realizada pela gerência, caso o valor do GCERA seja inferior a 60% (valor definido com base em um projeto piloto para alinhamento e nivelamento do perfil da equipe e complexidade do escopo do projeto a ser desenvolvido), é importante que treinamentos sejam realizados. Caso o valor seja igual ou acima então está dentro do esperado pela gerência.

A sexta métrica tem o objetivo de quantificar o número de defeitos de acordo com sua complexidade, como mostrado no Quadro 6-6.

Quadro 6-6–M6: Quantificar o número de defeitos de acordo com sua complexidade?

Métrica	**M6: Quantificar o número de defeitos de acordo com sua complexidade?**
Descrição	Agrupar a quantidade de defeitos encontrados de acordo com sua complexidade.
Composição da métrica	QDEAC = Quantidade de defeitos Encontrados de Acordo com sua Complexidade. COMP = Complexidade dos defeitos QTD = Quantidade de defeitos encontrados
Equação cálculo	QDEAC = $\sum_{QTD}^{COMP}$
Unidade de medida	Não se Aplica
Valor base	1.00
Procedimento coleta	A coleta será realizada através da ferramenta de relato de defeitos. Após a coleta os dados serão agrupados.
Forma apresentação	Gráfico de coluna
Procedimento análise	Com o resultado da coleta, é possível verificar como está a qualidade do produto. Os defeitos são coletados e substituídos na fórmula, onde é mostrado o percentual de defeito por nível de complexidade. Conforme definição no contrato, o acordo de nível de serviço (*servisse level agreement* – SLA) do projeto é definido da seguinte maneira: para aceitação do produto por parte do cliente, pode-se considerar pelo menos até 1 defeito importante, nenhum defeito crítico e qualquer quantidade de defeitos triviais. Como análise do resultado do

	QBEAC é possível definir se o produto será aceito pelo cliente ou não. Caso o valor não satisfaça o nível do SLA, é importante realizar as correções necessárias para aceitação do produto gerado antes de tornar disponível para homologação.

A sétima métrica tem o objetivo de analisar o grau de aderência das atividades da equipe em relação aos critérios de auditoria, como apresentado no Quadro 6-7.

Quadro 6-7–M7: Analisar o grau de aderência das atividades da equipe em relação aos critérios de auditoria?

Métrica	**M7: Analisar o grau de aderência das atividades da equipe em relação aos critérios de auditoria?**
Descrição	Analisar o grau de aderência das atividades da equipe de acordo com os critérios que foram definidos para verificar a qualidade do que está sendo gerado.
Procedimento coleta	Através de um questionário que será respondido pelo engenheiro de software é possível definir o quanto ele está aderente ao processo.
Composição da métrica	GAAECA = Grau de Aderência das Atividades da Equipe em relação aos Critérios de Auditoria. ROE = Resultados obtidos através das entrevistas realizadas com os Engenheiros de Software. T = Total de respostas do questionário
Equação cálculo	GAAECA = (ROE * 100)/T
Unidade de medida	Não se Aplica
Valor base	1%
Procedimento coleta	A coleta será feita através de um questionário, onde o engenheiro de software responde a questões do tipo: Você realiza a atividade de verificação do que foi implementado com o padrão de interface definido. Como resposta é possível esperar: Contempla, ou seja, é realizado uma verificação do que foi implementado com o padrão de interface; Contempla em parte, onde é possível inferir que o engenheiro de software só realiza parte do que está sendo solicitado; e Não contempla, onde não é realizado nada do que foi solicitado.
Forma apresentação	Gráfico de coluna.
Procedimento análise	Após a extração do resultado obtido através da entrevista, os dados serão agrupados em uma tabela, para então realizar a extração de um percentual que indique o quanto ele está aderente a uma determinada atividade do processo. Como definição existe três itens a serem avaliados, que são: Contempla se o total foi acima de 75% das perguntas respondidas; Contempla em parte, o valor tem que ser de 50% até 74%; e se o valor foi abaixo de 50% então não contempla. Para os dois últimos casos devem ser realizados treinamentos para que os artefatos produzidos estejam em adequação ao padrão de qualidade solicitado no processo.

A oitava métrica tem o objetivo de quantificar o percentual de aderência do produto gerado em relação ao critério de verificação dos requisitos, como mostrado noQuadro 6-8.

Quadro 6-8–M8: Quantificar o percentual de aderência do produto gerado em relação ao critério de verificação dos requisitos.

Métrica	**M8: Quantificar o percentual de aderência do produto gerado em relação ao critério de verificação dos requisitos.**
Descrição	Quantificar o percentual de aderência do produto gerado em relação ao critério que foram definidos para a verificação dos requisitos.
Composição da métrica	PAPG = Percentual de Aderência do Produto Gerado ROA = Resultado Obtido através de Auditoria T = Total de respostas do questionário
Equação cálculo	PAPG = (ROA * 100)/T
Unidade de medida	Não se Aplica
Valor base	1%
Procedimento coleta	Será realizada uma verificação no requisito e no código-fonte para identificar qual o grau de aderência. Esse grau de aderência será obtido através de pontuações, essas pontuações servem para verificar o grau de aderência entre o produto gerado e os requisitos. O responsável realiza a verificação do fluxo principal e atribui uma pontuação. Verifica a quantidade de fluxos alternativos e atribui uma pontuação.
Forma apresentação	A apresentação será em tabela, onde será mostrado o percentual de aderência entre o requisito e o código-fonte
Procedimento análise	É realizado uma auditoria no código-fonte e no documento de requisitos para então verificar o quão aderente está o documento em relação ao código-fonte. As faixas de valores descritas foi definida pela equipe de qualidade do projeto juntamente com a gerência do projeto. O motivo desta verificação é que a atividade de implementação é fortemente dependente das atividades realizadas na fase de levantamento de requisitos. Qualquer desconformidade pode gerar um produto que não atenda a real necessidade do cliente. As foram métricas definidas foram: se o total de perguntas respondidas de maneira correta for de 20%, é considerado Crítico, então o código precisa ser revisado para que se obtenha uma melhor qualidade do que está sendo gerado; se o valor for entre 21% até 74% então é considerável como aceitável, mas também deve ser realizado uma revisão no que foi gerado; o valor acima de 75% então o produto está de acordo com o padrão definido pela gerência ou a área de qualidade do produto.

A nona métrica tem o objetivo de quantificar o percentual de cobertura dos testes de unidade em relação as transações (fluxo principal e alternativo), como apresentado no Quadro 6-9.

Quadro 6-9–M9: Quantificar o percentual de cobertura dos testes de unidade em relação às transações (fluxo principal e alternativo).

Métrica	**M9: Quantificar o percentual de cobertura dos testes de unidade em relação às transações (fluxo principal e alternativo)?**
Descrição	Verificar o percentual de cobertura dos testes de unidades de acordo com o requisito, levando-se em conta o fluxo principal e os fluxos alternativos.
Composição da métrica	PCTU = Percentual de Cobertura de Teste de Unidade ROA = Resultado Obtido através de Auditoria T = Total de respostas do questionário
Equação cálculo	PCTU = (ROA * 100)/T
Unidade de medida	Não se Aplica
Valor base	1%
Procedimento coleta	Um especialista em desenvolvimento irá realizar um auditoria nos testes de unidades, verificando a aderência de acordo com o requisito. Essa aderência será de acordo com a cobertura de testes do fluxo principal e do fluxo alternativo.
Forma apresentação	Gráfico de coluna
Procedimento análise	Após a coleta de resultados realizada pelo auditor, os resultados levados para análise servem para verificar se os testes de unidades estão sendo eficientes na cobertura dos testes de acordo com o fluxo principal e os fluxos alternativos documentados no requisito. Foram estipuladas pela gerência três faixas de valores: de 0 a 50% é classificado como abaixo do esperado e precisa ser revisado; de 51 até 75% é aceitável, só que com restrições; e a última faixa é de 76% até 100% que é aceitável. Esses valores foram definidos em um projeto piloto, onde: inicialmente foi aplicado um *mentoring*, programação em par e disseminação do conhecimento, para capacitação da equipe sobre a técnica de teste de unidade; posteriormente, a pessoa responsável pela capacitação realizou uma avaliação dos resultados obtidos a partir da aplicação em um projeto piloto, com base em critérios que focam no padrão de implementação constante no processo de software; finalmente, de posse dos resultados foi realizado um *workshop* com os envolvidos e a gerência para alinhamento das metas com base nos resultados obtidos.

A décima métrica tem o objetivo de quantificar qual o percentual de aderência dos itens de verificação do padrão de interface em relação ao produto gerado, como apresentado no Quadro 6-10.

Quadro 6-10–M10: Quantificar o percentual de aderência dos itens de verificação do padrão de interface em relação ao produto gerado.

Métrica	**M10: Quantificar o percentual de aderência dos itens de verificação do padrão de interface em relação ao produto gerado?**
Descrição	Verificar o percentual de aderência dos itens do padrão de interface junto ao produto gerado
Composição da métrica	PAII = Percentual de Aderência aos Itens de Interface ROA = Resultado Obtido através de Auditoria T = Total de respostas do questionário
Equação cálculo	PAII = (ROA * 100)/ T
Unidade de medida	Não se Aplica
Valor base	1%
Procedimento coleta	Um especialista em desenvolvimento, irá realizar um auditoria entre o documento de padrão de interface e o produto gerado, verificando a aderência entre ambos. A aderência será realizada de acordo com o documento de padrão de interface.
Forma apresentação	Gráfico de coluna
Procedimento análise	Após a coleta de resultados realizada pelo auditor, os resultados levados para análise. O objetivo é verificar o quão aderente o produto que foi gerado está de acordo com o padrão de interface, a faixa de valores a seguir foi definido entre uma reunião entre a equipe de interface e a equipe de arquitetura do projeto. Foram estipuladas pela gerência três faixas de valores: de 0 a 50% é classificado com abaixo do esperado e precisa realizar uma reunião entre a equipe de desenvolvimento e de interface para avaliar o que pode ser contemplado, para aumentar a aderência do produto com o padrão de interface; de 51 até 75% é aceitável, só que com restrições; e a última faixa é de 76% até 100% que é normal.

A décima primeira métrica tem o objetivo de verificar qual a diferença entre o tempo estimado pela gerência e o tempo estimado pela equipe para realização das atividades, como apresentado no Quadro 6-11.

Quadro 6-11–M11: Qual a diferença entre o tempo estimado pela Gerência e o tempo estimado pela equipe para realização das atividades.

Métrica	**M11: Qual a diferença entre o tempo estimado pela Gerência e o tempo estimado pela equipe para realização das atividades?**
Descrição	Verificar a diferença entre o tempo estimado pela Gerência e o tempo estimado pela equipe que irá realizar a atividade.
Composição da métrica	DTETR = Diferença entre o Tempo Estimado pela gerência e o Tempo Real estipulado pela equipe TE = Tempo Estimado pela equipe (Coletado na Métrica 1) TEG = Tempo Estimado pela Gerência
Equação cálculo	DTETR = TEG – TE
Unidade de medida	Hora
Valor base	1.00
Procedimento coleta	A coleta será realizada com os gerentes e a equipe.
Forma apresentação	Gráfico de coluna
Procedimento análise	Após a coleta de resultados será substituído na fórmula: DTETR = TEG – TE. A margem dos valores aceitáveis é de 2 horas para cima ou abaixo. Essa margem de valores foi definido através de reuniões gerenciais, subsidiadas a partir dos valores obtidos em estimativas dos fatores ambientais definidos em pontos de caso de uso, técnica utilizada para definir o esforço da implementação. O valor sendo positivo e acima da margem definida, considera-se que possivelmente os gerentes estão superestimando as tarefas ou a equipe está subestimando o esforço para realizar a atividade. Caso o valor seja negativo e abaixo da margem, pode-se considerar possivelmente que os gerentes estão subestimando as tarefas ou a equipe está superestimando o esforço para a realização das atividades. Caso o valor seja igual a 0 ou esteja na margem definida, pode-se inferir que a diferença encontra-se dentro do padrão aceitável. Quando os valores não estiverem a dentro da margem, é interessante que seja realizada uma reunião entre a equipe de desenvolvimento e a gerência para realizar um melhor alinhamento das atividades.

Apêndice B - Resultado da Coleta das Métricas

Este apêndice apresenta a coleta do resultados referente as métricas da avaliação quantitativa, que é descrito a seguir.

Quadro 6-12– Período em que as métricas foram coletadas.

Períodos			
	Início	**Fim**	**Quantidade de dias**
Período 1	06/10/2010	10/11/2010	35
Período 2	12/11/2010	21/12/2010	39
Período 3	05/01/2011	12/02/2011	38
Total			**112**

Tabela 6-1 – Resultado da M1: Quantificar a velocidade da equipe de desenvolvimento.

	Período 1	**Período 2**	**Período 3**
Rápido	45%	40%	19%
Normal	43%	53%	77%
Lento	12%	7%	4%
Total	100%	100%	100%

Tabela 6-2 – Resultado da M2: Diferença entre o tempo estimado e o real definido pela equipe.

	Período 1	**Período 2**	**Período 3**
Acima de 2 horas	63%	41%	21%
Abaixo de 2 horas	22%	35%	34%
Entre o intervalo de 2 hora positivas e negativas	15%	24%	45%
Total	100%	100%	100%

Tabela 6-3 – Resultado da M3: Definir a disponibilidade da equipe para a atividade.

	Período 1	**Período 2**	**Período 3**
Igual ou acima de 34 horas semanais	18%	29%	33%
Abaixo de 34 horas semanais	82%	71%	67%
Total	100%	100%	100%

Tabela 6-4 – Resultado da M4: Avaliar a complexidade das atividades.

	Período 1	**Período 2**	**Período 3**
Subestimada	81%	54%	35%
Superestimada	12%	22%	14%
Normal	7%	24%	51%
Total	100%	100%	100%

Tabela 6-5 – Resultado da M5: Grau de conhecimento da equipe nas regras que envolvem a atividade

	Período 1	**Período 2**	**Período 3**
Inferior a 60%	55%	39%	28%
Igual ou acima de 60%	45%	61%	72%
Total	100%	100%	100%

Tabela 6-6 – Resultado da M6: Quantificar o número de defeitos de acordo com sua complexidade?

	Período 1	**Período 2**	**Período 3**
Contempla	37%	79%	82%
Contempla em parte	48%	16%	15%
Não Contempla	15%	5%	3%
Total	100%	100%	100%

Tabela 6-7 – Resultado da M7: Analisar o grau de aderência das atividades da equipe em relação aos critérios de auditoria?

	Período 1	**Período 2**	**Período 3**
Contempla	37%	79%	82%
Contempla em parte	48%	16%	15%
Não Contempla	15%	5%	3%
Total	100%	100%	100%

Tabela 6-8 – Resultado da M8: Quantificar o percentual de aderência do produto gerado em relação ao critério de verificação dos requisitos.

	Período 1	**Período 2**	**Período 3**
Normal	46%	74%	92%
Aceitável	31%	17%	6%
Crítico	23%	9%	2%
Total	100%	100%	100%

Tabela 6-9 – Resultado da M9: Quantificar o percentual de cobertura dos testes de unidade em relação às transações (fluxo principal e alternativo)?

	Período 1	**Período 2**	**Período 3**
Normal	34%	82%	86%
Aceitável	27%	15%	11%
Crítico	39%	3%	3%
Total	100%	100%	100%

Tabela 6-10 – Resultado da M10: Quantificar o percentual de aderência dos itens de verificação do padrão de interface em relação ao produto gerado?

	Período 1	**Período 2**	**Período 3**
Normal	39%	71%	92%
Aceitável	27%	21%	4%
Abaixo do esperado	34%	8%	4%
Total	100%	100%	100%

Tabela 6-11 – Resultado da M11: Qual a diferença entre o tempo estimado pela Gerência e o tempo estimado pela equipe para realização das atividades?

	Período 1	**Período 2**	**Período 3**
Superestimada	62%	36%	16%
Normal	23%	57%	81%
Subestimada	15%	7%	3%
Total	100%	100%	100%

Apêndice C - Questionário(*Survey*)

Abaixo é apresentado o questionário qualitativo aplicado na fábrica de software.

Questionário

Introdução

Com a evolução do processo de Projeto de Construção de Produtos (PCP) de software, assim como com a evolução do processo de desenvolvimento de software da fábrica, percebeu-se: um aumento de produtividade da equipe alocada para o projeto; aumento de qualidade do produto; e principalmente aumento da satisfação do cliente. Os motivos da satisfação devem-se: o cliente passou a acompanhar o andamento do projeto mais de perto; a transparência no desenvolvimento do PCP, já que toda documentação é disponibilizada para o mesmo; assim como a participação da equipe técnica do cliente na tomada de decisão das soluções do projeto; e a realização contínua de treinamentos ao cliente nos frameworks e tecnologias utilizadas no PCP, por parte do fornecedor. Com o desenvolvimento interativo, a partir dos módulos funcionais, o cliente consegue homologar o sistema com maior detalhe, garantindo maior precisão no atendimento das suas expectativas. Porém, esse processo está passando por constantes atualizações de forma que possa gerar um produto com melhor qualidade.

Objetivo

O processo de PCP necessita de algumas alterações para se tornar aderente às recomendações constantes em um programa de melhoria da qualidade organizacional. Um dos focos é o modelo MPS.BR[5]. Estas alterações ainda estão sendo diagnosticadas e em desenvolvimento para serem institucionalizadas em forma de procedimentos a serem incorporados ao processo.

O objetivo deste trabalho retrata a consolidação de um relato de experiência que descreve a evolução de um processo de PCP, o qual se encontra em constante melhoria para compor os resultados parciais de estudos provenientes da aplicação de um trabalho de melhoria do processo organizacional no contexto de uma dissertação de mestrado do PPGCC/UFPA – Programa de Pós-Graduação em Ciência da Computação. Este resultado está alinhado com os interesses organizacionais na aplicação de uma avaliação MPS.BR a partir das boas práticas constantes no cenário do processo de PCP deste modelo.

Instruções para responder as perguntas

Este questionário é dividido em duas seções. A primeira A.1 é relacionada ao Perfil do participante. Essas informações são úteis para saber qual a formação do participante no contexto de desenvolvimento de software.

A segunda, seção A.2, trata exclusivamente dos questionamentos referentes a parte do processo de software aplicado em uma Fábrica de Software, com foco nos ativos relacionados ao processo de PCP. Para responder as questões é necessário um entendimento do processo de software em avaliação, descrito no **Anexo 1**.

---

[5]Mps.Br - Melhoria de Processos do Software Brasileiro

Exemplo de Preenchimento:

Errado	
Instituição de origem?	(●) Pública ( ) Particular
Correto	
Instituição de origem?	(X) Pública ( ) Particular

A.1 Questionário do Perfil do Participante

Formação	
Tipo de curso?	( ) Engenharia ( ) Informática / Ciência da Computação ( ) Matemática ( ) Outro(s):_____
Grau de escolaridade?	( ) Ensino Médio Técnico ( ) Graduação ( ) Pós Graduação ( ) MBA
Experiência	
Como você classificaria seu entendimento sobre o desenvolvimento de produto de software?	( ) Excelente ( ) Alto ( ) Bom ( ) Médio ( ) Baixo ( ) Nenhum
Experiência ou a atividades (cargo) que mais já exerceu, ou exerce, na área da computação?	( ) Analista de Sistemas ( ) Codificação ( ) Gerente de Processo ( ) Gerente de Projeto ( ) Projetista de Processo ( ) Projetista de Sistemas ( ) SQA ( ) Teste ( ) Usuário
Já participou do desenvolvimento de que tipo de sistemas de computação?	( ) Grande Porte ( ) Médio Porte ( ) Pequeno Porte
Em qual área da Engenharia de Software você mais atuou?	( ) Especificação de Requisitos ( ) Projeto ( ) Codificação ( ) Teste ( ) Manutenção ( ) Garantia / Controle da Qualidade
Como você classificaria seu entendimento em desenvolvimento de software?	( ) Excelente ( ) Alto ( ) Bom ( ) Médio ( ) Baixo

	( ) Nenhum
Já participou de quantos processos de software?	( ) Nenhum ( ) Entre 1 e 2 ( ) Entre 3 e 7 ( ) Acima de 7
Quantos processos de software já definiu?	( ) Nenhum ( ) Entre 1 e 2 ( ) Entre 3 e 7 ( ) Acima de 7
Como você classificaria seu entendimento sobre qualidade de software?	( ) Excelente ( ) Alto ( ) Bom ( ) Médio ( ) Baixo ( ) Nenhum
Já usou quantos modelos/normas da qualidade?	( ) Nenhum ( ) Entre 1 e 2 ( ) Entre 3 e 7 ( ) Acima de 7

A.2 Questionário de Definição do Processo

Nesta segunda etapa você irá avaliar o processo mostrado no Anexo 1. O objetivo deste questionário é validar a utilização deste processo no cenário de aplicação de uma fábrica de software.

Para realizar o preenchimento do questionário, selecione uma das colunas de acordo com as seguintes características:

- Presença: Se uma determinada atividade está presente no Processo de Desenvolvimento de Software apresentado, no contexto do processo de PCP.
- Utilidade: Se uma determinada atividade tem utilidade no processo de Desenvolvimento de Software, no contexto do processo de PCP.
- Adequação: Se uma determinada atividade necessita de alguma adequação para o processo de Desenvolvimento de Software, no contexto do processo de PCP.

Os critérios usados para a avaliação das características listadas são:

Presença	Utilidade	Adequação do nível de detalhamento
1. Não é oferecido pelo processo e não gostaria de tivesse disponível.	1. Não é útil.	1. O detalhamento deve ser aumentado.
2. Não oferecido, mas gostaria que tivesse disponível.	2. Provavelmente é útil, mas ainda não apliquei.	2. O detalhamento não precisa ser modificado.
3. Oferecido, parcialmente.	3. É útil e já apliquei em diferentes implementações.	3. O detalhamento deve ser diminuído.
4. Oferecido.		

Informações para o Preenchimento
Errado

13	Desenvolvimento	Desenvolver Requisito	1	●	3	4	1	2	●	4	1	2	●	4

Correto

13	Desenvolvimento	Desenvolver Requisito	1	**X**	3	4	1	2	**X**	4	**X**	2	3	4

Atividades do Processo

Nº	Equipe	Atividade	Presença				Utilidade			Adequação		
1	Requisitos	Especificar Requisitos	1	2	3	4	1	2	3	1	2	3
2		Validar Requisitos	1	2	3	4	1	2	3	1	2	3
3		Liberar TAG de Requisitos	1	2	3	4	1	2	3	1	2	3
4		Corrigir Ajustes no Requisito	1	2	3	4	1	2	3	1	2	3
5	Testes	Projetar Testes	1	2	3	4	1	2	3	1	2	3
6		Registrar Ajustes	1	2	3	4	1	2	3	1	2	3
7		Executar Testes	1	2	3	4	1	2	3	1	2	3
8		Registrar Ajustes	1	2	3	4	1	2	3	1	2	3
9	Design	Especificar Diagrama Navegacional	1	2	3	4	1	2	3	1	2	3
10		Especificar Protótipo	1	2	3	4	1	2	3	1	2	3
11		Referenciar o Diagrama de Requisitos	1	2	3	4	1	2	3	1	2	3
12		Atualizar Protótipo para Desenvolvimento	1	2	3	4	1	2	3	1	2	3
13	Desenvolvimento	Desenvolver Requisito	1	2	3	4	1	2	3	1	2	3
14		Executar Testes de Unidade	1	2	3	4	1	2	3	1	2	3
15		Gerar Build	1	2	3	4	1	2	3	1	2	3
16		Corrigir Ajustes	1	2	3	4	1	2	3	1	2	3
17	Arquitetura	Projetar Solução Técnica	1	2	3	4	1	2	3	1	2	3
18		Refinar Solução Técnica	1	2	3	4	1	2	3	1	2	3
19		Definir Arquitetura	1	2	3	4	1	2	3	1	2	3
20		Implementar Interface de Integração	1	2	3	4	1	2	3	1	2	3
21		Realizar um *Mentoring* com a Equipe	1	2	3	4	1	2	3	1	2	3
22	Banco de Dados	Projetar Banco de Dados	1	2	3	4	1	2	3	1	2	3
23		Implementar Banco de Dados	1	2	3	4	1	2	3	1	2	3

Para as atividades, que por um acaso não tenham sido encontrados na listagem acima, descreva na tabela a seguir todos estas atividades, e avalie as colunas correspondentes segundo as mesmas características e critérios anteriormente utilizadas como parâmetro quanto ao uso das atividades constantes no processo de Desenvolvimento de Software, no contexto do processo de PCP, listados por você no questionário:

Nº	Equipe	Atividade	Presença				Utilidade			Adequação		
1			1	2	3	4	1	2	3	1	2	3
2			1	2	3	4	1	2	3	1	2	3
3			1	2	3	4	1	2	3	1	2	3
4			1	2	3	4	1	2	3	1	2	3
5			1	2	3	4	1	2	3	1	2	3

Anexo 1 do questionário

A Figura 1 exibe como está definido e institucionalizado atualmente o processo de construção de projeto e do produto utilizado na Fábrica de Software apresentada.

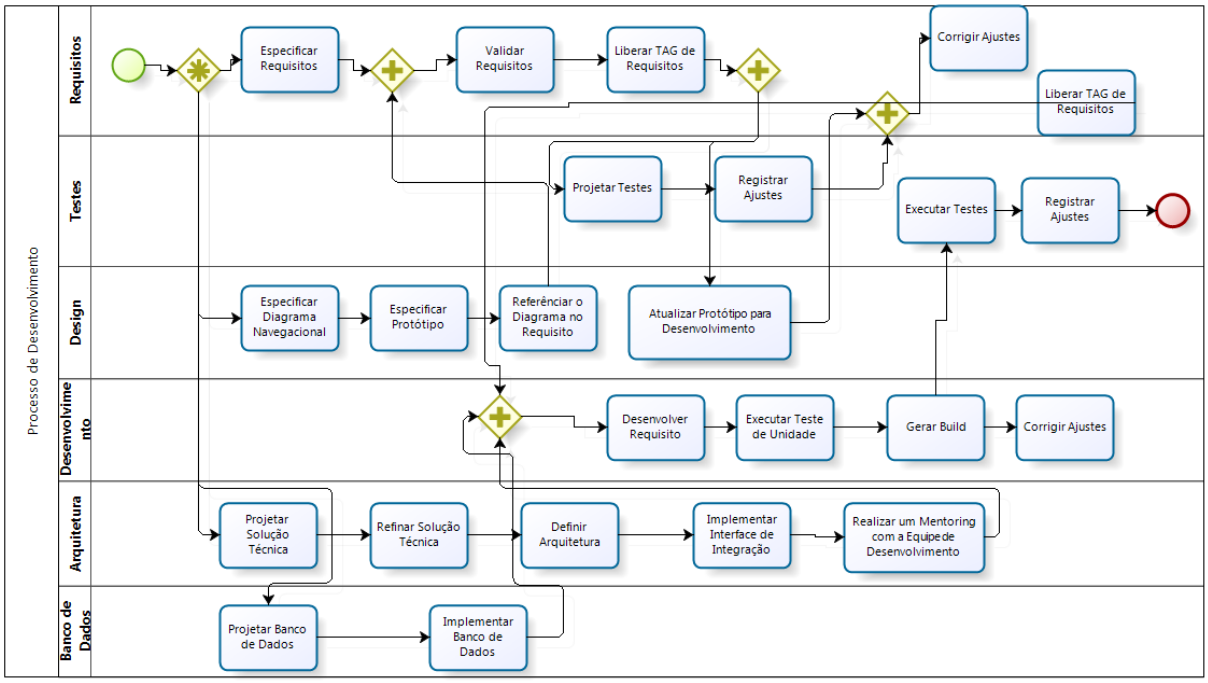

Figura. 1. Fluxo Detalhado da Fábrica de Software.

Apêndice D - Resultado do Questionário

Este apêndice apresenta os resultados coletados após a avaliação qualitativa, conforme descrito abaixo:

Tabela 6-12 – Resultado da pergunta: Tipo de curso?

Tipo de curso?				
Engenharia	**Informática / Ciência da Computação**	**Matemática**	**Outro(s):**	**Total**
10%	90%	0%	0%	100%

Tabela 6-13 – Resultado da pergunta: Grau de escolaridade?

Grau de escolaridade?				
Ensino Médio Técnico	**Graduação**	**Pós Graduação**	**MBA**	**Total**
	50%	50%		100%

Tabela 6-14 – Resultado da pergunta: Como você classificaria seu entendimento sobre o desenvolvimento de produto de software?

Como você classificaria seu entendimento sobre o desenvolvimento de produto de software?						
Excelente	Alto	Bom	Médio	Baixo	Nenhum	Total
21%	36%	29%	14%	0%	0%	**100%**

Tabela 6-15 – Resultado da pergunta: Experiência ou a atividades (cargo) que mais já exerceu, ou exerce, na área da computação?

Experiência ou a atividades (cargo) que mais já exerceu, ou exerce, na área da computação?									
Analista de Sistemas	**Codificação**	**Gerente de Processo**	**Gerente de Projeto**	**Projetista de Processo**	**Projetista de Sistemas**	**SQA**	**Teste**	**Usuário**	**Total**
36%	64%	0%	0%	0%	0%	0%	0%	0%	**100%**

Tabela 6-16 –Resultado da pergunta: Já participou do desenvolvimento de que tipo de sistemas de computação?

Já participou do desenvolvimento de que tipo de sistemas de computação?			
Grande Porte	**Médio Porte**	**Pequeno Porte**	**Total**
93%	7%	0%	**100%**

Tabela 6-17 - Em qual área da Engenharia de Software você mais atuou?

Em qual área da Engenharia de Software você mais atuou?						
Especificação de Requisitos	**Projeto**	**Codificação**	**Teste**	**Manutenção**	**Garantia / Controle da Qualidade**	**Total**
0%	14%	86%	0%	0%	0%	**100%**

Tabela 6-18 - Como você classificaria seu entendimento em desenvolvimento de software?

Como você classificaria seu entendimento em desenvolvimento de software?						
Excelente	**Alto**	**Bom**	**Médio**	**Baixo**	**Nenhum**	**Total**
21%	43%	36%	0%	0%	0%	**100%**

Tabela 6-19 - Já participou de quantos processos de software?

Já participou de quantos processos de software?				
Nenhum	**Entre 1 e 2**	**Entre 3 e 7**	**Acima de 7**	**Total**
0%	64%	36%	0%	**100%**

Tabela 6-20 - Como você classificaria seu entendimento sobre qualidade de software?

Como você classificaria seu entendimento sobre qualidade de software?						
Excelente	**Alto**	**Bom**	**Médio**	**Baixo**	**Nenhum**	**Total**
0%	7%	57%	36%	0%	0%	**100%**

Tabela 6-21 - Já usou quantos modelos/normas da qualidade?

Já usou quantos modelos/normas da qualidade?				
Nenhum	**Entre 1 e 2**	**Entre 3 e 7**	**Acima de 7**	**Total**
29%	64%	7%	0%	**100%**

Tabela 6-22 – Resultado do questionário.

Nº	Equipe	Atividade	Presença				Utilidade			Adequação		
			1	2	3	4	1	2	3	1	2	3
1	Requisitos	Especificar Requisitos	7%	0%	7%	86%	0%	0%	100%	64%	36%	0%
2		Validar Requisitos	0%	0%	57%	43%	0%	29%	71%	43%	57%	0%
3		Liberar TAG de Requisitos	0%	0%	7%	93%	0%	14%	86%	29%	64%	7%
4		Corrigir Ajustes no Requisito	0%	0%	36%	64%	0%	7%	93%	50%	50%	0%
5	Testes	Projetar Testes	0%	0%	36%	64%	0%	21%	79%	57%	43%	0%
6		Registrar Ajustes Requisitos	0%	7%	21%	71%	0%	29%	71%	29%	57%	14%
7		Executar Testes	0%	0%	21%	79%	0%	14%	86%	29%	64%	7%
8		Registrar Ajustes Código	0%	7%	14%	79%	0%	14%	86%	43%	43%	14%
9	Design	Especificar Diagrama Navegacional	0%	29%	57%	14%	0%	36%	64%	57%	43%	0%
10		Especificar Protótipo	0%	14%	21%	64%	0%	29%	71%	50%	50%	0%
11		Referenciar o Diagrama de Requisitos	0%	57%	7%	36%	0%	50%	50%	57%	43%	0%
12		Atualizar Protótipo para Desenvolvimento	0%	21%	36%	43%	0%	36%	64%	50%	43%	7%
13	Desenvolvimento	Desenvolver Requisito	0%	0%	7%	93%	0%	7%	93%	14%	86%	0%
14		Executar Testes de Unidade	0%	0%	29%	71%	0%	14%	86%	57%	29%	14%
15		Gerar Build	0%	0%	7%	93%	0%	14%	86%	0%	100%	0%
16		Corrigir Ajustes	0%	7%	21%	71%	0%	7%	93%	14%	79%	7%
17	Arquitetura	Projetar Solução Técnica	0%	0%	57%	43%	0%	7%	93%	64%	36%	0%
18		Refinar Solução Técnica	0%	7%	21%	71%	0%	21%	79%	50%	43%	7%
19		Definir Arquitetura	0%	0%	29%	71%	0%	7%	93%	36%	64%	0%
20		Implementar Interface de Integração	7%	7%	7%	79%	0%	29%	71%	14%	79%	7%

Nº	Equipe	Atividade	Presença				Utilidade			Adequação		
			1	2	3	4	1	2	3	1	2	3
21		Realizar um *Mentoring* com a Equipe	**0%**	**50%**	**43%**	**7%**	**0%**	**36%**	**64%**	**71%**	**21%**	**7%**
22	Banco de Dados	Projetar Banco de Dados	**0%**	**7%**	**14%**	**79%**	**0%**	**21%**	**79%**	**36%**	**64%**	**0%**
23		Implementar Banco de Dados	**0%**	**0%**	**7%**	**93%**	**0%**	**14%**	**86%**	**7%**	**93%**	**0%**

Apêndice E - Tabela de Atividades do Processo de Desenvolvimento

Este apêndice apresenta a tabela de todas as atividades do processo de desenvolvimento de software, conforme descrito abaixo:

Especificar Requisitos

<table>
<tr><th colspan="2">Objetivo</th></tr>
<tr><td colspan="2">• Esta atividade contempla a criação do documento de Regra de Negócio e Caso de Uso. Após a criação estes insumos serão liberados para o restante da Fábrica.</td></tr>
<tr><th>Critérios de Entrada</th><th>Artefatos de Entrada</th></tr>
<tr><td>• Reunião com o cliente</td><td>• Atas de Reunião</td></tr>
<tr><th colspan="2">Ações</th></tr>
<tr><td colspan="2">• O Analista de Requisito faz a elicitação junto ao cliente
• Criar uma Ata de reunião, contemplando as informações que o cliente relatou. Esta ata servirá como base para criação dos insumos de requisitos e regras.
• Finalizar a especificação do requisito após consenso na apresentação da solução inicial;
• Atualizar a Lista de requisitos e lista de atores (se necessário);
• Finalizar o documento de Definição de Regras de Negócio e de Definição de Interface necessárias ao Requisito;</td></tr>
<tr><th>Critérios de Saída</th><th>Artefatos de Saída</th></tr>
<tr><td>• Não se aplica</td><td>• Documento Regra de Negocio
• Documento de Caso de Uso
• Tabela de Mensagens
• Documento de Integração</td></tr>
<tr><th colspan="2">Responsáveis/ Papéis Envolvidos</th></tr>
<tr><td colspan="2">• Analista de Requisitos</td></tr>
<tr><th colspan="2">Templates</th></tr>
<tr><td colspan="2">• Template de Caso de Uso
• Template de Regra de Negócio</td></tr>
<tr><th colspan="2">Procedimentos</th></tr>
<tr><td colspan="2">• Não se aplica.</td></tr>
<tr><th colspan="2">Ferramentas de Apoio Utilizadas</th></tr>
<tr><td colspan="2">• Editor de Texto;
• Ferramenta de Modelagem UML;
• Sistema de BugTracking.</td></tr>
</table>

Validar Requisitos

Objetivo	
• Esta atividade tem como objetivo realizar a validação os artefatos gerados na Atividade de Especificação de Requisitos. Após a validação os artefatos são liberados para a Fábrica de Software.	
Critérios de Entrada	**Artefatos de Entrada**
• Regra de negócio especificado • Caso de Uso especificado	• Regra de Negócio • Caso de Uso
Ações	
• Validar com o cliente os artefatos gerados. • Anexar a Ata de reunião para uma eventual consulta do cliente. • O Gerente de Projeto deve consolidar as solicitações de reunião; • O Gerente de Projeto deve solicitar o agendamento das reuniões de validação para a equipe técnica do Cliente; • A Equipe técnica do Cliente agenda as reuniões com os Gestores do Negócio; • O Analista de Requisitos deve realizar reuniões com os Gestores do Negócio; • O Analista de Requisitos deve gerar a Ata de Reunião com as anotações da reunião.	
Critérios de Saída	**Artefatos de Saída**
• Receber o aceite do cliente	• Documento de aceite do cliente
Responsáveis/ Papéis Envolvidos	
• Equipe Técnica: Analista de Requisitos e o Cliente	
Templates	
• *Template* de Documento de Aceite do Cliente	
Procedimentos	
• Deve-se validar o requisito em uma reunião com o cliente.	
Ferramentas de Apoio Utilizadas	
• Editor de Texto	

Liberar TAG de Requisitos

Objetivo	
• Esta atividade também pode ser chamada de *Baseline*, onde seu objetivo é "marcar" uma linha no tempo em um conjunto de insumos, que servirão de base para outras áreas da Fábrica. O motivo da marcação, é que o requisito está em constante evolução.	
Critérios de Entrada	**Artefatos de Entrada**
• Não se aplica	• Solicitação de *baseline* atualizada. • E-mail de comunicação da nova *baseline*
Ações	
• Através do controle de versão, criar uma marcação em um conjunto de insumos. Esta marcação faz-se necessário devido à constante evolução no requisito. • Liberar TAG(*baseline*) de insumos para o restante da Fábrica. • Caso ao final desta auditoria não conformidades sejam detectadas, o Analista de Configuração deve reportar ações por e-mail para a resolução destas não conformidades e acompanhar de acordo com a atividade Acompanhar Ação constante no processo de Gestão de Configuração a resolução destas não conformidades; • Caso não conformidades não sejam detectadas, o Analista de Configuração deverá aplicar o rótulo específico nos artefatos indicados para caracterizar a nova *baseline*. A regra da nomenclatura destes rótulos deve estar definida no Plano de Gerência de Configuração; • Após o estabelecimento da *baseline*, o Analista de Configuração comunica a liberação da nova *baseline* aos grupos interessados, via e-mail.	
Critérios de Saída	**Artefatos de Saída**
• Notificar todas as áreas envolvidas.	• *TAG* de um conjunto de requisitos • E-mail de comunicação da nova *baseline*.

Responsáveis/ Papéis Envolvidos
• Analista de Configuração
Templates
• Não se aplica
Procedimentos
• Guia de Princípios de Gestão de Configuração
Ferramentas de Apoio Utilizadas
• Ferramenta de E-mail; • Ferramenta de Controle de Versão.

Corrigir Ajustes

Objetivo	
• Esta atividade tem como objetivo a correção de inconsistências encontradas na atividade de elicitação de Requisitos. Que pode ocorrer durante o desenvolvimento ou na fase de testes.	
Critérios de Entrada	**Artefatos de Entrada**
• Registrar ajustes no Sistema de *BugTracking*	• Regra de Negócio • Caso de Uso
Ações	
• Realizar a correção nos insumos com inconsistências, relatadas pela equipe de Teste.	
Critérios de Saída	**Artefatos de Saída**
• Corrigir insumos com inconsistências	• Regra de Negócio corrigido • Caso de Uso corrigido
Responsáveis/ Papéis Envolvidos	
• Analista de Requisitos, Administrador de BD, Arquiteto, Projetista de Software, Projetista de Interface, Analista de Teste e Engenheiro de Software	
Templates	
• *Template* de Ata de Reunião de Viabilidade do Projeto • *Template Checklist* da viabilidade técnica e funcional	
Procedimentos	
• Não se aplica	
Ferramentas de Apoio Utilizadas	
• Editor de Texto • Ambiente de Desenvolvimento	

Projetar Testes

Objetivo	
• Esta atividade tem realiza a criação do artefato de projeto de testes. Este artefato tem como objetivo orientar a equipe de testes durante a execução nos testes do sistema	
Critérios de Entrada	**Artefatos de Entrada**
• Não se aplica	• Regra de Negócio • Caso de Uso
Ações	
• Elaborar casos e cenários de teste descrevendo os passos a serem realizados e os critérios de sucesso ou falha utilizando como referência uma versão estável dos documentos de requisitos; • Estabelecer itens alvo de teste; • Definir procedimentos de aplicação das técnicas e estratégias.	
Critérios de Saída	**Artefatos de Saída**
• Não se aplica	• Projeto de Teste
Responsáveis/ Papéis Envolvidos	
• Analista de Teste	
Templates	
• *Template* de Projeto de Teste	
Procedimentos	

• Guia de Orientações para os Casos de Testes
Ferramentas de Apoio Utilizadas
• Editor de Texto; • Planilha Eletrônica.

Registrar Ajustes Requisitos

Objetivo	
• Registrar no sistema de *Bug Tracking*, alguma consistência encontrada nos artefatos gerados pela equipe de Requisitos	
Critérios de Entrada	**Artefatos de Entrada**
• Regra de negócio especificado • Caso de Uso especificado	• Regra de Negócio • Caso de Uso
Ações	
• Quando necessário, o Analista de Requisitos deve ler a Ata de Reunião, contendo as Anotações da Reunião de Aprovação do Requisito; • O Analista de Requisitos deve realizar os ajustes nos documentos gerados pela fase de Requisitos. • Registrar no sistema de *BugTracking* as inconsistências encontradas nos artefatos gerados pela equipe de Requisitos.	
Critérios de Saída	**Artefatos de Saída**
• Registrar no sistema de *BugTracking*, inconsistências encontradas nos artefatos de Requisito	• Regra de Negócio • Caso de Uso
Responsáveis/ Papéis Envolvidos	
• Equipe Técnica: Analista de Teste	
Templates	
• *Template* de Caso de Uso • *Template* de Regra de Negócio	
Procedimentos	
• Não se aplica	
Ferramentas de Apoio Utilizadas	
• Editor de Texto; • Ferramenta de Modelagem UML; • Sistema de *BugTracking*.	

Executar Testes

Objetivo	
• Esta atividade tem como objetivo a execução dos testes no sistema, baseado no artefato de Projeto de Testes. Com a finalidade de encontrar possíveis inconsistências no sistema.	
Critérios de Entrada	**Artefatos de Entrada**
• Regra de negócio validado • Caso de uso validado • Projeto de Testes pronto	• Regra de Negócio • Caso de Uso em definição • Projeto de Testes
Ações	
• Preparar ambiente de testes, conforme planejado no Plano de Testes; • Preparar a Planilha de Resultado de Testes para a rodada de testes com os testes a serem realizados (casos de testes ou *checklist*) e a identificação da versão a ser testada; • Executar os casos de testes planejados para a rodada e registrar os resultados na Planilha de Resultado de Testes; • Verificar, caso existam, as mudanças já corrigidas na ferramenta de Gestão de Mudanças do Projeto; • Registrar Solicitações de Mudança na ferramenta de Gestão de Mudanças do Projeto, quando necessário.. •	

Critérios de Saída	**Artefatos de Saída**
• Não se aplica	• Relatório de Testes
Responsáveis/ Papéis Envolvidos	
• Analista de Teste	
Templates	
• *Template* de Relatório de Testes	
Procedimentos	
• Guia de Orientações para os Casos de Testes	
Ferramentas de Apoio Utilizadas	
• Planilha Eletrônica; • Ferramenta de Gestão de Mudanças.	

Registrar Ajustes Código

Objetivo	
• Registrar no sistema de *Bug Tracking*, alguma consistência encontrada no sistema	
Critérios de Entrada	**Artefatos de Entrada**
• Aplicação disponibilizada para testes	• Não se aplica
Ações	
• Registrar no sistema de *BugTracking* as inconsistências encontradas nos artefatos gerados pela equipe de Desenvolvimento.	
Critérios de Saída	**Artefatos de Saída**
• Registrar no sistema de *BugTracking*, inconsistências encontradas no sistema	• Não se aplica
Responsáveis/ Papéis Envolvidos	
• Analista de Teste e Engenheiro de Software	
Templates	
• *Não se aplica*	
Procedimentos	
• Não se aplica	
Ferramentas de Apoio Utilizadas	
• Sistema de *BugTracking* • Aplicação disponibilizada para testes	

Especificar Diagrama Navegacional

Objetivo	
• Esta atividade tem o objetivo de especificar o diagrama navegacional da aplicação. Este diagrama orienta onde uma determinada funcionalidade ficará localizada na aplicação.	
Critérios de Entrada	**Artefatos de Entrada**
• Regra de negócio especificado • Caso de uso iniciada a especificação	• Regra de Negócio • Caso de Uso em definição
Ações	
• Especificar diagrama navegacional	
Critérios de Saída	**Artefatos de Saída**
• Não se aplica	• Diagrama Navegacional
Responsáveis/ Papéis Envolvidos	
• Projetista de Interface	
Templates	
• *Não se aplica*	
Procedimentos	
• Não se aplica	
Ferramentas de Apoio Utilizadas	
• Ferramenta de Fluxogramas	

Especificar Protótipo

Objetivo	
• O objetivo desta atividade é a elaboração do protótipo de interface do sistema. Este elaboração apoia a elaboração do requisito e no desenvolvimento do código-fonte.	
Critérios de Entrada	**Artefatos de Entrada**
• Caso de Uso em análise	• Regra de Negócio • Caso de Uso
Ações	
• Ler o requisito em questão; • Elaborar/Atualizar protótipo (navegável) caso necessário; • Caso necessário produzir imagem de ícones; • Validar junto ao analista de requisitos.	
Critérios de Saída	**Artefatos de Saída**
• Não se aplicam	• Plano de Interface do Usuário • Protótipo
Responsáveis/ Papéis Envolvidos	
• Analista de Interface	
Templates	
• *Template* do Plano de Interface do Usuário	
Procedimentos	
• Critérios de usabilidade • Padrão de interface	
Ferramentas de Apoio Utilizadas	
• Ferramenta de Design; • Ferramenta de Fluxogramas; • Editor Gráfico; • Editor de HTML.	

Referenciar Diagrama no Requisito

Objetivo	
• Esta atividade tem o objetivo de referenciar o diagrama de navegacional do sistema em um documento de Caso de Uso	
Critérios de Entrada	**Artefatos de Entrada**
• Não se aplica	• Caso de Uso
Ações	
• Referenciar o diagrama navegacional no documento de Caso de Uso.	
Critérios de Saída	**Artefatos de Saída**
• Não se aplica	• Diagrama Navegacional
Responsáveis/ Papéis Envolvidos	
• Projetista de Interface	
Templates	
• *Não se aplica*	
Procedimentos	
• Não se aplica	
Ferramentas de Apoio Utilizadas	
• Ferramenta de Fluxogramas	

Atualizar Protótipo Para Desenvolvimento

Objetivo	
• O objetivo desta atividade é a após a validação dos artefatos gerados pela equipe de requisito, atualizar o protótipo para servir de apoio na atividade de desenvolvimento da aplicação.	
Critérios de Entrada	**Artefatos de Entrada**
• Liberar TAG de Requisitos	• Regra de Negócio • Caso de Uso • Protótipo
Ações	
• Verificação de componentes novos • Atualização do protótipo • Verificar a sequência de campos de acordo com a sequência lógica de preenchimento.	
Critérios de Saída	**Artefatos de Saída**
• Não se aplicam	• Plano de Interface do Usuário • Protótipo
Responsáveis/ Papéis Envolvidos	
• Analista de Interface	
Templates	
• *Template* do Plano de Interface do Usuário	
Procedimentos	
• Critérios de usabilidade • Padrão de interface	
Ferramentas de Apoio Utilizadas	
• Ferramenta de Design; • Ferramenta de Fluxogramas; • Editor Gráfico; • Editor de HTML.	

Projetar Solução Técnica

Objetivo	
• O Objetivo desta atividade é definir se é possível através de um *checklist* determinar a viabilidade da codificação de um determinado requisito elicitado.	
Critérios de Entrada	**Artefatos de Entrada**
• Regra de negócio especificado • Caso de uso iniciada a especificação	• Regra de Negócio • Caso de Uso em definição
Ações	
• Caso não seja possível definir uma solução para o requisito, é realizado uma prova de conceito, ou seja, um teste sobre as ferramentas e/ou tecnologias, assim podendo determinar qual o melhor para ser utilizado. • Definir soluções técnicas para os requisitos em questão • Aplicação de critérios e objetivos a partir do *checklist*. • As reuniões são conduzidas pela equipe de arquitetos juntamente com os analistas de negócios e os projetistas do sistema. • A frequência desta reunião é sempre no início de um módulo	
Critérios de Saída	**Artefatos de Saída**
• Gerar a Ata de Reunião de Viabilidade do Projeto • Critérios da solução técnica verificados e avaliados.	• Ata de Reunião de Viabilidade do Projeto • *Checklist* preenchido
Responsáveis/ Papéis Envolvidos	
• Equipe Técnica: Analista de Requisitos, Administrador de BD, Arquiteto, Projetista de Software, Projetista de Interface, Analista de Teste e Engenheiro de Software	
Templates	
• *Template* de Ata de Reunião de Viabilidade do Projeto	

• *Template Checklist* da viabilidade técnica e funcional
Procedimentos
• Padrão de critérios de solução técnica.
Ferramentas de Apoio Utilizadas
• Editor de Texto • Ambiente de Desenvolvimento

Refinar solução técnica

Objetivo	
• O objetivo desta atividade é refinar a solução técnica, ou seja, aprimorar e refinar a definição inicial da equipe de arquitetura, analisando principalmente possíveis impactos em outras funcionalidades da aplicação.	
Critérios de Entrada	**Artefatos de Entrada**
• Analisar os documentos gerados pela equipe de Requisitos	• Documento de Regra de Negócio • Especificação de Caso de Uso •
Ações	
• Analisar a solução técnica definida na atividade de Projetar Solução técnica. • Verificar e analisar possíveis impactos com outros módulos. • Verificar as integrações com outros sistemas	
Critérios de Saída	**Artefatos de Saída**
• Não se aplica.	• Não se aplica
Responsáveis/ Papéis Envolvidos	
• Arquiteto de Software e Analista de Requisito	
Templates	
• Não se aplica	
Procedimentos	
• Não se aplica	
Ferramentas de Apoio Utilizadas	
• Editor de Texto. • Ferramenta de UML	

Definir Arquitetura

Objetivo	
• O objetivo desta Atividade é definir a Arquitetura do Software • A arquitetura deve ser elaborada ou atualizada de acordo com as solicitações dos requisitos • Caso a arquitetura atenda o requisito, o fluxo segue seu caminho normalmente até o desenvolvimento, testes e a entrega do produto. • Caso a arquitetura não atenda, então é necessário realizar ajustes e então depois seguir o fluxo do processo. • A arquitetura de software de um sistema consiste dos componentes de software, suas propriedades externas, e seus relacionamentos com outros softwares.	
Critérios de Entrada	**Artefatos de Entrada**
• Artefatos de entrada analisados. • Módulos implementados verificados • Relacionamento dos módulos verificado	• Documento de Regra de Negócio • Especificação de Caso de Uso • *Checklist* da viabilidade técnica e funcional • Ata de Reunião de Viabilidade do Projeto
Ações	
• Projetar a Arquitetura para atender a solicitação • Esta arquitetura define a estrutura do software que suporta os principais requisitos especificados, facilitando uma provável integração com novos requisitos.	

• A documentação da arquitetura descreve as decisões do projeto e os padrões adotados para resolver os problemas ocorridos. • A documentação da arquitetura do software facilita: a comunicação entre os *stakeholders*, registra as decisões iniciais acerca do projeto de alto-nível, e permite o reuso do projeto dos componentes e padrões entre projetos. • o Analista de Sistemas/Arquiteto juntamente com o Analista de Sistemas/Requisitos, usando como base o Plano do Projeto, os produtos gerados na fase de Iniciação e das atividades iniciais de Execução, serão responsáveis pela definição do Plano de Arquitetura. • O Plano de Arquitetura devem fornecer uma visão geral de arquitetura do sistema e devem • ser executados durante o ciclo de vida do projeto. Importante desta fase é a definição das • interfaces de integração dos produtos gerados para o desenvolvimento do software. • O Cliente deve proceder à aprovação de algumas partes do Plano de Arquitetura.

Critérios de Saída	**Artefatos de Saída**
• Gerar uma arquitetura para o sistema • Atualizar a arquitetura e o artefato de saída, caso a arquitetura sofra alteração.	• Plano de Arquitetura
Responsáveis/ Papéis Envolvidos	
• Arquiteto de Software	
Templates	
• *Template* do Plano de Arquitetura	
Procedimentos	
• Plano de Arquitetura atualizado	
Ferramentas de Apoio Utilizadas	
• Editor de Texto. • Ferramenta de UML	

Implementar Interface de Integração

Objetivo	
• O objetivo desta Atividade é realizar a codificação para integração com sistemas externos, através de *WebService*.	
Critérios de Entrada	**Artefatos de Entrada**
• Arquitetura definida	• Código-Fonte
Ações	
• Criar métodos que suportem a integração com sistemas externos.	
Critérios de Saída	**Artefatos de Saída**
• Não se aplica	• Código-fonte
Responsáveis/ Papéis Envolvidos	
• Arquiteto de Software	
Templates	
• Não se aplica.	
Procedimentos	
• Não se aplica.	
Ferramentas de Apoio Utilizadas	
• Ambiente de Desenvolvimento.	

Realizar *Mentoring* com a Equipe de Desenvolvimento

Objetivo	
• O objetivo desta atividade é realizar um *mentoring* para a equipe de desenvolvimento, apresentando a arquitetura do sistema, assim como as boas práticas de programação.	
Critérios de Entrada	**Artefatos de Entrada**
• Arquitetura Finalizada	• Não se aplica
Ações	
• Apresentar a arquitetura para a equipe de Desenvolvimento • Explicar formas de otimizar o desenvolvimento de código	

• Explicar e orientar a programação em conjunto com os *Frameworks*

Critérios de Saída	**Artefatos de Saída**
• Não se aplica	• Não se aplica

Responsáveis/ Papéis Envolvidos
• Arquiteto de Software e Engenheiro de Software
Templates
• Não se aplica
Procedimentos
• Não se aplica
Ferramentas de Apoio Utilizadas
• Ambiente de Desenvolvimento • Apresentação em Slides

Projetar Banco de Dados

Objetivo	
• Esta atividade tem como objetivo criar o projeto do Banco de Dados.	
Critérios de Entrada	**Artefatos de Entrada**
• Analisar os artefatos de entrada • Analisar a especificação do caso de uso, na seção "Atributos"	• Regra de Negócio • Caso de Uso
Ações	
• Verifica os fluxos principal e alternativo, especificados nos casos de uso da aplicação. • Verificar os modelos de dados já definidos para outros módulos funcionais da aplicação, para prover a integração da definição corrente.	
Critérios de Saída	**Artefatos de Saída**
• Estruturas de dados criadas no banco de dados	• Scripts de criação de estruturas de banco de dados • Dicionário de Dados
Responsáveis/ Papéis Envolvidos	
• Analista de Dados	
Templates	
• *Template* de Dicionário de Banco de Dados	
Procedimentos	
• Convenção da Base de Dados	
Ferramentas de Apoio Utilizadas	
• Ferramenta de Modelagem de banco de dados. • Editor de texto	

Implementar Banco de Dados

Objetivo	
• Esta atividade tem com objetivo realizar a criação do Banco de Dados da aplicação.	
Critérios de Entrada	**Artefatos de Entrada**
• Analisar os artefatos de entrada • Analisar a especificação do caso de uso, na seção "Atributos"	• Documento de Regra de Negócio • Especificação de Caso de Uso
Ações	
• O Analista de dados realiza a criação do: o *SCHEMA* o *SEQUENCE* o Chaves primárias o Chaves estrangeiras • Criação de *View* quando necessário; • Sempre que possível, deve-se evitar a redução de nomes das tabelas ou *views*; • Criação das *Sequences* das colunas chaves.	

• Criação de *Stored Procedures* (Se necessário) • Criação de *Functions* (Se necessário) • Criação de *Trigger* (Se necessário) • Verificar o documento que contempla a nomenclatura das tabelas. • Analisar o relacionamento entre tabelas

Critérios de Saída	**Artefatos de Saída**
• Não se aplica	• Banco de Dados
Responsáveis/ Papéis Envolvidos	
• Analista de Dados	
Templates	
• *Template* de Dicionário de Banco de Dados	
Procedimentos	
• Não se aplica	
Ferramentas de Apoio Utilizadas	
• Ferramenta de Modelagem de banco de dados. • Editor de texto	

Planejar Migração de Dados.

Objetivo	
• Esta atividade tem como objetivo analisar o banco de dados do sistema atual(legado) e elabora uma estratégia para realizar uma carga no sistema que está sendo desenvolvido. Esta atividade é opcional, ocorrendo somente nas funcionalidades já existentes do sistema legado.	
Critérios de Entrada	**Artefatos de Entrada**
• As estruturas de dados do sistema que está sendo desenvolvido devem estar criadas • Estrutura de dados do sistema legado deve está mapeada. • Verificar o volume dos dados contidos nas tabelas.	• Documento de Regra de Negócio • Especificação de Caso de Uso • Dicionário de Dados do sistema legado, caso exista. • Dicionário de Dados do sistema que está sendo desenvolvido. • Banco de dados do sistema legado. • Banco de dados do sistema em desenvolvimento.
Ações	
• Analisar o sistema de banco de dados já existente. • Criar um mapeamento das tabelas e colunas para serem migradas para o sistema que está sendo desenvolvido. • Ao final é importante verificar como as exceções (dados inconsistentes) serão tratadas. • Quando for necessário, com base no Diagrama de Banco de Dados definido e na estrutura do banco de dados legado, o Analista de Banco de Dados e o Administrador de Banco de Dados especificam um planejamento, contendo regras e premissas, para a execução da migração do banco de dados a ser realizada para a operação e testes do projeto usando a estrutura-base definida no Diagrama de Banco de Dados. • O interesse nesta fase é o planejamento do porte para a nova estrutura de banco de dados, os dados atuais em operação no cliente, a fim de prover o legado destes dados.	
Critérios de Saída	**Artefatos de Saída**
• Os dados que estão sendo migrados foram verificados. • Estruturas de dados devidamente preenchidas com os dados migrados do sistema legado.	• Plano de Migração de Dados.
Responsáveis/ Papéis Envolvidos	
• Analista de Dados	
Templates	
• *Template* do Plano de Migração de Dados	
Procedimentos	

• Convenção da Base de Dados
Ferramentas de Apoio Utilizadas
• Ferramenta de Modelagem de banco de dados. • Ferramenta para migração do banco de dados. • Editor de Texto. • Sistema de Gerenciamento de Banco de Dados (SGBD)

Migrar de Dados.

Objetivo	
• Após o planejamento da migração, esta atividade tem o objetivo de executar a migração de dados do sistema legado para o banco de dados do sistema que está sendo desenvolvido.	
Critérios de Entrada	**Artefatos de Entrada**
• Migração de dados planejada.	• Plano de Migração de Dados. • Dicionário de Dados do sistema legado, caso exista. • Dicionário de Dados do sistema que está sendo desenvolvido. • Banco de dados do sistema legado. • Banco de dados do sistema em desenvolvimento.
Ações	
• Comparar os dados entre a base de dados • Verifica layout dos dados que serão importados • Realizar o tratamento das exceções (dados inconsistentes) encontradas. • Realiza a migração de dados • Verifica em pequenas amostragens o conteúdo migrado.	
Critérios de Saída	**Artefatos de Saída**
• Banco de dados do sistema em desenvolvimento com os dados do banco de dados do sistema legado já migrados	• Não se aplica.
Responsáveis/ Papéis Envolvidos	
• Analista de Dados	
Templates	
• Não se aplica	
Procedimentos	
• Não se aplica	
Ferramentas de Apoio Utilizadas	
• Ferramenta de modelagem de banco de dados. • Ferramenta para migração do banco de dados. • Editor de Texto. • Sistema de Gerenciamento de Banco de Dados (SGBD)	

Desenvolver Requisitos

Objetivo	
• O objetivo desta Atividade é desenvolver o requisito que foi especificado.	
Critérios de Entrada	**Artefatos de Entrada**
• Arquitetura do sistema elaborada • Documento de modelo de implementação definido.	• Documento de Regra de Negócio • Especificação de Caso de Uso • Plano de arquitetura • Modelos de análises • Documento de modelo de implementação definido.
Ações	

• Desenvolver o requisito de acordo com o que está especificado nos Casos de Uso • Desenvolver o Caso de Uso de acordo com a arquitetura especificada • Codificar em cada camada do sistema de acordo com o que foi especificado no documento de arquitetura. • Deve seguir os padrões de codificação conforme definido no modelo de Implementação. • Corrigir os problemas relatados pela equipe de testes.

Critérios de Saída	**Artefatos de Saída**
• Unidade de Código-Fonte gerado • Teste unitário definido	• Código-Fonte

Responsáveis/ Papéis Envolvidos
• Engenheiro de Software
Templates
• Não se aplica
Procedimentos
• Documento de modelo de implementação definido. • Plano de Arquitetura.
Ferramentas de Apoio Utilizadas
• Editor de Texto • Ambiente de Desenvolvimento. • Ferramenta de *BugTracking*.

Executar Teste de Unidade

Objetivo
• O objetivo desta Atividade é realizar os testes de unidades no código fonte gerado e de integração com outros módulos ou através de serviços de integração com outro sistema externo.

Critérios de Entrada	**Artefatos de Entrada**
• Teste unitário definido • Unidade de Código-Fonte gerado	• Código-Fonte

Ações
• O engenheiro de software constrói testes de unidade para verificar a integração com outros sistemas externos; • Esses testes determinam se a comunicação e troca de informação está sendo realizada conforme especificada; • Verificar se os resultados da execução dos testes de unidade foram satisfatórios, segundo critérios de corretude e funcionalidade. Em caso negativo, verificar se os critérios de avaliação estão corretos ou se a própria implementação da funcionalidade apresenta inconformidade com as regras de negócio; • Nesta fase os Engenheiros de Software farão a execução dos testes unitários seguindo padrões e modelos especificados mediante o conhecimento da Especificação de Casos de Uso, do Plano de Arquitetura e dos artefatos gerados durante a análise e projeto dos Casos de Uso. • Realizar testes de unidade, para testar a persistência da aplicação com o banco de dados. • Após a conclusão dos testes unitários, os códigos fontes são liberados para que sejam realizados os testes integrados. • O teste unitário deve abranger somente a camada de negócios, isto é, não é obrigatório ocorrer a partir de interfaces gráficas que constituem a camada de apresentação e deve seguir especificação da ferramenta utilizada e os padrões de codificação definidos; • Caso durante os testes sejam identificadas algumas inconsistências, esta deverá ser registrada na ferramenta adotada para iniciar um processo de correção. • Quando necessário, a equipe deve realizar os Testes Funcionais com base no documento Testes Funcionais.

Critérios de Saída	**Artefatos de Saída**
• Teste unitário executado • Teste de integração executado. • Quando necessário, inconsistências	• Código fonte da classe de testes unitários; • Testes Funcionais, quando necessário.

registradas na ferramenta de *BugTracking*	
Responsáveis/ Papéis Envolvidos	
• Engenheiro de Software	
Templates	
• Não se aplica.	
Procedimentos	
• Não se aplica.	
Ferramentas de Apoio Utilizadas	
• Ambiente de Desenvolvimento.	

Gerar Build

Objetivo	
• O objetivo desta Atividade é realizar a liberação do produto ou parte do produto para homologação do cliente.	
Critérios de Entrada	**Artefato de Entrada**
• Produto ou sistema testado.	• Relatório de Testes do Sistema
Ações	
• Nesta atividade deve-se inicialmente fazer um *check-out* de todos os componentes do repositório principal, identificados na atividade Distribuir Demanda que fazem parte da implementação de tal demanda. Posteriormente integrar componentes em um build. • Verificar *checklist* de entregáveis • Criar uma TAG (rótulo que controla a versão em uma ferramenta automatizada) dos produtos de trabalho potencialmente entregáveis (Documentação + Código). • Gerar *Baseline* de entrega (*release*). • Colocar o produto ou parte do produto de maneira usável (atendendo a padrão de qualidade definidos pela equipe de teste, e acordados com o cliente) em um ambiente de homologação. • Realizar apresentação do produto entregue ao cliente.	
Critérios de Saída	**Artefato de Saída**
• Build do sistema gerado	• Build do Sistema • *Release notes* • Termo de entrega
Responsáveis/ Papéis Envolvidos	
• Gerente de configuração • Analista de requisitos	
Templates	
• *Template* de Termo de entrega. • *Template* do *Release notes*	
Procedimentos	
• Documento de gestão de configuração.	
Ferramentas de Apoio Utilizadas	
• Editor de Texto • Ferramenta de Controle de versão. • Ferramenta de *BugTracking*.	

Homologação

Objetivo	
Esta atividade tem como objetivo acompanhar a avaliação do software efetuada pelo cliente, de acordo com pré-requisitos definidos no Plano de Homologação.	
Critérios de Entrada	**Artefatos de Entrada**
• Agendar o período de homologação com o cliente.	• Plano de Homologação; • Pacote da Entrega.
Ações	
• Apresentar requisito(s) a ser(em) homologado(s); • Apresentar o Roteiro de Negócio para Homologação;	

<table>
<tr><td colspan="2">• Repetir os Testes Funcionais no ambiente de homologação, para assegurar o atendimento dos requisitos funcionais toda vez que houver uma liberação. Para esta ação, utilizar a documentação de sistema elaborada durante o desenvolvimento, como: especificação de requisitos, regras de negócio, tabelas de mensagens, roteiro de homologação;
• Discutir ocorrências encontradas com o Cliente e acompanhar o registro na Ferramenta de Gestão de Mudanças;
• Registrar Ações Diárias.</td></tr>
<tr><td>Critérios de Saída</td><td>Artefatos de Saída</td></tr>
<tr><td>• Não se aplica.</td><td>• Relatório de Homologação</td></tr>
<tr><td colspan="2">Responsáveis/ Papéis Envolvidos</td></tr>
<tr><td colspan="2">• Analista Homologador.</td></tr>
<tr><td colspan="2">Templates</td></tr>
<tr><td colspan="2">• Não se aplica</td></tr>
<tr><td colspan="2">Procedimentos</td></tr>
<tr><td colspan="2">• Não se aplica</td></tr>
<tr><td colspan="2">Ferramentas de Apoio Utilizadas</td></tr>
<tr><td colspan="2">• Ambiente de Homologação;
• Ferramenta de E-mail.</td></tr>
</table>