WESLEI ALVIM DE TARSO MARINHO

ADAPTAÇÃO DA FASE DE DESENVOLVIMENTO DO PROCESSO DE UMA EMPRESA DE PEQUENO PORTE PARA DESENVOLVIMENTO COM REUSO BASEANDO-SE NO CMMI

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS
MINAS GERAIS – BRASIL
2006

WESLEI ALVIM DE TARSO MARINHO

ADAPTAÇÃO DA FASE DE DESENVOLVIMENTO DO PROCESSO DE UMA EMPRESA DE PEQUENO PORTE PARA DESENVOLVIMENTO COM REUSO BASEANDO-SE NO CMMI

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração:

Engenharia e Qualidade de Software, Reuso

Orientadora:

Prof.ª Olinda Nogueira Paes Cardoso

LAVRAS

MINAS GERAIS – BRASIL

2006

Ficha Catalográfica preparada pela Divisão de Processo Técnico da Biblioteca Central da UFLA

Marinho, Weslei Alvim de Tarso

Adaptação da Fase de Desenvolvimento do Processo de uma Empresa de Pequeno Porte para Desenvolvimento com Reuso Baseando-se no CMMI

Monografia de Graduação – Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. Engenharia de Software. 2. Qualidade de Software. 3. Reuso. I. MARINHO, W. A. de T.. II. Universidade Federal de Lavras. III. Título.

WESLEI ALVIM DE TARSO MARINHO

ADAPTAÇÃO DA FASE DE DESENVOLVIMENTO DO PROCESSO DE UMA EMPRESA DE PEQUENO PORTE PARA DESENVOLVIMENTO COM REUSO BASEANDO-SE NO CMMI

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 27 de abril de 2006.

_____

Prof. Marcelo Silva de Oliveira

_____

Prof. Rêmulo Maia Alves

_____

Prof.ª Olinda Nogueira Paes Cardoso
(Orientadora)

LAVRAS

MINAS GERAIS – BRASIL

Dedico este trabalho a minha família pelo apoio e incentivo dispensados.

Agradeço...

A Deus, por ter me concedido a oportunidade de estar aqui.

À Prof.ª Olinda, pelo apoio e orientação concedidos na elaboração deste trabalho.

À Tamara, por me manter trabalhando.

Ao Rafael e Leandro, pelo auxílio no trabalho.

Ao Leonardo, pela constante ajuda.

A todos da SWQuality, por estarem sempre presentes e prestativos.

Aos Professores Marcelo e Rêmulo, por estarem dispostos a avaliar este trabalho.

A todos aqueles que, de alguma forma, me auxiliaram no decorrer do curso.

RESUMO

Resumo: este trabalho apresenta a adaptação da fase de desenvolvimento de um processo de uma pequena empresa produtora de software para o desenvolvimento com reuso utilizando o *Capability Maturity Model Integration* (CMMI). O CMMI é um método para avaliação e medição da maturidade do processo de desenvolvimento de software elaborado pelo *Software Engineering Institute* (SEI). Foi realizado o estudo de algumas áreas de processo, analisando as atividades e os produtos típicos de trabalho relacionados com reuso. A partir desta análise foram realizadas as adaptações do processo para reuso incluindo novas atividades e artefatos. Esta adaptação engloba a inserção de várias novas atividades, a modificação de atividades já existentes e a criação de novos papéis e artefatos na fase do processo.

Palavras-chaves: componentes de software, desenvolvimento baseado em componentes, melhoria do processo de software, CMMI.

ABSTRACT

Absctract: *This work presents the adaptation of the development's phase of a software development company's process for the development with reuse using CMMI, Capability Maturity Model Integration, a method for evaluation and measuring of the software development's process developed by the SEI, Software Engineering Institute. A study of some process areas was done analyzing activities and typical work products related to reuse. The process's adaptations for reuse was made using this analysis for include new activities and artifacts. This adaptation embodies the insertion of several new activities, the existing activities' modification and the creation of new roles and artifacts.*

Keywords: *software components, component-based development, software process improvement, CMMI.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

CBD - *Component-based Development*

CBSE - *Component-based Software Engineering*

CMMI – *Capability Maturity Mode Integration*

COM - *Component Object Model*

EJB - *Enterprise Java Beans*

IEEE – *Institute of Electrical and Electronic Engineers*

MCT – Ministério da Ciência e Tecnologia

PEPP – Processo de Software para Empresas de Pequeno Porte descrito por Aguiar (2004)

SEI – *Software Engineering Institute*

SG – *Specific Goal*

SP – *Specific Practice*

1. INTRODUÇÃO

Neste primeiro capítulo apresenta-se uma breve introdução do trabalho proposto, juntamente com as motivações para a realização do mesmo, bem como seus objetivos e justificativas.

1.1. Objetivos e Justificativas

Segundo Rossi (2004), as organizações estão cada vez mais dependentes de sistemas de informação para realização de seus negócios. Com isto, uma das preocupações na área de desenvolvimento de software é a obtenção de sistemas que atendam as necessidades atuais e que sejam flexíveis para acompanhar as mudanças tecnológicas e práticas de negócio.

Durante a década de 1990 várias pesquisas referentes ao desenvolvimento baseadas em componentes, nas tecnologias envolvidas e nas formas de reutilização de componentes foram desenvolvidas. As pesquisas mostram que a reutilização de componentes de software tem sido considerada uma das formas para se obter redução dos custos e do tempo de desenvolvimento e aumento da produtividade e da qualidade do produto de software (Rossi, 2004).

O objetivo deste trabalho é realizar a adaptação da fase de Desenvolvimento de um processo de uma empresa de pequeno porte, que trabalha com desenvolvimento de software, para o desenvolvimento com reuso utilizando o *Capability Maturity Model Integration* (CMMI). O CMMI é um modelo para avaliação e medição da maturidade do processo de desenvolvimento de software desenvolvido pelo *Software Engineering Institute* - SEI.

Para alcançar este objetivo foram planejadas três atividades principais. Primeiramente a realização de um estudo bibliográfico relacionado ao desenvolvimento de software com reuso. Após esta primeira atividade, a realização de um estudo mais aprofundado de algumas áreas de processo do CMMI, identificadas como relacionadas ao reuso, análise das atividades e os produtos típicos de trabalho mais relevantes. A seguir, a aplicação desta análise, resultante da segunda atividade na fase de Desenvolvimento do processo de software, na empresa SWQuality Consultoria e Sistemas.

Como justificativa deste trabalho é apresentada a análise econômica de Boehm (1999). Baseado nas motivações da utilização de componentes, Boehm realiza uma análise

econômica de três estratégias para o aumento da produtividade de software: (1) trabalhar mais rapidamente, (2) trabalhar mais inteligentemente, (3) evitar trabalho desnecessário através do reuso. Os resultados desta pesquisa, para o Departamento de Defesa dos EUA, indicaram uma taxa de redução de custos de 8% adotando-se a estratégia (1); de 17% adotando-se a (2); e de 47%, ao ser adotada a estratégia (3). Ele sugere também que o projeto de componentes reutilizáveis de software é o meio mais eficaz para aumento de produtividade (produzir mais rapidamente, com menores custos).

1.2. Organização do Trabalho

Este trabalho está organizado da seguinte forma: o Capítulo 2 faz uma apresentação sobre componentes, o desenvolvimento de software baseado em componentes e sobre a Engenharia de Software baseada em Componentes, além de uma breve revisão dos trabalhos correlatos; no Capítulo 3 são apresentados processos, CMMI e o PEPP que é o processo que será alvo de modificação; a metodologia utilizada é apresentada no Capítulo 4; no Capítulo 5 são apresentados os trabalhos realizados, assim como partes do processo modificado e o Capítulo 6 apresenta as conclusões e os trabalhos futuros.

2. COMPONENTES, CBD E CBSE

Para uma melhor compreensão do trabalho, este capítulo apresenta conceitos fundamentais à utilização de componentes de software, que são: *Component-based Development* (CBD) e *Component-based Software Engineering* (CBSE), sendo estes itens muito importantes para o tema em estudo.

2.1. CBD

2.1.1. O que é CBD e porque usar CBD?

CBD é uma sigla para *Component Based Development* que significa desenvolvimento baseado em componentes. Duas idéias primordiais são os pilares do desenvolvimento baseado em componentes. Primeiramente, o desenvolvimento de uma aplicação pode ser melhorado significativamente se a aplicação pode ser rapidamente montada a partir de componentes de software pré-fabricados. A seguir, uma crescente coleção de componentes de software, que são capazes de interagir entre si, estará disponível em catálogos gerais e específicos. Colocadas juntas, estas duas idéias movem o desenvolvimento de aplicações de uma atividade artesanal, a qual foi se tornando cada vez mais enfraquecida nos seus 30 primeiros anos de existência, para um processo industrial, ajustado para encontrar negócios modernos, globais, altamente dinâmicos e competitivos (Short, 1997).

Sistemas de informação enfrentam vários problemas (Short, 1997). Entre eles podemos citar:

- Responder às requisições de implementar rapidamente e efetivamente aplicações que suportam novos *workflows* em processos de negócios reestruturados e com funções cruzadas;
- Forjar ligações entre pacotes de software, aplicações construídas nas próprias empresas e sistemas legados;
- Adaptar software de prateleira com os requisitos únicos corporativos.
- Encontrar caminhos para fazer aplicações de níveis departamentais trabalharem em maior escala e serem integradas em nível corporativo;
- Atualizar e melhorar aplicações existentes rápida e economicamente, sem sacrificar qualidade e usabilidade.

De acordo com Short (1997) estes problemas levam à crença de que o desenvolvimento baseado em componentes é “a resposta que anuncia um processo de desenvolvimento de aplicações reformado”. Podem-se dividir as vantagens do desenvolvimento baseado em componentes em duas categorias, baseadas e não baseadas em reuso. Entre as vantagens que não são baseadas em reuso podemos encontrar:

- Contenção de complexidade – projetos em desenvolvimento com foco em um ou em pequeno número de componentes tem seu escopo reduzido e riscos mais facilmente gerenciáveis. Equipes de projetos menores são geralmente mais produtivas;
- Oportunidade para massivo desenvolvimento em paralelo – fronteiras do projeto definidas em redor de estáveis definições de componentes encorajam desenvolvimento externo (*outsourcing*). *Outsourcing* de manutenção também pode ocorrer uma vez que quem publica os componentes podem revender e manter componentes desenvolvidos internamente;
- Estáveis definições de componentes também encorajam a flexibilidade – um componente que garante satisfazer um requisito pode ser substituído por outro que satisfaça aquele e alguns novos requisitos;
- Teste incremental – componentes facilitam teste de unidade e suportam a construção de testes progressivos;
- Componentes encapsulados agem como um *firewall* para mudanças – efeitos das mudanças são limitados e a manutenção se torna mais fácil e barata.

Dentre as vantagens que são baseadas em reuso encontramos:

- Redução do tempo de entrega – pela consulta de catálogos de componentes, algumas funcionalidades são normalmente encontradas. Componentes selecionados podem ser produzidos externamente;
- Redução de custos de desenvolvimento – o uso de componentes externos faz com que sejam reduzidos os custos de desenvolvimento uma vez que as equipes internas se focam em novas e únicas funcionalidades;
- Aumento de produtividade – equipes de desenvolvimento focam mais na montagem do sistema do que em desenvolvimento. Novos desenvolvedores podem ser mais produtivos mais cedo;
- Reduzido riscos de problemas – uma vez que pré-fabricados componentes são certificados e testados. Interações válidas são bem especificadas e documentadas;
- Maior consistência no reuso – componentes impõem uma arquitetura padrão para as aplicações;

- Seleção do melhor em um conjunto de soluções – o mercado de componentes se desenvolve e o fornecimento de pontos de funções empacotados como componentes cresce. Competição entre fornecedores faz com que o preço diminua.

2.1.2. Desenvolvimento Com e Para Reuso

Com a popularização do CBD, a necessidade de novos processos voltados para esse paradigma é uma realidade. Isso acontece porque os processos de desenvolvimento tradicionais não se adequam totalmente ao desenvolvimento de sistemas baseados em componentes. Mais especificamente, esses processos devem conter fases e métodos que ofereçam, entre outras coisas, técnicas que permitam o empacotamento de componentes com o objetivo específico de serem reutilizados.

É importante notar aqui que há questões distintas relacionadas ao uso de componentes no desenvolvimento de software (desenvolvimento com reuso) e ao desenvolvimento de componentes (desenvolvimento para reuso) propriamente ditos. As principais vantagens atribuídas à tecnologia de componentes são relativas ao desenvolvimento com reuso (ganho de produtividade, redução de *time to market*[1], etc.). O desenvolvimento do componente em si tende a ser mais complexo e demorado devido à preocupação extra com reuso (MCT, 2005).

2.2. Componentes

2.2.1. O que é um componente?

O que exatamente vem a ser um componente? Esta é uma questão difícil de responder de um modo fácil, mesmo que seja restrita a noção de componente às tecnologias atuais, porque a palavra componente é usada informalmente para significar uma variedade de coisas, cada uma com validade igual (Cheesman & Daniels, 2001).

Na literatura encontramos diferentes definições de componentes. Por exemplo, na visão geral técnica da Microsoft de *Component Object Model* (COM), o componente é definido como "um pedaço de software compilado, o qual está oferecendo um serviço", de acordo com Box, citado por Crnkovic & Larson (2002). D'Souza & Wills definem componentes como parte de software reusável a qual é desenvolvida independentemente e pode ser combinada com outros componentes para construir maiores unidades. Estas partes podem ser adaptadas, mas não modificadas, citados por Crnkovic & Larson (2002).

[1] *Time to market* é uma expressão que indica o tempo que um produto leva desde a sua concepção até o momento em que ele está disponível para venda.

Segundo o MCT (2005), uma analogia que pode auxiliar na compreensão sobre componentes de software é o brinquedo Lego (2006). As peças de Lego, clássico brinquedo de montar, podem simbolizar a idéia de componentes de software em um sentido mais geral. As peças do Lego permitem a montagem de automóveis, casas, aviões, navios e muitas outras mais, limitada apenas pela criatividade dos montadores. A montagem no Lego consiste em reunir diversas peças de diferentes formatos, encaixá-las entre si e obter, no final, uma figura totalmente nova. Existem algumas peças que podem e provavelmente serão utilizadas na construção de qualquer uma das montagens. As pequenas peças retangulares (aquelas que vêm em maior número) provavelmente estarão presentes em todas as montagens feitas com o brinquedo. Outras peças, por sua vez, têm funções mais específicas e serão utilizadas dentro de sua especialidade, dificilmente se adaptando a outros contextos. Um eixo, por exemplo, pode ser utilizado em automóveis, aviões, mas provavelmente não o será na montagem de casas. Nada impede, porém, que o montador visualize uma nova função para o eixo em um novo contexto e o utilize em uma montagem. Este exemplo, embora simples, mostra que componentes são reutilizáveis, podem possuir uso mais geral ou específico, embora todos sejam genéricos dentro do escopo considerado, e componentes se interagem com outros componentes.

O MCT (2005) apresenta as propriedades de um componente de software. A Tabela 2.1 exibe algumas destas propriedades.

Tabela 1 - Propriedades dos Componentes de Software

Propriedade	Descrição	Vantagem
Uso múltiplo	Utilizável em diversos projetos.	Compartilhamento do custo do desenvolvimento entre diversos projetos. Ampliação paulatina da confiança no componente, ao ser empregado em diversos projetos.
Sem contexto específico (genérico)	Desenvolvido independentemente de qualquer projeto específico ou contexto de sistema. No caso de necessidades específicas, estas devem estar explícitas.	Aumento da probabilidade do componente ser reutilizado
Composição	Pode ser combinado com outros componentes, com base nos contratos das interfaces.	Alta produtividade no desenvolvimento alcançada através de montagem de sistemas complexos a partir de componentes.
Encapsulamento	Apenas as interfaces são visíveis e a implementação não pode ser alterada.	Evita variações múltiplas; todos os usos de um componente se beneficiam de um mesmo esforço de manutenção e atualização.
Unidades independentes	Componentes podem ser configurados e instalados como unidades atômicas independentes e atualizados posteriormente independentemente do resto do sistema.	Permite que clientes alterem, atualizem e combinem componentes mesmo com o sistema em produção, aumentando a competição entre os fornecedores de componentes.

Fonte: MCT (2005)

São encontradas, comumente, nas definições de componentes as seguintes características (Crnkovic & Larson, 2002):

- Componentes são unidades de composição;
- Componentes precisam ser especificados de um modo que é possível sua composição com outros componentes;
- A composição entre componentes e sua integração em sistemas precisa ocorrer de um modo previsível.

Os princípios fundamentais para componentes são os mesmos da Orientação a Objetos (Cheesman & Daniels, 2001):

- Unificação de dados e função – um objeto de software consiste de dados (ou estados) e funções que processam estes dados;
- Encapsulamento – o cliente do objeto é isolado do modo em que os dados do objeto de software é armazenado ou como as funções são implementadas. O cliente depende da especificação e não da implementação. Esta separação é muito importante e a chave para gerenciar dependências e reduzir o acoplamento;
- Identidade – cada objeto de software possui a sua identidade independente do seu estado.

E o que difere o componente de um objeto de software é a explícita separação de sua especificação da sua implementação e a divisão da sua especificação em interfaces (Cheesman & Daniels, 2001).

2.2.2. Formas de um componente

Um componente pode aparecer em várias e diferentes formas (Cheesman & Daniels, 2001). São elas:

- Padrão de componente (*Component Standard*) – ambiente padrão que serve como um modelo onde os componentes se enquadram. Como exemplos de tais padrões, temos o EJB (*Enterprise Java Beans*) e o COM+ (*Component Object Model*) da Microsoft. Grandes organizações podem possuir definidos os seus próprios padrões;
- Especificação do componente (*Component Specification*) – é a especificação de uma unidade de software que descreve o comportamento de um conjunto de objetos do componente e define uma unidade de implementação;
- Interface do componente (*Component Interface*) – define o conjunto de comportamentos que podem ser oferecidos pelo componente;

- Implementação do componente (*Component Implementation*) – consiste na realização da especificação do componente. A implementação pode ser instalada e substituída independentemente de outros componentes. Não é necessariamente um único item físico como um único arquivo;
- Componente instalado (*Installed Component*) – representa uma cópia da implementação do componente registrada em um ambiente de execução. Este registro permite ao ambiente identificar os componentes instalados e criar uma nova instância do mesmo;
- Objeto do componente (*Component Object*) – instância de um componente instalado. É um conceito de tempo de execução. Um objeto com seus próprios dados e única identidade. Um componente instalado pode possuir muitos objetos do componente (o que requer uma identificação explícita) ou um único (cuja identificação pode ser implícita).

As formas que podem ser tomadas pelo componente são exibidas na Figura 2.1.

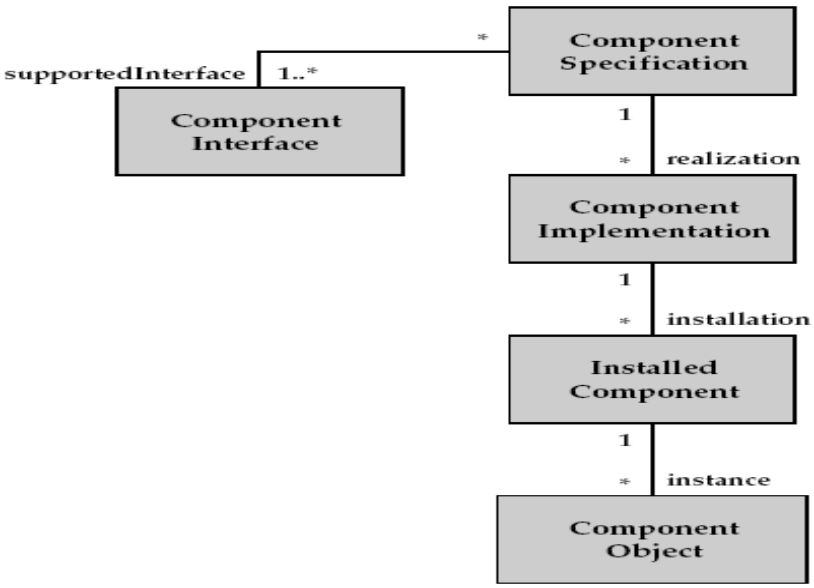

Figura 2.1 - Formas assumidas por um componente
Fonte: Cheesman & Daniels (2001)

2.2.3. Arquitetura de Componentes e Arquitetura de Sistemas

Chessman & Daniels (2001) definem arquitetura de sistemas como o conjunto de partes que compõe uma completa instalação de software, incluindo as responsabilidades destas partes, suas interconexões e mesmo a tecnologia apropriada; e a arquitetura de componentes como o conjunto

de componentes de software ao nível da aplicação, seus relacionamentos estruturais e suas dependências comportamentais.

Em seus estudos, Chessman & Daniels (2001) dividem a arquitetura de um sistema em quatro camadas: Interface com o Usuário, Diálogo com o Usuário, Serviços de Sistema e Serviços de Negócio. Uma breve descrição destas camadas é apresentada a seguir:

- Interface com o usuário – compreende a apresentação das informações ao usuário e captura suas entradas de dados. Sistemas externos podem ser utilizados também;
- Diálogo com o usuário – gerencia os diálogos do usuário em uma sessão;
- Serviços de sistema – representação externa do sistema, provendo acesso aos serviços do sistema. Esta camada funciona como uma fachada para os serviços providos pela camada inferior, provendo um contexto no qual serviços de negócio mais genéricos são utilizados de acordo com as necessidades de um sistema em particular;
- Serviços de negócio – implementação do núcleo das informações do negócio, regras e transformações. Os componentes desta camada são reutilizados em diferentes sistemas.

Na Figura 2.2 são apresentadas as camadas da arquitetura do sistema.

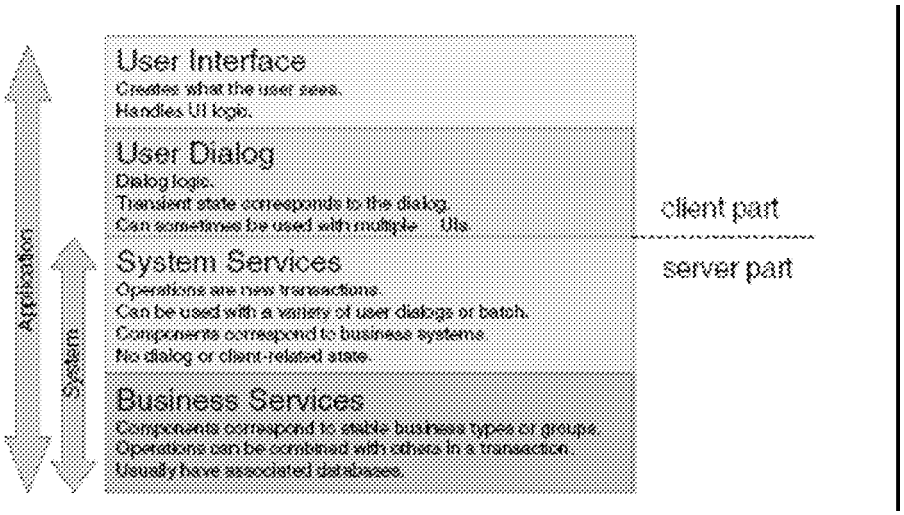

Figura 2.2 - Camadas da arquitetura do sistema
Fonte: Cheesman & Daniels (2001)

2.3. CBSE

Segundo Crnkovic & Larsson (2002), clientes e fornecedores têm esperado muito do desenvolvimento em componentes, mas suas expectativas não têm sido sempre alcançadas. A experiência tem mostrado que o desenvolvimento baseado em componentes requer uma abordagem

sistemática com foco nos aspectos de componentes no desenvolvimento de software. As disciplinas tradicionais da engenharia de software precisam ser ajustadas para a nova abordagem, e novos procedimentos precisam ser desenvolvidos. A engenharia de software baseada em componentes, do inglês *Component-Based Software Engineering (CBSE)*, tem sido reconhecida como uma nova subdisciplina da engenharia de software. As maiores metas da engenharia de software baseada em componentes são:

- Prover suporte ao desenvolvimento de sistemas como montagem de componentes;
- Auxiliar no desenvolvimento de componentes como entidades reusáveis;
- Facilitar a manutenção e atualização de sistemas pela customização e substituição de seus componentes.

A construção de sistemas a partir de componentes e a construção de componentes e a construção de componentes para diferentes sistemas requerem metodologias e processos estabelecidos não apenas em relação aos aspectos de desenvolvimento e manutenção, mas também ao inteiro ciclo de vida do componente e do sistema, incluindo aspectos organizacionais, de mercado e legal, entre outros. Em adição aos temas específicos da engenharia de software baseada em componentes como especificações de componentes e tecnologias, algumas disciplinas e processos da engenharia de software requerem metodologias específicas para aplicação em CBD. Muitas destas metodologias ainda não têm sido estabelecidas na prática, e algumas ainda não têm sido refinadas teoricamente. O progresso do desenvolvimento de software em um futuro próximo dependerá muito do estabelecimento com sucesso da CBSE, um ponto que é reconhecido pela indústria e academia (Crnkovic & Larson, 2002).

De acordo com Crnkovic & Larson (2002), CBD e CBSE estão iniciando suas fases de expansão. CBD é reconhecido como uma nova abordagem poderosa que melhorará significativamente – se não revolucionar – o uso e desenvolvimento de software em geral. Pode-se esperar que componentes e serviços baseados em componentes sejam amplamente utilizados por pessoas que não são programadores quando construindo suas aplicações. Ferramentas para construir tais aplicações pela montagem de componentes serão desenvolvidas. Atualização automática de componentes através da internet, já presente em muitas aplicações hoje, será o meio padrão de melhoria das aplicações.

Outra tendência observada é a padronização de componentes específicos de domínio no nível de interfaces. Isto facilita a construção de aplicações a partir de componentes adquiridos de diferentes vendedores. Crnokovic & Larson (2002) afirmam que a padronização de componentes específicos de domínios requer a padronização dos processos específicos de domínio. O suporte à troca de informações entre componentes, aplicações e sistemas distribuídos através da internet está sendo desenvolvido e tecnologias como o XML (*Extensible Markup Language*) são utilizadas para troca de informações através da internet.

Crnokovic & Larson (2002) também apresentam alguns dos desafios da CBSE. Tais desafios são: a especificação dos componentes, os modelos dos componentes, o ciclo de vida do software baseado em componentes, a previsibilidade da composição, componentes confiáveis (*trusted components*), certificação de componentes, configuração dos componentes, suporte de ferramentas e sistemas confiáveis (*dependable systems*).

Embora a especificação dos componentes tenha sido um problema endereçado desde o início do desenvolvimento de modelos de componentes, não se tem chegado a um consenso sobre como deve ser especificado um componente.

Dos modelos de componentes, pode-se dizer que mesmo que os modelos de desenvolvimento existentes demonstrem tecnologias poderosas, eles possuem muitas características ambíguas, são incompletos e difíceis de usar. As relações entre a arquitetura do sistema e os modelos de componentes não estão definidas precisamente.

O ciclo de vida de software baseado em componentes está se tornando mais complexo porque muitas fases são separadas em atividades não sincronizadas. Por exemplo, o desenvolvimento do componente pode ser totalmente independente do desenvolvimento do sistema que irá usar o componente. O processo de engenharia dos requisitos é muito mais complexo porque os possíveis componentes candidatos, normalmente, faltam uma ou mais características que o sistema requer.

Ainda relacionado ao ciclo de vida do software baseado em componentes, mesmo que alguns componentes estejam individualmente bem encaixados no sistema, não é óbvio que eles irão funcionar de modo ótimo em combinação com os outros no sistema. Estas restrições podem requerer uma outra abordagem na engenharia de requisitos – uma análise da praticidade dos requisitos em relação aos componentes disponíveis e da conseqüente modificação dos requisitos. Uma estratégia para gerenciar riscos na seleção e evolução dos componentes é necessária por causa das muitas incertezas que cercam os processos de seleção e evolução dos componentes.

Similarmente, muitas questões restam nas fases posteriores do ciclo de vida de sistemas baseados em componente. Desde que os sistemas baseados em componentes incluem componentes com diferentes ciclos de vida, o problema da evolução do sistema se torna significativamente mais complexo. Muitos tipos diferentes de problemas ainda não foram resolvidos. Existem questões técnicas (Pode o sistema ser tecnicamente atualizado pela substituição de componentes?), questões administrativas e organizacionais (Quais componentes podem ser atualizados, quais componentes poderiam e quais precisam ser atualizados?), questões legais (Quem é responsável pela falha do sistema? Quem produziu o sistema ou quem produziu os componentes?) e inúmeras outras. CBSE é uma abordagem nova e pequena experiência tem sido adquirida considerando-se a manutenção de tais sistemas.

Mesmo que seja assumido que todos os atributos relevantes do componente estejam especificados, não há necessariamente o conhecimento de que estes atributos sejam os correspondentes do sistema ao qual eles irão fazer parte. A abordagem ideal – derivar atributos do sistema a partir de atributos dos componentes – ainda é tema de pesquisa.

Desde que a tendência é entregar componentes na forma binária e o processo de desenvolvimento do componente está fora do controle do usuário do componente, questões relacionadas à confiabilidade do componente se tornam muito importantes. Um meio de classificação dos componentes é a certificação dos mesmos. Apesar da existência de uma crença comum de que certificação significa absoluta confiabilidade, de fato isto meramente provê os resultados de testes realizados e uma descrição do ambiente em que os testes foram realizados. Embora a certificação seja um procedimento padrão em muitos domínios, ela ainda não tem sido estabelecida em software em geral e especificamente não para componentes de software.

Sistemas complexos podem incluir muitos componentes, os quais, por sua vez, incluem outros componentes. Em muitos casos, composições de componentes são tratadas como componentes. Quando estruturas complexas começam a ser trabalhadas, problemas envolvendo configurações estruturais começam a aparecer. Por exemplo, duas composições podem possuir o mesmo componente. Este componente será tratado como duas entidades ou o sistema aceitará o componente como uma única instância, comum para as duas composições? O que ocorre se duas versões diferentes do componente são incorporadas nas duas composições? Qual versão será selecionada? O que ocorre se as duas versões não são compatíveis? Os problemas de atualização dinâmica de componentes já são conhecidos, mas suas soluções ainda são temas de pesquisas. Um meio de manusear estes tipos de sistemas complexos com muitos componentes é fazendo uso de arquiteturas de linhas de produto que impõem regras para a configuração de componentes.

O propósito da engenharia de software é prover soluções práticas para problemas práticos, e a existência de ferramentas apropriadas é essencial para o desempenho da CBSE. Ferramentas, como o *Visual Basic* da *Microsoft*, têm provado ser extremamente bem sucedidas, mas muitas outras ferramentas ainda têm que aparecer: ferramentas de seleção e avaliação de componentes, repositórios de componentes e ferramentas para gerência dos repositórios, ferramentas de teste de componente, ferramentas para projeto baseado em componentes, ferramentas de análise de sistemas em tempo de execução e ferramentas de configuração de componentes, entre outras. O objetivo da CBSE é construir sistemas de modo simples e eficiente, e isto pode ser alcançado unicamente com um extensivo suporte de ferramentas.

O uso de CBD em domínios de segurança ou críticos, sistemas de tempo real e diferentes sistemas de controle de processos, no qual requisitos de confiabilidade são particularmente rigorosos, é muito desafiante. Um grande problema com a CBD é a possibilidade limitada de

garantir a qualidade e outros atributos não funcionais dos componentes e, deste modo, uma incapacidade de garantir atributos específicos do sistema.

3. PROCESSOS, CMMI e PEPP

Este capítulo dá uma visão geral sobre processos, o modelo de maturidade *Capability Maturity Model Integration* (CMMI) e o Processo de Software para Empresas de Pequeno Porte baseado no modelo CMMI (PEPP).

3.1. Processos

O conceito de processo é quase que intuitivo. As engenharias comumente descrevem processos como sendo diversas operações pelas quais passa um produto até ele ficar pronto [Souza 2004].

Algumas definições de processo seguem abaixo:

- A ABNT (1994) define processo como um conjunto de atividades inter-relacionadas, que transforma entradas em saídas;
- O *Institute of Electrical and Electronic Engineers* (IEEE, 1990) define processo como uma seqüência de passos realizados para um determinado propósito.

Esta definição pode ser aplicada a qualquer atividade, seja ela da manufatura ou não.

Paulk (1995) define Processo de Software como um conjunto de atividades, métodos, práticas e tecnologias que as pessoas utilizam para desenvolver e manter software e seus produtos relacionados. A Figura 3.1 ilustra esta definição.

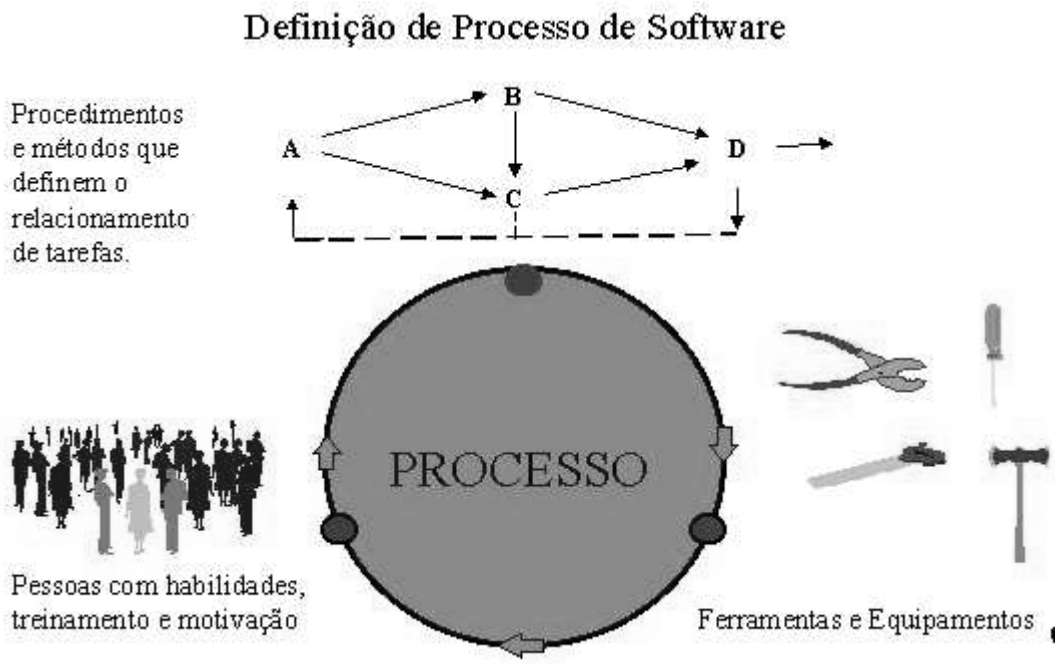

Figura 3.1 - Processo e seus componentes
Fonte: Aguiar (2004)

Ainda segundo Paulk (1995), considerando que o software é resultado do processo de desenvolvimento, espera-se que a sua qualidade seja altamente influenciada pela qualidade do processo que o gera.

Focando-se somente no produto deixam-se de lado assuntos relacionados como a escalabilidade, ou seja, o aumento do tamanho da equipe leva ao risco de se perder qualidade. Além disso, não há a preocupação de como realizar melhor as tarefas.

Focando-se no processo, prevêem-se a repetição de resultados, tendências futuras para os projetos, mantendo as características do produto. Um processo bem controlado evita surpresas e minimiza os riscos (Aguiar, 2004).

3.2. O Modelo CMMI

A sigla CMMI representa as iniciais de *Capability Maturity Model Integration* e nomeia tanto um projeto, quanto os modelos resultantes deste projeto. O Projeto CMMI, que pode ser traduzido como Projeto de Integração dos Modelos de Maturidade e Capacidade, está sendo executado pelo *Software Engineering Institute* (SEI), em cooperação com a indústria e governo. O objetivo é consolidar um *framework* para modelos, evoluir e integrar modelos desenvolvidos pelo SEI (inicialmente os modelos SW-CMM, SE-CMM e IPD-CMM), e gerar seus produtos associados, incluindo material de treinamento e método de avaliação. Estes são os três modelos que foram evoluídos e integrados inicialmente: a versão 2.0 do *Capability Maturity Model for Software* (SW-CMM), o *System Engineering Capability Maturity Model*: EIA 731 (SE-CMM) e o *Integrated Product Development Capability Maturity Model* (IPD-CMM).

Segundo Salviano (2003), esta integração e evolução tiveram como objetivo principal a redução do custo da implementação de melhoria de processo multidisciplinar baseada em modelos. Multidisciplinar porque, além da engenharia de software, o CMMI cobre também a engenharia de sistemas, aquisição e a cadeia de desenvolvimento de produto. Esta redução de custo é obtida por meio da eliminação de inconsistências; redução de duplicações, melhoria da clareza e entendimento, utilização de terminologia comum, utilização de estilo consistente, estabelecimento de regras de construção uniformes, manutenção de componentes comuns, garantia da consistência com a Norma ISO/IEC 15504.

O nome do modelo CMMI pode ser traduzido para “Modelo Integrado de Maturidade da Capacidade”. Em agosto de 2000 foi lançada a versão 1.0 do CMMI, também chamada de CMMI-SE/SW v1 [SEI 2004]. Após outras versões intermediárias, em 10 de março de 2002 foi lançada a versão atual denominada de CMMI-SE/SW/IPPD/SS Version 1.1 (*CMMI^{SM} for Systems Engineering / Software Engineering / Integrated Product and Process Development / Supplier*

Sourcing, Versão 1.1, em português CMMI[SM] para Engenharia de Sistemas / Engenharia de Software / Desenvolvimento Integrado de Produtos e Processos / Fornecimento).

3.2.1. Arquitetura do Modelo

A arquitetura do CMMI é composta basicamente pela definição de um conjunto de áreas de processo, organizadas em duas representações diferentes: uma como um modelo por estágio, semelhante ao SW-CMM, e outra como um modelo contínuo (semelhante à ISO/IEC 15504). A versão atual é composta por 25 áreas de processo, conforme ilustrado na Figura 3.2.

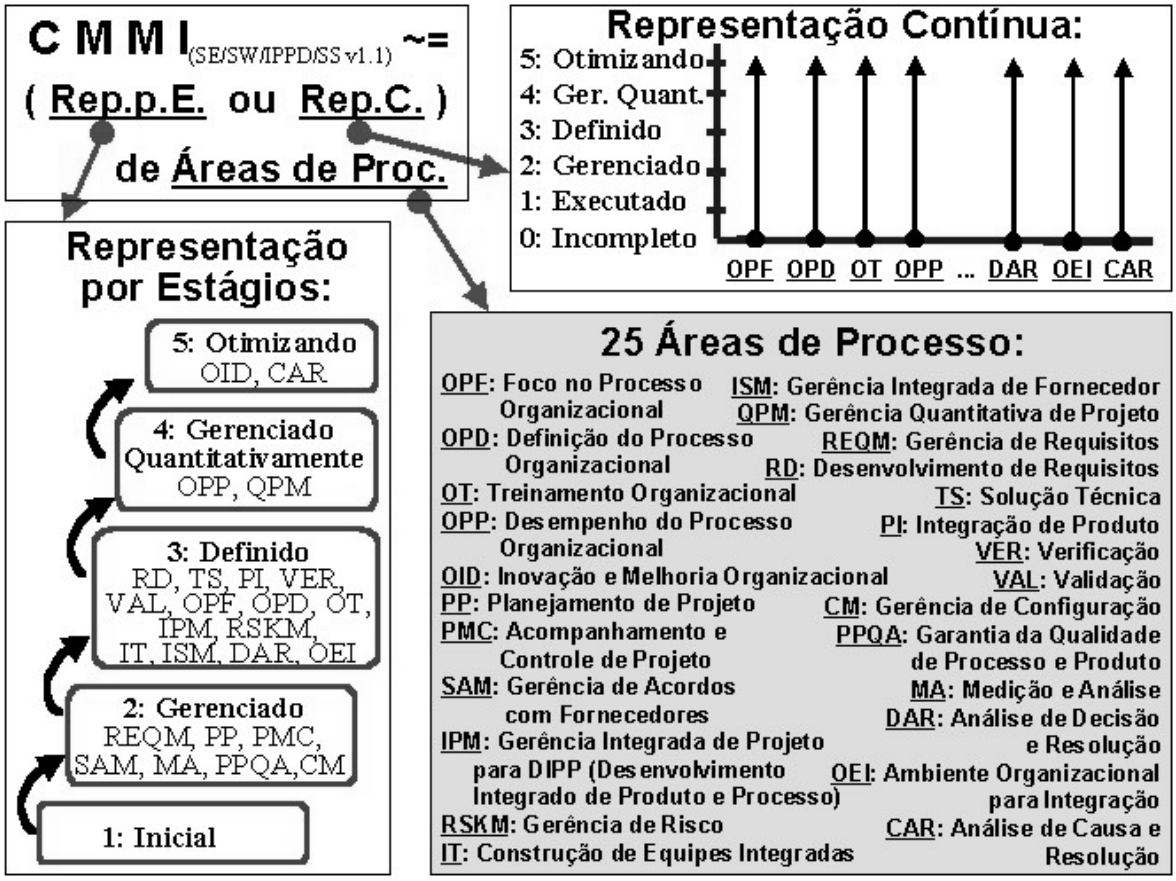

Figura 3.2 - CMMI: Áreas de Processo em duas Representações: por Estágio e Contínua
Fonte: Aguiar (2004)

Cada área de processo é definida no modelo por meio da descrição de seu propósito, notas introdutórias, relacionamentos com outras áreas, metas específicas, metas genéricas, práticas e subpráticas específicas, práticas e subpráticas genéricas, produtos de trabalho típicos e referências para outros elementos do modelo relacionados. A descrição de cada área ocupa cerca de 20 páginas.

Na representação por estágio, as 25 áreas de processo estão agrupadas em 4 níveis de maturidade: níveis 2, 3, 4 e 5. O nível 1 não contém nenhuma área de processo e a única exigência para que a empresa de software seja qualificada neste nível é a sua própria existência. Em relação a

esta representação, o processo, de desenvolvimento e manutenção, de software ou sistema de uma unidade organizacional, pode estar classificado em um dos seguintes cinco níveis de maturidade:

- Nível 1: Inicial (*Initial*)
- Nível 2: Gerenciado (*Managed*)
- Nível 3: Definido (*Defined*)
- Nível 4: Gerenciado Quantitativamente (*Quantitatively Managed*)
- Nível 5: Otimizando (*Optimizing*)

Cada nível de maturidade é definido basicamente pelo conjunto de áreas de processo do nível.

Na representação contínua, as mesmas 25 áreas de processo estão agrupadas em quatro grupos (gerência de processos, gerência de projetos, engenharia e suporte) e são definidos seis níveis de capacidade de processo. Nesta representação, o conjunto de atividades correspondente a cada uma destas áreas de processo, pode ter sua capacidade de execução classificada em um dos seguintes seis níveis de capacidade de processo:

- Nível 0: Incompleto (*Incomplete*)
- Nível 1: Executado (*Performed*)
- Nível 2: Gerenciado (*Managed*)
- Nível 3: Definido (*Defined*)
- Nível 4: Gerenciado Quantitativamente (*Quantitatively Managed*)
- Nível 5: Otimizando (*Optimizing*)

Cada nível de capacidade é definido por um conjunto de características que o processo deve satisfazer para estar naquele nível.

A descrição de cada área de processo ocupa cerca de 20 páginas do modelo de referência do CMMI. A área de Planejamento de Projetos, por exemplo, contém 8 metas, 8 práticas, 31 subpráticas e ocupa 29 páginas.

Em relação à disciplina de gerência de projeto, o modelo CMMI apresenta um conjunto compreensivo de áreas de processo para orientar a implantação e evolução das atividades desta gerência. O CMMI evoluiu e ampliou as orientações para gerência de projeto do SW-CMM e dos outros modelos precursores. A representação como um modelo contínuo, consistente com a Norma ISO/IEC 15504, permite uma maior flexibilidade na utilização das áreas de processo, facilitando a obtenção de melhorias adicionais.

Como o objetivo do CMMI é representar no modelo metas e recomendações genéricas para orientar a melhoria dos processos em geral, não são descritas soluções prontas para serem executadas nas organizações. Cabe a cada organização entender e interpretar estas orientações no contexto, objetivo e estratégia de negócio da organização para obter melhorias relevantes.

3.2.2. Disciplinas do Modelo

Segundo Salviano (2003), o objetivo do CMMI é ser um *framework* extensível, para poder acrescentar novas disciplinas. Cada modelo do CMMI inclui uma ou mais disciplinas, de forma integrada. Atualmente os modelos do CMMI cobrem 4 disciplinas:

- Engenharia de Sistemas (*System Engineering* - SE);
- Engenharia de Software (*Software Engineering* - SW);
- Desenvolvimento de produtos e processos integrados (*Integrated Product and Process Development* - IPPD);
- Aquisição (*Supplier Sourcing* - SS).

Conforme Aguiar (2004), quatro modelos estavam disponíveis na versão 1.1, e todos nas representações por estágio e contínua. Estes modelos estão relacionados a seguir, identificados pelas siglas das disciplinas tratadas:

- CMMI-SE/SW/IPPD/SS;
- CMMI-SE/SW/IPPD;
- CMMI-SE/SW ;
- CMMI-SW.

Conforme resumido na Figura 3.2, apresentada anteriormente, a base dos modelos CMMI são as áreas de processo, que estão organizadas em níveis de maturidade (na representação por estágio) e em categorias de processo (na representação contínua).

Na representação por estágio, as áreas de processo são organizadas em quatro níveis de maturidade. A seguir estão relacionadas as áreas de processo em cada nível de maturidade. Cada área de processo é identificada por uma sigla (baseada no nome em inglês), uma tradução do nome para o português e o nome original em inglês:

Nível 2:

- REQM: Gestão de Requisitos (*Requirements Management*);
- PP: Planejamento de Projeto (*Project Planning*);
- PMC: Acompanhamento e Controle de Projeto (*Project Monitoring and Control*);
- SAM: Gestão de Acordos com Fornecedores (*Supplier Agreement Management*);
- CM: Gestão de Configuração (*Configuration Management*);
- PPQA: Garantia da Qualidade de Processo e Produto (*Process and Product Quality Assurance*);
- MA: Medição e Análise (*Measurement and Analysis*).

Nível 3:

- RD: Desenvolvimento de Requisitos (*Requirements Development*);
- TS: Solução Técnica (*Technical Solution*);
- PI: Integração de Produto (*Product Integration*);
- VER: Verificação (*Verification*);
- VAL: Validação (*Validation*);
- OPF: Foco no Processo Organizacional (*Organizational Process Focus*);
- OPD: Definição do Processo Organizacional (*Organizational Process Definition*);
- OT: Treinamento Organizacional (*Organizational Training*);
- IPM: Gestão Integrada de Projeto (*Integrated Project Management*);
- RSKM: Gestão de Riscos (*Risk Management*);
- DAR: Análise de Decisão e Resolução (*Decision Analysis and Resolution*).

Nível 4:

- QPM: Gestão Quantitativa de Projeto (*Quantitative Project Management*);
- OPP: Desempenho do Processo Organizacional (*Organizational Process Performance*).

Nível 5:

- CAR: Análise de Causa e Resolução (*Causal Analysis and Resolution*);
- OID: Inovação e Melhoria Organizacional (*Organizational Innovation and Deployment*).

Na representação contínua, as áreas de processo são organizadas em quatro categorias de processo. A seguir estão relacionadas as áreas de processo em cada categoria de processo. Cada área de processo é identificada por uma sigla (baseada no nome em inglês), uma tradução do nome para o português e o nome original em inglês:

Categoria Gestão de Projeto:

- PP: Planejamento de Projeto (*Project Planning*);
- PMC: Acompanhamento e Controle de Projeto (*Project Monitoring and Control*);
- SAM: Gestão de Acordos com Fornecedores (*Supplier Agreement Management*);
- IPM: Gestão Integrada de Projeto (*Integrated Project Management*);
- RSKM: Gestão de Riscos (*Risk Management*);

- QPM: Gestão Quantitativa de Projeto (*Quantitative Project Management*).

Categoria Gestão de Processo:

- OPF: Foco no Processo Organizacional (*Organizational Process Focus*);
- OPD: Definição do Processo Organizacional (*Organizational Process Definition*);
- OT: Treinamento Organizacional (*Organizational Training*);
- OPP: Desempenho do Processo Organizacional (*Organizational Process Performance*);
- OID: Inovação e Melhoria Organizacional (*Organizational Innovation and Deployment*).

Categoria Engenharia:

- REQM: Gestão de Requisitos (*Requirements Management*);
- RD: Desenvolvimento de Requisitos (*Requirements Development*);
- TS: Solução Técnica (*Technical Solution*);
- PI: Integração de Produto (*Product Integration*);
- VER: Verificação (*Verification*);
- VAL: Validação (*Validation*).

Categoria Apoio:

- CM: Gestão de Configuração (*Configuration Management*);
- PPQA: Garantia da Qualidade de Processo e Produto (*Process and Product Quality Assurance*);
- MA: Medição e Análise (*Measurement and Analysis*);
- CAR: Análise de Causa e Resolução (*Causal Analysis and Resolution*);
- DAR: Análise de Decisão e Resolução (*Decision Analysis and Resolution*).

Em relação ao SW-CMM, o CMMI por estágio é uma revisão deste modelo, com ajustes. No nível 2 de maturidade, por exemplo, foi incluída a área de processo de medição e análise como uma ampliação deste assunto, que já era coberto em parte em cada área do SW-CMM. No nível 3, por exemplo, a área de engenharia de produto do SW-CMM foi mais bem descrita por meio de cinco áreas de processo: Desenvolvimento de Requisitos, Solução Técnica, Integração de Produto, Verificação, e Validação. Outras mudanças ocorrem para refletir melhor a orientação para atendimento dos níveis de maturidade (Salviano, 2003).

Em relação a utilização para melhoria de processo, o CMMI por estágio sugere abordagens semelhantes àquelas utilizadas com sucesso com o SW-CMM. Em relação ao CMMI contínuo, a tendência é toda a experiência de utilização da ISO/IEC 15504 ser aproveitada para ajustar as abordagens já utilizadas com sucesso pelo SW-CMM.

3.2.3. Elementos do Modelo

Os elementos que compõe o modelo CMMI são agrupados em três categorias que refletem como eles devem ser interpretados:

- Elementos Requeridos: metas específicas e genéricas são elementos requeridos do modelo. Estes elementos devem ser estabelecidos pelos processos definidos e implementados pela organização. São essenciais para avaliar a satisfação de uma área de processo. O estabelecimento (ou satisfação) das metas é usado em avaliações como base para satisfação da área de processo e maturidade organizacional. Apenas a declaração das metas é um elemento requerido do modelo, o título e qualquer nota associada são considerados elementos informativos do modelo;
- Elementos Esperados: práticas específicas e as genéricas são elementos esperados no modelo. Estes elementos descrevem o que uma organização tipicamente irá implementar para satisfazer um componente requerido. Servem como guia para aqueles que implementam melhorias e/ou executam avaliações. Espera-se que as práticas descritas, ou as alternativas aceitáveis para elas, estejam presentes nos processos planejados e implementados da organização, antes que as metas sejam consideradas satisfeitas. Apenas a declaração da prática é um elemento esperado do modelo, o título e qualquer nota associada são considerados elementos informativos do modelo;
- Componentes Informativos: subpráticas, produtos de trabalho típicos, particularidades da disciplina, elaborações de práticas genéricas, títulos, notas e referências são elementos informativos do modelo. Estes elementos ajudam o usuário do modelo a entender as metas e práticas e como elas podem ser estabelecidas, fornecendo detalhes para ajudar a pensar como estabelecer as práticas e metas.

Características comuns são elementos não classificados do modelo, apenas constituem um grupo que fornece uma maneira de apresentar as práticas genéricas.

Quando se usa um modelo CMMI como guia, processos são planejados e implementados em conformidade com os elementos esperados e requeridos das áreas de processo. Conformidade

com uma área de processo significa que nos processos planejados e implementados existe um processo associado (ou processos) que endereça as práticas específicas e genéricas da área de processo, ou alternativas, que geram um resultado de acordo com a meta associada com aquela prática específica ou genérica.

3.3. PEPP

O Processo de Software para Empresas de Pequeno Porte - PEPP, baseado no modelo CMMI, foi descrito por Aguiar (2004).

O PEPP foi criado baseando-se na realidade da empresa SWQuality Consultoria e Sistemas. Esta empresa está situada na cidade de Lavras-MG. Segundo Aguiar (2004), a empresa é caracterizada como pequena empresa, devido ao seu número de funcionários.

A SWQuality possui três linhas de negócio:

- Fábrica de Software: desenvolve projetos e produtos sob encomenda para clientes (inclusive para outras empresas de software);
- Ensino a Distância: editora e elabora conteúdo para cursos a distância. Administra e customiza ambiente virtual de ensino;
- Consultoria: avalia empresas de software e implanta modelos de processo (CMMI, MPS.BR, ISO/IEC 15504);

Na construção do processo, foi utilizada a ferramenta Microsoft Office Visio (2006) para o desenho do processo. Esta ferramenta foi escolhida pela facilidade de uso e grande quantidade de recursos oferecidos, dentre eles o recurso para publicação no formato de páginas para internet.

3.3.1. Modelo de Ciclo de Vida de Projetos do PPEP

O primeiro passo para a criação do processo foi a definição do Modelo de Ciclo de Vida de Projetos (Aguiar, 2004). Ele ficou divido em quatro fases. Estas fases foram definidas baseando-se na realidade da empresa.

Para cada fase foram definidas quais atividades seriam necessárias para que os produtos de trabalho fossem gerados. Para cada atividade, era atribuído um, ou mais, responsável pela execução da mesma. Normalmente, para cada atividade são atribuídos os artefatos de entrada e de saída, podendo existir atividades que não possuam artefatos. Os artefatos de entrada podem ser *templates* de documentos, documentos preenchidos, padrões do processo, elementos informativos. Os artefatos de saída podem ser um, ou mais, artefatos de entrada atualizados ou preenchidos. As quatro fases do modelo de ciclo de vidas de projeto são exibidas na Figura 3.3.

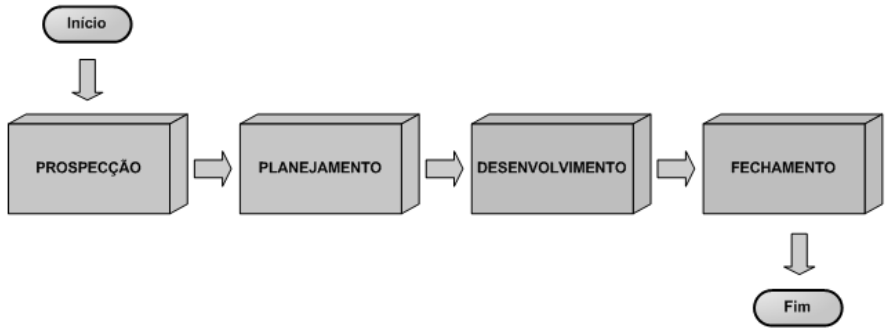

Figura 3.3 - Modelo de Ciclo de Vidas de Projetos de acordo com o PEPP
Fonte: Aguiar (2004)

3.3.1.1. Fase de Prospecção do PEPP

Esta fase é caracterizada pelo atendimento de uma solicitação de projeto feita pelo cliente. Após a definição do escopo do projeto, é elaborada uma proposta para o cliente. Nesta proposta, estará delimitado o escopo, o cronograma de execução, o valor e as restrições contratuais do projeto. A aprovação desta proposta pelo cliente implica na mudança para a fase seguinte - Fase de Planejamento.

3.3.1.2. Fase de Planejamento do PEPP

Esta fase é caracterizada pelo refinamento do escopo, acordado na proposta, e pelo planejamento de como o projeto será executado.

Na Fase de Planejamento, normalmente, o escopo do projeto não está muito detalhado, o que pode gerar desgastes ou até prejuízos entre as partes envolvidas no momento de homologação do projeto. Para minimizar os riscos, deve-se detalhar o escopo num nível atômico, abrangendo uma quantidade de detalhes que seja de segurança para empresa, e após esse detalhamento, obter o aval do cliente com esta nova versão do escopo.

Outro fator preponderante para a entrega do projeto de acordo com o estabelecido na proposta é a elaboração de um plano de projeto. Este plano contém as diretrizes que todo projeto dentro da empresa deve seguir, além dos específicos do projeto. Os passos para construção deste plano são: estimar o esforço, o custo, identificar os riscos e os recursos da empresa e do cliente que serão necessários para a execução do projeto.

Para finalizar o plano devem ser definidas as iterações do projeto. Após a aprovação do plano pela alta gerência deve-se obter o compromisso do cliente com plano. Esta ação faz com que o cliente esteja ciente do que foi planejado, assumindo as responsabilidades a ele atribuídas, e também aproxima do desenvolvimento e dos possíveis problemas que possam acontecer.

3.3.1.3. Fase de Desenvolvimento do PEPP

A Fase de Desenvolvimento de um projeto é a fase na qual as atividades definidas na fase de Planejamento são executadas.

O primeiro passo desta fase é definir a arquitetura do projeto. Em seguida são elaborados o projeto conceitual, o lógico e o físico. Após estas atividades, são produzidos os códigos fontes e os casos de teste. Para a conclusão da fase, o código é testado e, em seguida, homologado no ambiente definido pelo cliente, podendo também ser homologado no ambiente de desenvolvimento da empresa.

3.3.1.4. Fase de Fechamento do PEPP

Na fase do Fechamento, após a homologação do projeto na Fase de Desenvolvimento, o projeto é implantado em um projeto piloto e, posteriormente, no ambiente de produção do cliente. Também são planejadas as atividades de garantia e manutenção, caso existam.

3.3.2. Institucionalização do Processo

Para facilitar a institucionalização e o uso do processo na empresa, foi criada uma versão navegável do processo.

A versão navegável foi gerada a partir da ferramenta Visio, através da funcionalidade disponível para exportar o processo para uma forma navegável pela internet. Através da versão navegável é possível visualizar todas as fases, etapas e atividades do processo, bastando estar conectado à internet. É possível também ter acesso a todos os *templates* utilizados pelo processo.

4. METODOLOGIA

As pesquisas têm se tornado um instrumento amplamente utilizado e reconhecido na maior parte dos países do mundo. Esta seção apresentará o esclarecimento do tipo de pesquisa utilizada para a fundamentação de toda teoria anteriormente descrita e a explicação de como o trabalho foi realizado.

4.1. Tipo de Pesquisa

A classificação dos tipos de pesquisas varia de acordo com o enfoque dado, segundo interesses, condições, campos, objetivos, etc. Cabe ao pesquisador a escolha do método que melhor atender as suas necessidades.

O método de pesquisa utilizado neste trabalho é de natureza exploratória, que é a adaptação de uma fase de um processo de software, fundamentado em pesquisa bibliográfica e documental.

A pesquisa exploratória visa o aprimoramento de idéias ou a descoberta de intuições, ou seja, prover o pesquisador de um maior conhecimento sobre o tema ou problema de pesquisa em questão. Este tipo de pesquisa proporciona maior familiaridade com o problema, tornando-o mais explícito.

Segundo Gil (1991), o estudo de caso é caracterizado pelo estudo profundo e exaustivo de um ou de poucos objetos, de maneira que permita o seu amplo e detalhado conhecimento.

4.2. Procedimentos Metodológicos

Primeiramente, realizou-se uma revisão bibliográfica sobre componentes, engenharia de software baseada em componentes, CMMI, processos e desenvolvimento baseado em componentes. Foram consultados teses, dissertações, artigos científicos e a literatura de um modo geral.

A partir do estudo da literatura, foram identificadas as áreas de processo do CMMI que mais se relacionavam com o desenvolvimento baseado em componentes com reuso. As áreas de processo que foram identificadas como mais significativas ao desenvolvimento baseado em componentes são *Requirement Development*, *Product Integration* e *Technical Solution*. Foi então realizada uma análise envolvendo essas três áreas de processos do CMMI e foram identificadas as atividades e produtos de trabalhos relacionados com produção, uso e integração de componentes.

Utilizando-se os resultados da análise das três áreas de processos do CMMI foi feita uma análise no PEPP e inseridas as novas práticas identificadas no mesmo.

5. RESULTADOS E DISCUSSÃO

Neste capítulo é apresentada a análise realizada nas áreas de processos RD (*Requirement Development*), TS (*Technical Solution*) e PI (*Product Integration*). É exibida também de modo geral, a fase do ciclo de vida do projeto Desenvolvimento do PEPP e é apresentado parte das modificações realizadas no processo. Por motivos organizacionais e comerciais, a fase do ciclo de vida do projeto Desenvolvimento do PEPP e as modificações realizadas no processo serão abordadas de forma a assegurar propriedade intelectual do PEPP.

A parte teórica referente às áreas de processo do CMMI foi retirada de Chrissis et al. (2004).

5.1. A análise das áreas de processo do CMMI

Como dito anteriormente, a partir do estudo sobre componentes e o CMMI foram identificadas três áreas de processo que mais se relacionam com o desenvolvimento para reuso. As áreas de processo que foram classificadas como relevantes são: RD (*Requirement Development*), TS (*Technical Solution*) e PI (*Product Integration*). A seguir será dada uma breve descrição de cada uma destas áreas de processo e o que foi identificado em cada uma delas, como sendo relevante ao desenvolvimento baseado em componentes com reuso.

5.1.1. Requirement Development

A área de processo *Requirement Development* (RD) ou desenvolvimento de requisitos tem como objetivo produzir e analisar requisitos de clientes, produtos e componentes de produtos. Esta área de processo descreve três tipos de requisitos: requisitos de clientes, requisitos de produtos e requisitos de componentes de produtos. Juntos, estes requisitos tratam as necessidades dos *stakeholders* relevantes, incluindo aquelas pertinentes às diversas fases do ciclo de vida do produto (por exemplo, critérios de testes de aceitação) e atributos do produto (por exemplo, segurança, confiabilidade e possibilidade de manutenção). Os requisitos também tratam as restrições causadas pela seleção de soluções de *design* (por exemplo, integração de produtos comerciais de prateleira).

Foi identificado como relevante dentro desta área de processo a prática específica (*Specific Practice*) Alocar Requisitos de Componentes de Produtos (SP 2.2-1 *Allocate Product-Component Requirements*), que se encontra na meta específica (*Specific Goal)* Desenvolver Requisitos do Produto (*SG 2 Develop Product Requirements*).

5.1.2. *Technical Solution*

A área de processo *Technical Solution* (TS) ou Solução Técnica tem como objetivo criar o *design*, desenvolver e implementar soluções para os requisitos. Soluções, *designs* e implementações englobam produtos, componentes de produtos e processos relacionados com o ciclo de vida do produto, isoladamente ou combinados, conforme apropriado.

Foram identificadas como relevantes ao desenvolvimento com reuso as seguintes práticas específicas da TS:

- SP 1.1-1 *Develop Alternative Solutions and Selection Criteria;*
- SP 1.1-2 *Develop Detailed Alternative Solutions and Selection Criteria;*
- SP 1.3-1 *Select Product-Component Solutions;*
- SP 2.1-1 *Design the Product or Product Component;*
- SP 2.3-1 *Establish Interface Descriptions;*
- SP 2.3-3 *Design Interfaces Using Criteria;*
- SP 2.4-3 *Perform Make, Buy, or Reuse Analyses.*

5.1.3. *Product Integration*

A área de processo *Product Integration* (PI), ou Integração do Produto, tem como objetivos: montar o produto a partir dos componentes do produto, garantindo que o produto, quando integrado, funcione apropriadamente; e realizar a entrega do produto.

Foram identificadas como relevantes ao desenvolvimento com reuso as seguintes práticas específicas da PI:

- SP 1.1-1 *Determine Integration Sequence;*
- SP 1.3-3 *Establish Product Integration Procedures and Criteria;*
- SP 2.1-1 *Review Interface Descriptions for Completeness;*
- SP 2.2-1 *Manage Interfaces;*
- SP 3.1-1 *Confirm Readiness of Product Components for Integration;*
- SP 3.2-1 *Assemble Product Components;*
- SP 3.3-1 *Evaluate Assembled Product Components.*

5.2. A fase de Desenvolvimento Original do PEPP

As atividades definidas na fase de Planejamento são executadas na fase de Desenvolvimento de um projeto. Esta fase segue a de Planejamento e antecede a de Fechamento.

A arquitetura do projeto é definida na fase de Desenvolvimento do projeto. Seguem as seguintes atividades: elaboração do projeto conceitual, do lógico e do físico, seguido pela produção dos códigos fontes e casos de teste. Esta fase é concluída com a realização de testes no código e homologação no ambiente definido pelo cliente, podendo também esta homologação ocorrer no ambiente de desenvolvimento da empresa.

A fase de Desenvolvimento do PEPP é exibida na Figura 5.1.

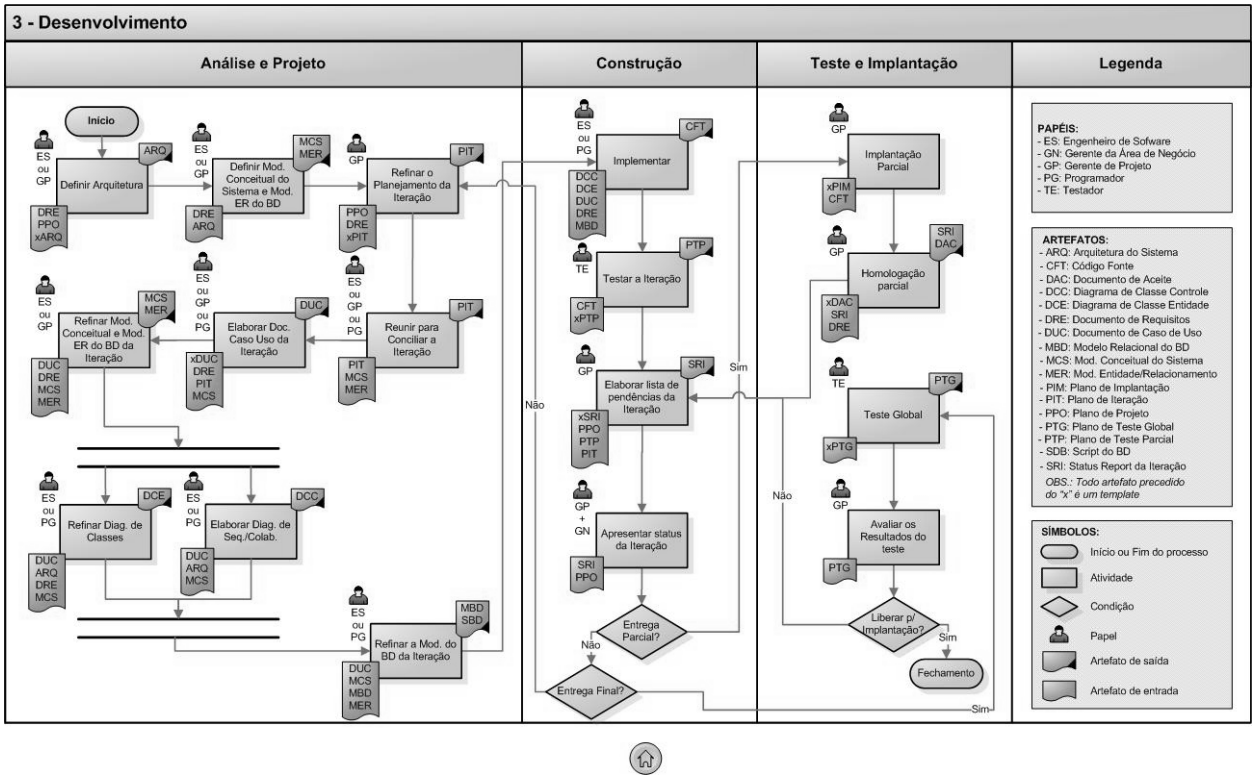

Figura 5.1 - Fase de Desenvolvimento Original do PEPP
Fonte: Elaborada pelo autor

A Figura 5.1 exibe a fase de Desenvolvimento do PEPP na sua forma original, antes das modificações realizadas neste trabalho.

A fase de Desenvolvimento é divida em três etapas que são:

- Análise e Projeto – são realizadas aqui as atividades relacionadas com a análise e projeto (*design*) do sistema;
- Construção – onde é realizada a implementação do sistema;
- Testes e Implantação – são efetuados aqui os testes do sistema assim como a implantação do mesmo.

Os papéis que desenvolvem as atividades exibidas nesta fase são:

- ES – Engenheiro de Software;
- GN - Gerente da Área de Negócio;
- GP – Gerente de Projeto;

- PG – Programador;
- TE – Testador.

5.3. A fase de Desenvolvimento Atualizada do PEPP

Após a adaptação do processo, surgiu a necessidade da criação de uma nova etapa, que é a de Integração. Nesta etapa são realizadas as atividades relacionadas com a integração do produto.

Na adaptação, surgiu a necessidade de dois novos papéis que são:

- EC – Engenheiro de Componentes, este papel é representado por um indivíduo com conhecimento dos componentes disponíveis e possuindo também conhecimentos sobre os serviços prestados e requisitos de cada um dos mesmos, assim como conhecimento sobre os domínios de aplicações desenvolvidas pela organização. Este indivíduo também é responsável pela avaliação de componentes disponíveis a partir de fontes externas e sobre critérios de seleção entre diferentes componentes;
- IC – Integrador de Componentes, papel representado por um indivíduo com conhecimento sobre as atividades desenvolvidas na integração dos componentes de produto.

Na adaptação foram criadas diversas novas atividades. A primeira atividade identificada foi a alocação de requisitos a componentes de produto. Esta atividade é exibida na Tabela 2.

Tabela 2 - Atividade – Alocar Requisitos de Componentes de Produtos

<table>
<tr><td colspan="2">Atividade - Alocar Requisitos de Componentes de Produtos</td></tr>
<tr><td colspan="2">Finalidade:
⇒ Fazer um mapeamento entre requisitos e componentes e documentar este relacionamento.</td></tr>
<tr><td>Artefatos de Entrada:
⇒ Documento de Requisitos
⇒ Documento de Caso de Usos
⇒ Template da Planilha de Alocação de Requisitos</td><td>Artefatos de Saída:
⇒ Planilha de Alocação de Requisitos</td></tr>
<tr><td colspan="2">Papel:
⇒ Gerente da Área de Negócio
⇒ Engenheiro de Componentes</td></tr>
<tr><td colspan="2">Ferramentas:
⇒ Planilha Eletrônica</td></tr>
<tr><td colspan="2">Passos:
1. Analisar os requisitos e casos de uso da iteração
2. Realizar o relacionamento entre componentes e casos de uso</td></tr>
</table>

Fonte: Elaborada pelo autor

Foi criada também uma outra atividade que é responsável pela identificação de restrições do *design* através do relacionamento entre componentes e requisitos. O detalhamento desta atividade é exibido na Tabela 3.

Tabela 3 - Atividade – Identificar e Alocar Restrições de *Design* a Componentes de Produto

<table>
<tr><th colspan="2">Atividade – Identificar e Alocar Restrições de Design a Componentes de Produto</th></tr>
<tr><td colspan="2">Finalidade:
⇒ Identificar as restrições de design através do relacionamento entre componentes e requisitos.
⇒ Alocar estas restrições a componentes de produto</td></tr>
<tr><td>Artefatos de Entrada:
⇒ Documento de Caso de Uso
⇒ Planilha de Alocação de Requisitos
⇒ Arquitetura do Sistema
⇒ Modelo Conceitual do Sistema</td><td>Artefatos de Saída:
⇒ Documento de Restrições de Design</td></tr>
<tr><td colspan="2">Papel:
⇒ Engenheiro de Software
⇒ Engenheiro de Componentes
⇒ Integrador de Componentes</td></tr>
<tr><td colspan="2">Ferramentas:
⇒ Planilha Eletrônica</td></tr>
<tr><td colspan="2">Passos:
1. Analisar a alocação de requisitos
2. Analisar restrições entre alocação de requisitos a componentes e arquitetura do sistema
3. Atualizar Planilha de Alocação de Requisitos</td></tr>
</table>

Fonte: Elaborada pelo autor

Foi identificada e inserida no processo a atividade de identificação de possíveis soluções alternativas para determinado requisito, fazer uma seleção dentre as possíveis soluções e documentar os critérios de seleção. Esta atividade está detalhada na Tabela 4.

Tabela 4 - Atividade – Desenvolver Soluções Alternativas e Critérios de Seleção

Atividade – Desenvolver Soluções Alternativas e Critérios de Seleção	
Finalidade: ⇒ Identificar as possíveis alternativas entre componentes de produtos ⇒ Seleção da melhor solução ⇒ Documentar critérios de seleção	
Artefatos de Entrada: ⇒ Arquitetura do Sistema ⇒ Documento de Requisitos ⇒ *Template* de Documento de Soluções ⇒ *Template* do Documento de Critérios de Seleção	Artefatos de Saída: ⇒ Documento de Soluções Alternativas ⇒ Documento de Soluções Selecionadas (incluindo justificativas) ⇒ Documento de Critérios de Seleção
Papel: ⇒ Engenheiro da Área de Negócio ⇒ Engenheiro de Componentes	
Ferramentas: ⇒ Planilha Eletrônica	
Passos: 1. Identificar as soluções alternativas 2. Identificar questões de *design* para cada alternativa de solução 3. Analisar cada alternativa de solução de com os critérios seleção 4. Fazer análise de comprar, construir ou reusar as possíveis soluções 5. Documentar o processo	

Fonte: Elaborada pelo autor

Uma outra atividade identificada e de elevado grau de importância é a de planejamento do *design* do produto. Esta atividade estabelece e mantém critérios pelo qual o *design* possa ser avaliado e identifica também os métodos apropriados para o *design* do produto. Esta atividade é descrita em detalhes na Tabela 5.

A atividade de verificação da aderência do *design* aos critérios planejados também foi identificada e documentada. Esta atividade se encontra detalhada na Tabela 6.

Tabela 5 - Atividade – Planejar o *Design* do Produto

Atividade – Planejar o *Design* do Produto	
Finalidade: ⇒ Estabelecer e manter critérios contra os quais o *design* pode ser avaliado ⇒ Identificar os métodos apropriados de *design* para o produto	
Artefatos de Entrada: ⇒ Arquitetura do Sistema ⇒ Documento de Requisitos ⇒ *Template* do Documento de Critérios do *Design* ⇒ *Template* do Documento de Métodos para o *Design*	Artefatos de Saída: ⇒ Documento de critérios do *design* ⇒ Documento de métodos para o *design*
Papel: ⇒ Engenheiro de Software ⇒ Gerente de Projetos	
Ferramentas: ⇒ Planilha Eletrônica	
Passos: 1. Estabelecer e manter critérios contra os quais o *design* pode ser avaliado. 2. Análise da arquitetura do sistema e documento de requisitos em relação ao método de *design* apropriado 3. Documentar os critérios e métodos apropriados	

Fonte: Elaborada pelo autor

Tabela 6 - Atividade – Assegurar Aderência do *Design*

Atividade – Assegurar Aderência do *Design*	
Finalidade: ⇒ Assegurar que o *design* adere aos padrões e critérios de *design* aplicáveis ⇒ Assegurar que o *design* adere aos requisitos alocados.	
Artefatos de Entrada: ⇒ Documento de Critérios do *Design* ⇒ Documento de Métodos para o *Design* ⇒ Arquitetura do Sistema ⇒ Diagramas de Classes de Controle ⇒ Diagramas de Classes de Entidades ⇒ Modelo Conceitual do Sistema	Artefatos de Saída: ⇒ Documento de Aderência do *Design* aos Requisitos Alocados
Papel: ⇒ Engenheiro de Software ⇒ Gerente de Projetos ⇒ Gerente da Área de Negócio	
Ferramentas: ⇒ Planilha Eletrônica	
Passos: 1. Análise de aderência com relação aos critérios de *design* 2. Análise de aderência com relação aos métodos de *design* 3. Análise de aderência aos requisitos alocados 4. Documentar a aderência aos requisitos alocados	

Fonte: Elaborada pelo autor

Foram identificadas as atividades de estabelecimento da descrição das interfaces e a de construção dos conectores. O estabelecimento das descrições das interfaces é exibida na Tabela 7. A construção de conectores está descrita na Tabela 8.

Tabela 7 - Atividade – Estabelecer Descrições de Interface

Atividade – Estabelecer Descrições de Interface	
Finalidade: ⇒ Identificar e documentar as interfaces necessárias	
Artefatos de Entrada: ⇒ Arquitetura do Sistema ⇒ Modelo Conceitual do Sistema ⇒ Documento de Soluções Selecionadas	Artefatos de Saída: ⇒ *Design* das Interfaces ⇒ Contratos da Interface
Papel: ⇒ Engenheiro de Software ⇒ Engenheiro de Componentes ⇒ Integrador de Componentes	
Ferramentas: ⇒ Ferramenta de Modelagem UML ⇒ Editor de Texto	
Passos: 1. Análise de integração das soluções selecionadas 2. Identificar interfaces necessárias 3. Projetar as interfaces necessárias 4. Documentar as interfaces necessárias	

Fonte: Elaborada pelo autor

Tabela 8 - Atividade – Construção de Conectores

Atividade – Construção de Conectores	
Finalidade: ⇒ Construir os conectores necessários para integração dos componentes	
Artefatos de Entrada: ⇒ *Design* da Interface ⇒ Contratos da Interface	Artefatos de Saída: ⇒ Conectores
Papel: ⇒ Engenheiro de Software ⇒ Programador	
Ferramentas: ⇒ Ambiente Integrado de Desenvolvimento	
Passos: 1. Implementação dos conectores de acordo com o especificado	

Fonte: Elaborada pelo autor

Para a integração dos componentes é fundamental que seja determinada uma ordem correta de integração. Esta ordem, selecionada dentre as possíveis ordens de integração, é documentada, bem como as justificativas utilizadas para a seleção da mesma. Esta atividade de determinação da ordem correta de integração do produto é apresentada detalhadamente na Tabela 9.

Tabela 9 - Atividade – Determina Seqüência de Integração

Atividade – Determina Seqüência de Integração	
Finalidade: ⇒ Determinar as possíveis ordens de integração do produto ⇒ Selecionar uma ordem de integração do produto	
Artefatos de Entrada: ⇒ Documento de Soluções Selecionadas ⇒ *Design* das interfaces	Artefatos de Saída: ⇒ Seqüência de integração do produto ⇒ Justificativas para selecionar ou rejeitar seqüências de integração
Papel: ⇒ Engenheiro de Software ⇒ Integrador de Componentes	
Ferramentas: ⇒ Editor de texto	
Passos: 1. Identificar os componentes de produtos a serem integrados. 2. Identificar as verificações da integração do produto a serem executadas, utilizando a definição das interfaces entre os componentes de produtos. 3. Identificar seqüências alternativas de integração de componentes de produtos. Isto pode incluir definir as ferramentas e equipamentos de testes específicos para suportar a integração do produto. 4. Selecionar a melhor seqüência de integração. 5. Periodicamente rever a seqüência de integração do produto e revisá-la, conforme necessário. 6. Avaliar a seqüência de integração do produto para assegurar que as variações nos cronogramas de produção e entrega não tiveram um impacto adverso na seqüência ou comprometeram os fatores sobre os quais as decisões anteriores foram tomadas. 7. Registrar as justificativas para as decisões tomadas.	

Fonte: Elaborada pelo autor

Além da determinação da ordem correta de integração, tem-se a necessidade do estabelecimento de critérios e procedimentos de integração do produto. Os critérios e procedimentos tratam de (entre outros):

- Verificação de interfaces;
- Requisitos derivados para a montagem e suas interfaces externas;
- Substituições permitidas de componentes;
- Limites do custo do teste;
- Equilíbrio de qualidade/custo para operações de integração;
- Intervalo de tempo do pedido até a entrega;
- Disponibilidade de pessoal.

A atividade de estabelecimento de critérios e procedimentos de integração é detalhada na Tabela 10.

Tabela 10 - Atividade – Estabelecer Critérios e Procedimentos de Integração do Produto

Atividade – Estabelecer Critérios e Procedimentos de Integração do Produto	
Finalidade: ⇒ Identificar e estabelecer critérios de integração do produto ⇒ Identificar e estabelecer procedimentos de integração do produto	
Artefatos de Entrada: ⇒ Documento de Soluções Selecionadas ⇒ *Design* das interfaces ⇒ Seqüência de integração do produto	Artefatos de Saída: ⇒ Procedimento de integração do produto ⇒ Critérios de integração do produto
Papel: ⇒ Engenheiro de Software ⇒ Engenheiro de Componentes ⇒ Integrador de Componentes	
Ferramentas: ⇒ Editor de texto	
Passos: 1. Estabelecer e manter os procedimentos de integração do produto para os componentes do produto. 2. Estabelecer e manter critérios para a integração e avaliação dos componentes do produto.	

Fonte: Elaborada pelo autor

Uma outra atividade identificada foi a da confirmação de que os produtos e os componentes de produtos estão prontos para a integração. Esta atividade tem como objetivo confirmar, antes da montagem, que cada componente do produto exigido para montar o produto foi apropriadamente identificado, funciona de acordo com sua descrição e que as interfaces do componente do produto estão em conformidade com as descrições de interfaces. O detalhamento desta atividade está na Tabela 11.

Tabela 11 - Atividade – Confirmar que os Componentes do Produto estão Prontos para Integração

Atividade – Confirmar que os Componentes do Produto estão Prontos para Integração	
Finalidade: ⇒ Confirmar, antes da montagem, que cada componente do produto exigido para montar o produto foi apropriadamente identificado, funciona de acordo com sua descrição e que as interfaces do componente do produto estão em conformidade com as descrições de interfaces.	
Artefatos de Entrada: ⇒ Documento de Soluções Selecionadas ⇒ *Design* das interfaces ⇒ Contratos das interfaces ⇒ Seqüência de integração do produto ⇒ Procedimento de integração do produto ⇒ Critérios de Integração do Produto	Artefatos de Saída: ⇒ Documentos de aceitação para os componentes do produto recebidos ⇒ Recibos de entrega ⇒ Listas de pacotes verificados ⇒ Relatórios de exceção
Papel: ⇒ Integrador de Componentes	
Ferramentas: ⇒ Editor de Textos	
Passos: 1. Rastrear o status de todos os componentes do produto, a partir do momento em que ficam disponíveis para integração. 2. Assegurar que os componentes do produto são entregues no ambiente de integração do produto em concordância com a seqüência de integração do produto e procedimentos disponíveis. 3. Confirmar o recebimento de cada componente do produto apropriadamente identificado. 4. Assegurar que cada componente do produto recebido atende sua descrição. 5. Verificar o status da configuração contra a configuração esperada. 6. Executar uma pré-verificação (por exemplo, por inspeção visual e utilizando medidas básicas) de todas as interfaces físicas, antes de conectar os componentes do produto.	

Fonte: Elaborada pelo autor

A atividade de montagem do produto a partir dos seus componentes foi identificada e documentada. Nesta atividade, os componentes do produto são montados de acordo com a seqüência de integração e procedimentos de integração disponíveis, obtendo-se o produto. O detalhamento desta atividade está na Tabela 12.

Foi identificada uma atividade de testes de integração, cujo objetivo é avaliar o produto montado de acordo com suas interfaces. Esta atividade é documentada na Tabela 13.

Tabela 12 - Atividade – Montar o Produto a Partir de seus Componentes

Atividade – Montar o Produto a Partir de seus Componentes	
Finalidade: ⇒ Montar os componentes do produto de acordo com a seqüência de integração e procedimentos disponíveis obtendo o produto.	
Artefatos de Entrada: ⇒ Seqüência de integração do produto ⇒ Procedimento de integração do produto ⇒ Critérios de Integração do Produto ⇒ Conectores ⇒ Componentes do Produto	Artefatos de Saída: ⇒ Produto montado
Papel: ⇒ Integrador de Componentes	
Ferramentas: ⇒ Ambiente de Desenvolvimento Integrado	
Passos: 1. Assegurar que a seqüência de montagem está sendo executada de forma apropriada. 2. Revisar a seqüência de integração do produto e procedimentos disponíveis, conforme apropriado. 3. Realizar a montagem do produto.	

Fonte: Elaborada pelo autor

Tabela 13 - Atividade – Testes de Integração

Atividade – Testes de Integração	
Finalidade: ⇒ Avaliar os componentes do produto montados com relação à compatibilidade da interface.	
Artefatos de Entrada: ⇒ Produto montado ⇒ Contratos da Interface	Artefatos de Saída: ⇒ Relatórios de exceção ⇒ Relatórios de avaliação de interfaces
Papel: ⇒ Testador	
Ferramentas: ⇒ Editor de Texto	
Passos: 1. Conduzir a avaliação dos componentes do produto montados, seguindo a seqüência de integração do produto e procedimentos disponíveis. 2. Registrar os resultados da avaliação. Exemplos de resultados incluem: • Qualquer adaptação necessária nos procedimentos de integração • Qualquer mudança na configuração do produto (peças de reposição, novo release) • Avaliação de desvios de procedimentos	

Fonte: Elaborada pelo autor

A fase de Desenvolvimento do processo, já modificada, é exibida na Figura 5.2.

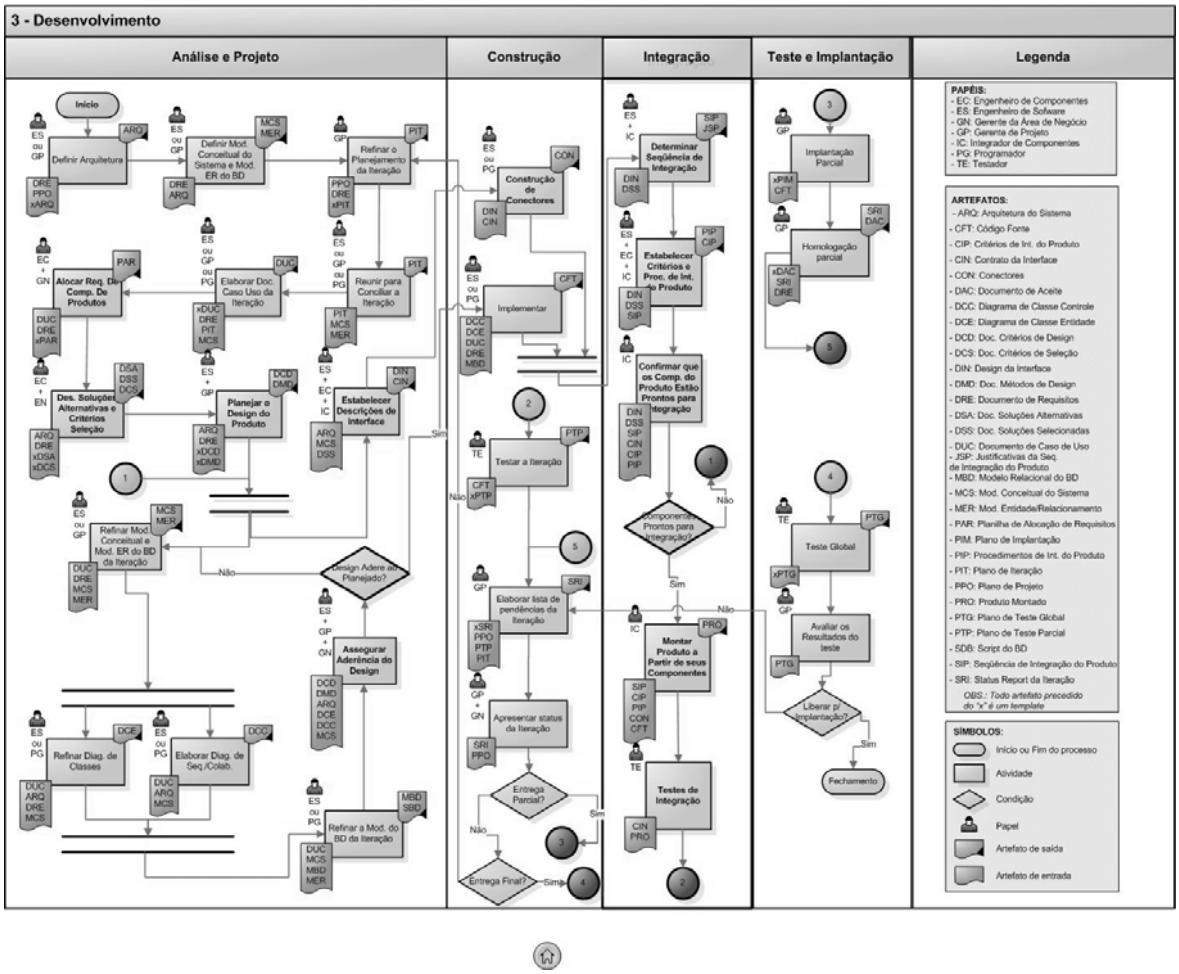

Figura 5.2 - Fase de Desenvolvimento Atualizada do PEPP
Fonte: Elaborada pelo autor

5.4. Institucionalização do Processo

O processo original, a partir do qual foi feita a modificação, já se encontra institucionalizado na empresa.

Esta versão do processo, para desenvolvimento com reuso, será aplicada na execução de um projeto piloto na qual estarão sendo observadas questões relacionadas a: dificuldade de uso, inconsistência e erros existentes no processo. Em paralelo estarão sendo acrescentadas e/ou corrigidas tanto as práticas que não foram atendidas na primeira versão do processo quanto os problemas encontrados durante o uso.

Acredita-se que o uso do novo processo ocorrerá sem grandes impactos na empresa, pois o processo manteve características do processo original, não sofrendo alterações drásticas.

6. CONCLUSÃO E TRABALHOS FUTUROS

6.1. Conclusão

O objetivo deste trabalho é realizar a adaptação de um processo de uma empresa de pequeno porte, que trabalha com desenvolvimento de software, para o desenvolvimento com reuso utilizando o CMMI.

Para atender ao objetivo do trabalho, foi realizado inicialmente um estudo sobre componentes, CBD e sobre o CMMI para serem aplicados na construção de uma versão de um processo de desenvolvimento de software para desenvolvimento com reuso.

Baseado nestes estudos, foi identificado, a partir de suas práticas específicas, que as áreas de processo que mais se relacionavam com o desenvolvimento baseado em reuso, eram as áreas de processo Desenvolvimento de Requisitos, Solução Técnica e Integração de Produto.

As três áreas de processo relacionadas com o desenvolvimento baseado em componente com reuso foram estudadas com maior profundidade e foram identificadas as principais práticas específicas relevantes ao CBD. Utilizando-se estas práticas específicas foi feita a modificação da fase de Desenvolvimento do PEPP, produzindo-se uma versão do PEPP para desenvolvimento com reuso.

Dentre as principais adaptações realizadas no PEPP encontram-se a criação de novos papéis, atividades e de uma nova etapa na fase de Desenvolvimento. Como exemplo de atividade criada, temos a Atividade Estabelecer Critérios e Procedimentos de Integração do Produto, a qual tem como finalidade identificar e estabelecer critérios e procedimentos de integração do produto, produzindo os artefatos Procedimento de Integração do Produto e Critérios de Integração do Produto. Esta atividade é executada pelo Engenheiro de Software, juntamente com o Engenheiro de Componentes e Integrador de Componentes. As novas atividades assim como as modificações realizadas foram detalhadas no Capítulo 5.

O processo resultante ainda não foi aplicado efetivamente em nenhum projeto para observação de resultado prático. Mas espera-se que os principais objetivos do reuso sejam refletidos na organização. Estes objetivos são: redução dos custos e do tempo de desenvolvimento e aumento da produtividade e da qualidade do produto de software.

Este trabalho também proporcionou ao autor um aprofundamento dos seus conhecimentos sobre os tópicos relacionados, bem como alguma experiência em desenho e modificação de processos.

6.2. Trabalhos Futuros

Uma das mais importantes continuações deste trabalho está na monitoração e acompanhamento do processo durante a execução do projeto piloto.

Além do acompanhamento, podem-se incluir novas atividades para reuso baseando-se em outros modelos como o OOSPICE (http://www.oospice.com/), FAA iCMM (http://www.faa.gov/aio/ProcessEngr/iCMM/index.htm) e ISO/IEC 15504 (http://www.sei.cmu.edu/iso-15504/).

A inclusão de atividades do desenvolvimento para reuso, fazendo com que o processo seja válido tanto para desenvolvimento para reuso quanto para desenvolvimento com reuso, também pode ser considerada.

Um outro trabalho significativo seria uma generalização do processo, o que faria com que o mesmo possa ser aplicado em outras organizações.

Finalizando, a realização de uma avaliação da maturidade do processo com relação ao desenvolvimento com reuso pode ser considerado também como um bom trabalho.

7. REFERÊNCIAS BIBLIOGRÁFICAS

AGUIAR, H. V. **PEPP: Processo de Software Para Empresas de Pequeno Porte Baseado no Modelo CMMI**. 2004. Monografia (Graduação em Ciência da Computação) – Universidade Federal de Lavras, Lavras. Disponível em <http://www.comp.ufla.br/monografias/ano2004/ano2004.htm>. Acesso em 03/03/2006.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **Gestão da qualidade e garantia da qualidade**: NBR ISO 8402. Rio de Janeiro, 1994.

BOEHM, B. Managing Software Productivity and Reuse**. Computer**, vol. 32, n. 9 p. 111-113, set. 1999. Disponível em
< http://ieeexplore.ieee.org/iel5/2/17107/00789755.pdf?isnumber=17107&arnumber=789755>.
Acesso em 02/03/2006.

CHEESMAN, J.; DANIELS, J. **UML Components: A Simple Process for Specifying Component-Based Software** – Addison-Wesley, 2001. 176p.

CHRISSIS, M.B.; KONRAD M.; SHRUM, S. **"CMMI – Guidelines for Process Integration and Product Improvement"** – SEI Series in Software Engineering – Addison-Wesley. 2004.

CRNKOVIC, I.; LARSSON, L. (Editors). **Building Reliable Component-Based Software Systems** – Artech House, 2002.

GIL, A. C. **Como Elaborar Projetos de Pesquisa**. 3. ed. São Paulo: Atlas, 1991.

INSTITUTE OF ELECTRICAL AND ELECTRONIC ENGINEERS. IEEE Std 610.12-1990 – **Standard glossary of software engineering terminology**. Piscataway: IEEE, 1990.

LEGO. **The LEGO Group.** Disponível em <http://www.lego.com/eng/default.aspx>. Acesso em 20/04/2006.

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA. Sociedade para Promoção da Excelência do Software Brasileiro – SOFTEX e Departamento de Política Científica e Tecnológica - DPCT/UNICAMP – "**Estratégia Nacional para Componentes de Software**".

PAULK, M.C.; WEBER, C.V.; CURTIS, B.; CHRISSIS, M.B.; et al.. **The Capability Maturity Model: Guidelines for Improving the Software Process**. Estados Unidos: Addison-Wesley. 1995.

ROSSI, A. C. "**Representação do Componente de software na FARCSOFT: Ferramenta de apoio à reutilização de componentes de software**". Dissertação de mestrado. Poli/USP, 2004.

SALVIANO, C. F. **Melhoria e Avaliação de Processo com ISO/IEC 15504 (SPICE) e CMMI**, 1ª ed, Editora UFLA, 2003.

SHORT, K. **Component Based Development and Object Modeling**. Sterling Software Disponível em <http://www.cool.sterling.com>. Acesso em 10/09/2005.

SOUZA, A. D. **Estudo e avaliação da área de processo de planejamento de projeto de acordo com o modelo CMMI-SW Nível 2 na empresa SWQuality situada em Lavras-MG**. 2004.

Monografia (Graduação em Ciência da Computação) – Universidade Federal de Lavras, Lavras. Disponível em <http://www.comp.ufla.br/monografias/ano2004/ano2004.htm>. Acesso em 03/03/2006.

VISIO. **Microsoft Office**. Disponível em <http://office.microsoft.com/pt-br/FX010857981046.aspx> . Acesso em: 03/03/2006.