UNIVERSIDADE DO VALE DO RIO DOS SINOS
UNISINOS

CoCoMo II
Um modelo para estimativa de custos de Projetos de Software

PABLO ARIEL DO PRADO LÓPEZ

São Leopoldo/ RS
2005

UNIVERSIDADE DO VALE DO RIO DOS SINOS
UNISINOS
CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE INFORMÁTICA

ANÁLISE DE SISTEMAS

CoCoMo II
Um modelo para estimativa de custos de Projetos de Software

PABLO ARIEL DO PRADO LÓPEZ

Orientador: Professor MSc. Candido Fonseca da Silva

Monografia submetida como requisito parcial para a obtenção do título de Bacharel em Informática.

São Leopoldo / RS

Junho 2005

AGRADECIMENTOS

Este trabalho teve a colaboração de muitas pessoas, que, direta ou indiretamente, tornaram possível o seu término. O trabalho evoluiu a partir da motivação de fazer da Informática um instrumento capaz de gerar vantagem competitiva às organizações.

Primeiramente, gostaria de agradecer aos meus pais, Orestes Mário do Prado Britos e Mabel Susana López Tavarez, ao meu irmão Mário Daniel e meu sobrinho William Daniel que, com amor e respeito souberam me acompanhar e me motivar durante essa jornada. Um agradecimento mais do que especial à minha mãe, que não está mais neste plano, mas que sempre me motivou a nunca desistir perante as dificuldades e momentos adversos; ela estará sempre em meu coração e dedico este momento a ela. Aos instrutores e colegas dos cursos que realizei antes de ingressar na universidade, por terem despertado meu interesse pela área de Sistemas de Informação e me conduzido nos primeiros passos da pesquisa científica e análise de sistemas.

Meus agradecimentos aos colegas, amigos, minha noiva Luciana e ao meu cachorrinho Floppy, que sempre entenderam, compartilharam, acompanharam e contribuíram para evolução deste trabalho.

Agradeço ainda, em especial, ao meu orientador, professor Candido, que muito me auxiliou, ensinou, incentivou e apoiou em todo o decorrer deste trabalho e no meu crescimento pessoal, com toda a sua experiência, dedicação, atenção e confiança em mim depositada.

RESUMO

Esta monografia mostra como o estabelecimento de estimativas se constitui em uma das principais atividades do planejamento do projeto de software. Propõe-se uma metodologia para a elaboração de estimativas utilizando o modelo Constructive Cost Model II (CoCoMo II) e apresenta-se o resultado da aplicação dessa metodologia considerando:

- Comparação de custo estimado pela empresa, o custo realmente alcançado e o custo proposto pela metodologia em projetos concluídos;
- Custo de implementação da metodologia;
- No aspecto de recursos humanos: tempo de treinamento da equipe que utilizará o modelo;
- No aspecto tecnológico e de projetos: comparação do custo estimado de projetos em andamento, em planejamento, e projetos concluídos, para justificar a mudança de paradigma na elaboração de estimativas.

ABSTRACT

This paper shows as estimates establishment constitutes one of the main activities of software project planning. A methodology for the elaboration of estimates is considered using the model Constructive Cost Model II (CoCoMo II) and it is presented the result of the application of this methodology considering:

- *Comparison of cost esteem for the company, the cost really reached and the cost considered for the methodology in finished projects;*
- *Methodology implementation costs;*
- *In the aspect of human resources : team training time that will use the model;*
- *In project and technological aspects: comparison of the esteem cost of projects in progress, planning, and finished projects, to justify the change of paradigm in the estimates elaboration.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO I – APRESENTAÇÃO

1.1 Introdução

Uma das funções do gerente de projetos de desenvolvimento de software é ter a capacidade de estimar e mensurar o custo, o tempo e esforço exigidos para os projetos de software a serem desenvolvidos ou em desenvolvimento. Para isto, ele deve conhecer a capacidade de sua equipe e os recursos com os quais pode contar para executar as atividades. Desta forma, adequando-se ao custo disponível e à qualidade desejada, o gerente poderá estabelecer prioridades ("*trade-offs")*[1] para a realização dessas atividades.

Ao elaborar uma estimativa para o desenvolvimento de um projeto de software, é desejável que haja um conhecimento sobre técnicas de estimativas e uma visão global do projeto a ser gerenciado, o que capacita os profissionais da área de informática a quantificar, administrar e planejar mais efetivamente os sistemas a serem produzidos.

Planejamentos com base empírica podem tornar inviável o projeto de software, justamente por não atender requisitos básicos, que abordam as etapas do ciclo de produção de software[2]. As conseqüências se tornam visíveis no resultado final, em retrabalho[3] durante a execução dos processos, ou o não atendimento das necessidades previstas e o atraso das atividades com o conseqüente atraso do projeto. Por esse motivo, é visível a necessidade de que as organizações realizem estimativas para o desenvolvimento de um projeto de software, para verificar a viabilidade do projeto, aumentando a competitividade e evitando perdas.

Ao se utilizar uma metodologia para planejamento adequado, permite-se que as atividades previstas para o desenvolvimento de um projeto tornem o trabalho mais claro em termos de tempo, de esforço e custo, respeitados seus *trade-offs*. Desta forma, cada integrante da equipe tem a sua função bem definida dentro do projeto e pode desenvolver o seu trabalho com o objetivo focado especificamente na atividade a ser desenvolvida.

No presente trabalho, será utilizado para estimativa de custos de projetos de software, o modelo *Constructive Cost Model* – CoCoMo, desenvolvido originalmente em 1981 pelo professor Dr Barry Boehm, pesquisador da Universidade do Sudeste da Califórnia (USC – *University of Southern California*) e sua equipe de cientistas e pesquisadores, encontrando-se atualmente na sua versão 2.0, de acordo com BOEHM (2000).

---

[1] *Trade-offs* são prioridades para a execução dos projetos; os mais utilizados são custo, tempo e qualidade, não necessariamente nesta ordem.

[2] Segundo PRESSMAN (1995), as etapas básicas do ciclo de produção de software são: a análise de requisitos, projeto, desenvolvimento, testes e implantação / manutenção.

[3] O retrabalho, de acordo com PRESSMAN (1995), caracteriza-se como ciclos em torno da mesma atividade, seja de análise, projeto, ou desenvolvimento de software.

1.2 Objetivos do Trabalho

1.2.1 Objetivo Geral

O objetivo geral do presente trabalho é avaliar, comparar e demonstrar, através do estudo de caso, que a utilização do CoCoMo II poderá trazer melhorias aos processos de planejamento e desenvolvimento de projetos de software.

Para isto, utilizou-se a metodologia em empresas que utilizam Tecnologia da Informação como suporte às suas atividades principais.

1.2.2 Objetivos Específicos

O presente trabalho apresenta como objetivos específicos:

- Demonstrar como a estimativa do custo dos produtos de projetos de desenvolvimento de software, com a utilização do modelo CoCoMo II, torna o seu desenvolvimento mais viável, em termos financeiros e de pessoal alocado em cada atividade;
- Comparar o controle de execução de produtos de software realizados hoje, com o proposto pelo modelo e verificar que as variáveis tempo, esforço, prazo e qualidade poderão ser mais facilmente atingidas, ou superadas, aplicando o modelo, do que pela percepção utilizada hoje em dia (com base empírica).
- Apresentar e propor uma metodologia para utilização do modelo estudado, que possa ser utilizada de forma adequada em empresas de perfil semelhante ao da empresa considerada no estudo de caso

1.3 Justificativa

O planejamento é de fundamental importância para o desenvolvimento de um projeto de software, visto que executar um projeto implica em realizar atividades que ainda não haviam sido realizadas anteriormente. Para planejar de forma adequada, torna-se necessária a combinação de todos os requisitos que compõem a etapa de planejamento de um projeto de software, que é o foco do presente trabalho, como ilustrado na figura 1. Os requisitos devem estar claramente definidos, para que seja alcançado o propósito de obter os resultados esperados, a saber: i) um projeto com qualidade; ii) concluído no prazo estipulado e iii) dentro dos custos alocados.

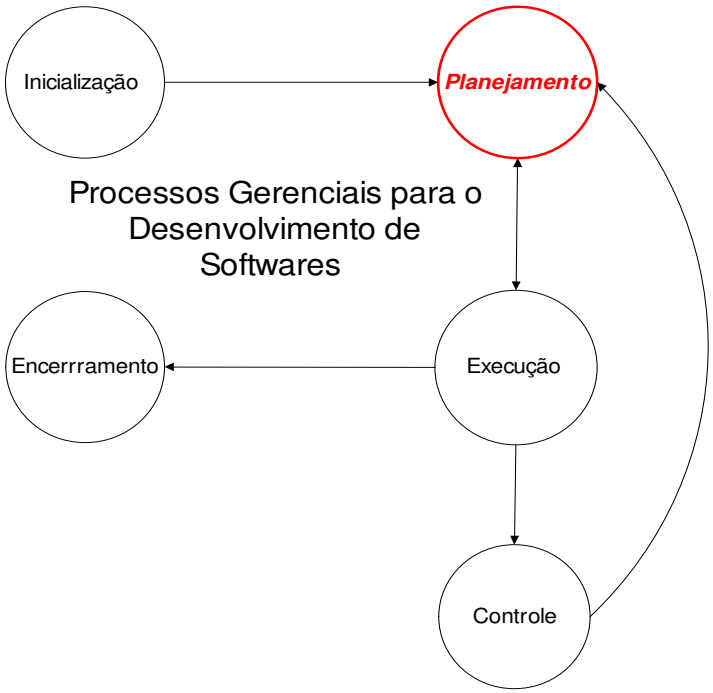


Figura 1 - Processos Gerenciais

Uma grande parte das empresas, atualmente, não utiliza planejamentos embasados para a produção de software; desta maneira, esta produção, com base em estimativas e planejamento com qualidade, é um elemento de competitividade que, apesar de extremamente relevante, ainda é pouco praticada segundo TRINDADE (1999). Essa pouca prática se deve, seja pela cultura das organizações ou a opinião de seus responsáveis, seja pela não utilização de metodologias adequadas ou, ainda, porque essas estimativas são realizadas baseadas no *feeling*[4] e experiência de quem coordena e desenvolve, sem que haja alguma estimativa anterior ou projeto semelhante como parâmetro real.

Contudo, constata-se que a qualidade dos produtos de projetos de software das empresas dependerá, de maneira mais intensa, de um planejamento elaborado de forma cada vez mais preciso, da qualidade das estimativas realizadas, e do gerenciamento adequado dos recursos disponíveis. Estas características podem ser implementadas nas empresas que ainda não utilizam estimativas embasadas através do modelo CoCoMo II. Justifica-se assim, um estudo aprofundado das variáveis e fatores necessários que influenciam na implantação desta metodologia.

1.4 Metodologia de Pesquisa

A metodologia considerada mais apropriada ao presente trabalho é a do estudo de caso qualitativo segundo YIN (1994), pois é considerado um método holístico para o desenvolvimento de pesquisas, e também permite que a contextualização do problema seja bem desenvolvida. O perfil de empresa estudada caracteriza-se pela sua atividade principal ser a área de engenharia, utilizando a Tecnologia da Informação (TI) como ferramenta de suporte às suas atividades

[4] Vem do inglês Sentimento, Percepção, Afinidade com alguma atividade, objeto ou pessoa;

principais. São utilizadas normas e padrões de TI, aplicando modelos de engenharia de software, para o planejamento e desenvolvimento dos seus processos de software, assim como no desenvolvimento de seus produtos de software. Desta forma o modelo CoCoMo II foi utilizado para avaliar o planejamento de desenvolvimento de produtos da empresa, pela comparação da forma com que todos esses processos são realizados atualmente, com o resultado da aplicação do modelo CoCoMo II.

Os critérios para escolha da empresa alvo para o estudo de caso seguem, segundo GIL (1994), o critério da conveniência, ou acessibilidade, visto que neste método o pesquisador seleciona os dados a que tem acesso, e estes representam o todo dentro de sua pesquisa, ou estudo de caso, que é o realizado neste contexto. O trabalho está dividido em 3 fases, descritas a seguir:

1. Pesquisa do modelo CoCoMo II, utilizando como referências: literaturas, artigos, publicações e material colhido da Internet.

Nesta fase, o foco da pesquisa abrange: a origem do modelo, seu desenvolvimento, formas de utilização para estimativa de custo e estudos atuais que vêm sendo feito sobre o modelo.

2. Estudo de caso em uma empresa que possui o perfil citado anteriormente, com base nas metodologias para realização de estudos de caso descritas segundo GIL (1994).

O estudo de caso foi focalizado nos processos de desenvolvimento de software de um setor desta empresa, o que caracteriza um estudo de caso simples com apenas uma unidade de análise de acordo com YIN (1994).

Durante o estudo de caso, foram colhidas informações para três momentos estudados:

- As estimativas dos produtos de software são em número de 4 (quatro), sendo que dois estão concluídos, outro está em andamento e o quarto está em fase de planejamento. Cada produto é um momento diferente de estudo, onde se realizou uma comparação entre a estimativa da forma como é elaborada atualmente, a estimativa elaborada pelo CoCoMo II e o que foi efetivamente realizado no desenvolvimento dos produtos. Através destas comparações, pode-se verificar a influência do modelo CoCoMo II nos processos de planejamento, apresentando para a empresa *case* os resultados obtidos.

3. Elaboração de uma documentação contendo o mapeamento dos processos estudados e suas conclusões como uma proposta de utilização do modelo CoCoMo II para elaboração de estimativas para os projetos de desenvolvimento de software com o resultado das discussões com a empresa alvo.

A figura 2 apresenta a ordem em que a metodologia foi desenvolvida.

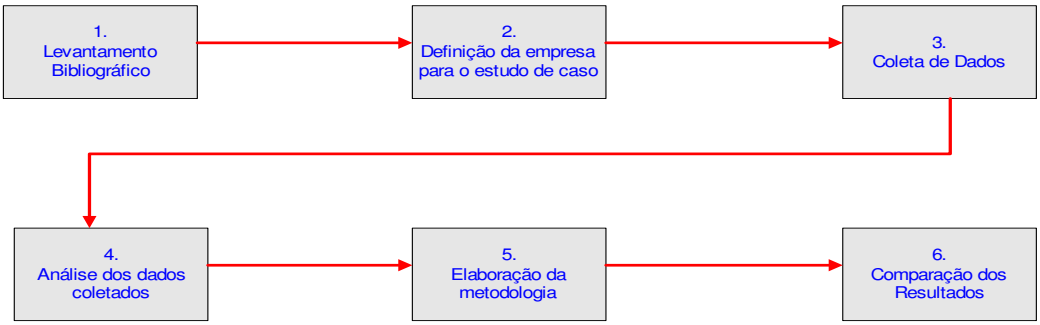


Figura 2 - Seqüência de aplicação da metodologia

1.5 Estruturação do Trabalho

O trabalho está estruturado em cinco (05) capítulos: a apresentação, a revisão bibliográfica, a metodologia do estudo, os resultados do estudo de caso e a análise integrada dos resultados.

No capítulo I (um), descreve-se sucintamente o tema, a justificativa, os objetivos da pesquisa e a estrutura do trabalho. No capítulo II (dois), apresenta-se a revisão bibliográfica, abrangendo os tópicos relacionados ao trabalho. No capítulo III (três), apresenta-se a metodologia utilizada no levantamento de dados para a confecção do trabalho. No capítulo IV (quatro), é apresentado o resultado do estudo de caso. E, por fim, no capítulo V (cinco) confronta-se a teoria com a prática, fazendo uma análise integrada do estudo de caso à luz do referencial teórico, bem como um comparativo dos resultados obtidos.

CAPÍTULO II – REVISÃO BIBLIOGRÁFICA

2.1 Gerência de Projetos

2.1.1 Conceito

A Gerência de Projetos é a combinação de pessoas, técnicas e sistemas necessários à administração dos recursos indispensáveis ao objetivo de atingir o êxito final do projeto, de acordo com DINSMORE (1992). Pode, ainda, ser definida como a aplicação de conhecimentos, habilidades, ferramentas e técnicas às atividades do projeto, de forma a atingir e exceder as necessidades e expectativas das partes envolvidas.

O ato de gerenciar um projeto pode ser entendido e realizado de várias formas diferentes; essas formas vão depender da cultura e dos objetivos da empresa. Muitas organizações ainda desenvolvem conceitos próprios para orientar e gerenciar suas atividades.

Para melhor entender as práticas e conceitos de Gerência de Projetos, existe um conceito relevante ao contexto dessa atividade, de acordo com o PMI (2004), "Um projeto é um empreendimento temporário, conduzido para criar um produto ou serviço único, que possui um início e um final definido." e consiste em nove (9) áreas principais, de acordo com PMBOK (2004):

2.1.1.1 Gerência de Custo

Descreve os processos necessários para assegurar que o projeto seja completado dentro do orçamento previsto e é composto pelo planejamento de recursos, estimativa de custos, orçamento de custos e controle de custos.

2.1.1.2 Gerência de Tempo

Descreve os processos necessários para assegurar que o projeto termine dentro do prazo previsto e é composto pela definição das atividades, seqüenciamento das atividades, estimativa de duração das atividades, desenvolvimento do cronograma e controle do cronograma.

2.1.1.3 Gerência de Escopo (Abrangência)

Descreve os processos necessários para assegurar que o projeto contemple todo o trabalho requerido, e nada mais que o trabalho requerido, para completar o projeto com sucesso e é composto pela iniciação, planejamento do escopo, detalhamento do escopo, verificação do escopo e controle de mudanças do escopo.

2.1.1.4 Gerência da Qualidade

Descreve os processos necessários para assegurar que as necessidades que originaram o desenvolvimento do projeto serão satisfeitas e é composto pelo planejamento da qualidade, garantia da qualidade e controle da qualidade.

2.1.1.5 Gerência das Comunicações

Descreve os processos necessários para assegurar que a geração, captura, distribuição, armazenamento e pronta apresentação das informações do projeto sejam feitas de forma adequada e no tempo certo, sendo composto pelo planejamento das comunicações, distribuição das informações, relato de desempenho e encerramento administrativo.

2.1.1.6 Gerência de Recursos Humanos

Descreve os processos necessários para para proporcionar a melhor utilização das pessoas envolvidas no projeto e é composto pelo planejamento organizacional, montagem da equipe e desenvolvimento da equipe.

2.1.1.7 Gerência de Riscos

Descreve os processos necessários que dizem respeito à identificação, análise e resposta a riscos do projeto e é composto pela identificação dos riscos, quantificação dos riscos, desenvolvimento das respostas aos riscos e controle das respostas aos riscos.

2.1.1.8 Gerência de Integração

Descreve os processos necessários para assegurar que os diversos elementos do projeto sejam adequadamente coordenados e é composto pelo desenvolvimento do plano de projeto, execução do plano de projeto e controle geral de mudanças.

2.1.1.9 Gerência de Aquisições

Descreve os processos necessários para a aquisição de mercadorias e serviços fora da organização que desenvolve o projeto e é composto pelo planejamento das aquisições, preparação das aquisições, obtenção das propostas, seleção de fornecedores, administração dos contratos e encerramento do contrato.

2.1.2 O Gerenciamento de Projetos

Gerenciamento de projetos é a aplicação de conhecimentos, habilidades, ferramentas e técnicas para projetar atividades de forma a atender ou exceder as expectativas dos "*stakeholders*" concernentes ao produto ou serviço gerado pelo projeto.

Atender ou exceder às expectativas dos "stakeholders" relativamente ao projeto, invariavelmente implica em:

- Atender a escopo, tempo, custo e qualidade.
- Atender a requisitos de projeto e expectativas que não tinham sido identificadas na definição do projeto.

Para atender a estas metas é necessário que a missão do projeto esteja bem definida, isto é deve definir-se os aspectos relativos a:

- **Estratégia**: que problema deve ser solucionado
- **Pessoas:** a quem ele se destina
- **Processos:** como o projeto deve ser elaborado para atingir seus objetivos

A Gerência deve formar uma equipe onde todos os seus membros entendam o relacionamento de suas atividades com a dos outros membros (Fluxo) e garantir que todas as pessoas envolvidas no projeto tenham visão abrangente do projeto, de modo que todas as atividades previstas sejam executadas conforme o cronograma estabelecido.

Também é necessário garantir que as pessoas envolvidas sejam capazes de ter a mesma visão de todos os processos que estão sendo desenvolvidos, ou seja, garantir a Visibilidade do Projeto.

2.1.3 O Gerente de Projetos

"A disciplina de gerência de projeto é independente da área de aplicação. Os gerentes de projeto, não!" , DINSMORE (1992).

Três características devem ser consideradas na alocação de um gerente a um projeto: gerenciais, administrativas e técnicas.

Entretanto, nenhuma destas características deve tender a zero. Portanto, é necessário preparar os recursos humanos dedicados ao gerenciamento de projetos nestas três dimensões, segundo DINSMORE (1992).

Em resumo, as principais características do gerente de projeto são:

- Competência técnica;
- Liderança;
- Resolução de problemas;
- Suporte gerencial;
- Construção de equipes.

As pessoas envolvidas (usuário, cliente, patrocinadores, equipe) são o principal recurso de um projeto. Entretanto, as equipes podem variar em experiência, apresentando diferentes graus de autonomia e habilitação no trabalho em equipe. Portanto, além de investir na capacitação da equipe, o gerente de projetos deve utilizar diferentes formas de exercício da liderança conforme o tipo de equipe disponível em DINSMORE (1992).

Além disso, o uso de metodologias de gerenciamento, como a utilização de estimativas e métricas, com suas práticas e ferramentas relacionadas, pode determinar o sucesso ou fracasso de um projeto. Neste caso, o principal aspecto a ser considerado é a perseverança no uso e a confiança nos resultados obtidos pela prática destas metodologias.

2.2 Normas e Padrões

2.2.1 Normas

As normas técnicas são um processo de simplificação pois reduzem a crescente variedade de procedimentos e produtos. Assim, elas eliminam o desperdício, o retrabalho e facilitam a troca de informações entre fornecedor e consumidor ou entre clientes internos.

Como instrumento, as normas técnicas contribuem em quatro aspectos:

- Qualidade: fixando padrões que levam em conta as necessidades e desejos dos usuários.
- Produtividade: padronizando produtos, processos e procedimentos
- Tecnologia: consolidando, difundindo e estabelecendo parâmetros consensuais entre produtores, consumidores e especialistas, colocando os resultados à disposição da sociedade.
- Marketing: regulando de forma equilibrada as relações de compra e venda

Dentre as diversas instituições que regulamentam e aplicam as normas, a empresa estudada no *case* utiliza duas para aplicar e implementar e regulamentar

padrões nos seus processos : A ABNT (Associação Brasileira de Normas Técnicas) – <http://www.abnt.com.br> e a ISO (*International Organization for Standardization*), na qual possui a certificação ISO 9001/2000.

2.2.2 Padrões

Padrões definem normas a serem seguidas obrigatoriamente ou preferencialmente, sendo uma maneira testada ou documentada de atingir um objetivo qualquer.

Os padrões referem-se à comunicação de problemas e soluções. Em outras palavras, os padrões permitem documentar um problema conhecido recorrente e sua solução em um contexto específico e comunicar esse conhecimento para outras pessoas, sendo uma das melhores práticas comprovadas e documentadas por alguém e que permitem solucionar um determinado problema em um determinado contexto, segundo PRESSMAN (1995).

> *"Cada padrão é uma regra de três partes, que expressa uma relação entre um certo contexto, um problema e uma solução".PRESSMAN (1995).*

Um padrão é melhor aplicado quando ele estiver inserido no contexto ao qual foi designado, ou seja, se existe um problema deve-se procurar entre os padrões já conhecidos qual a melhor solução. A característica de tentar aplicar um padrão sem um problema é conhecida como *Anti-Patterns*, segundo PRESSMAN (1995).

Dentre os diversos padrões existentes para desenvolvimento de projetos de software, a empresa *case* está em processo de estudo e implantação dos padrões a seguir apresentados:

2.2.2.1 *Project Management Institute* – PMI

O *Project Management Institute*, **PMI® -** PMI (2004) é uma organização de profissionais da área de gerenciamento de projetos. O PMI visa promover e ampliar o conhecimento existente sobre gerenciamento de projetos, assim como melhorar o desempenho dos profissionais e organizações da área.

O PMI estabelece padrões de gerenciamento de projeto, provê seminários, programas educacionais e certificação profissional que cada vez mais as organizações desejam para os seus líderes de projeto.

Para isto, o PMI apóia a criação de redes de informação e de intercâmbio entre os profissionais no mundo inteiro. Um dos instrumentos para alcançar os seus objetivos é o apoio à formação de seções locais. Um outro instrumento é a formação de SIGs (*Specific Interest Group*) para agregar interessados no mesmo ramo de atuação.

2.2.2.2 *Project Management Body of Knowledge* – PMBOK

O *Project Management Body of Knowledge*, **PMBOK® -** PMBOK (2004) é um guia publicado pelo PMI onde se descreve a somatória de conhecimento e as melhores práticas dentro da área de gerência de projetos. Todo o conhecimento reunido neste guia é comprovado e não se restringe somente a práticas tradicionais, mas também às inovadoras e avançadas. Ele é um material genérico que serve para todas as áreas de conhecimento, ou seja, tanto para construção de edifício ou

processo de fabricação industrial como para a produção de software. Um outro objetivo do PMBOK é a padronização de termos utilizados em gerência de projetos.

Este guia é organizado em áreas de conhecimento e, por sua vez, cada área de conhecimento é descrita através de processos. Cada área de conhecimento se refere a um aspecto a ser considerado dentro da gerência de projetos. As áreas de conhecimento são as citadas na seção 2.1 deste trabalho (Gerência de Projetos).

Diante do contexto apresentado e dentro do perfil em que a empresa a ser estudada se enquadra, os conceitos sobre gerenciamento de projetos apresentados no PMBOK têm sido estudados de forma a tornarem-se úteis, viáveis e aplicáveis e assim padronizar suas ações em busca de atingir e superar suas metas em relação a custo, prazo e qualidade dos projetos em desenvolvimento e a serem desenvolvidos.

2.2.2.3 *International Organization for Standardization* – ISO

A ISO é o organismo que estabelece os padrões internacionais de trabalho e de garantia de qualidade nas empresas. Desde 1987 foi criada uma série de normas, conhecidas como ISO 9000, que deram início a um sistema de gestão da qualidade, verificado em ISO (2004b).

A ISO 9000 estabelece as orientações básicas para a correta seleção e uso das normas, tanto que seus objetivos são esclarecer as diferenças e inter-relações entre os principais conceitos da qualidade e, fornecer diretrizes para seleção e uso de normas que servem para gestão da qualidade. A empresa *"case"* possui a certificação 9001-2000, que garante a qualidade externa, e pretende utilizar os conceitos da futura norma ISO/IEC 15504 - ISO (2004b) - que se refere à realização de avaliações dos processos de software.

Para isso pretende, portanto, identificar as áreas problemáticas e estabelecer ações sistemáticas de melhoria. Como a melhoria da qualidade do produto final é tipicamente atingida pela melhoria do próprio processo produtivo, a melhoria dos processos de software é um dos principais objetivos da empresa.

2.2.2.4 *Capability Maturity Model* – CMM

O SEI (*Software Engineering Institute*), centro de pesquisa e desenvolvimento da *Carnegie Mellon University*, define CMM como Modelo de Maturidade da Capacidade (*Capability Maturity Model*). O primeiro CMM desenvolvido pelo SEI foi o SW-CMM (*Capability Maturity Model for Software*), focado apenas na disciplina da engenharia de *software*. Depois desse, outros CMM's foram desenvolvidos, focados em outras disciplinas, como engenharia de sistemas, sub-contratação, pessoas e desenvolvimento integrado de produtos.

O CMM originou-se de pesquisas e trabalhos do SEI, baseando-se na visão de Watts Humphrey, pesquisador vindo da IBM. Seus trabalhos objetivaram adaptar uma metodologia de qualidade de processos para a área de software, visando que esta pudesse ser adotada para a contratação de produção externa de software, no sentido de estabelecer um nível de maturidade que o prestador de serviços deveria ter para atender estas demandas, de acordo com CMU/SEI (1997).

O SW-CMM descreve os elementos-chave da evolução de um processo de *software* imaturo para um processo maduro e disciplinado. Abrange práticas para planejamento, engenharia e gestão do desenvolvimento de *software* que, quando

seguidas, melhoram a habilidade da organização em atender metas para custos, cronograma, funcionalidade e qualidade do produto.

2.2.2.5 *Capability Maturity Model Integration* – CMMI

Para solucionar os problemas gerados pelo surgimento de outros CMM's e de normas não contempladas, o SEI iniciou um projeto chamado CMMI, com o objetivo de gerar uma nova versão do CMM para resolver esses problemas, vide SEI (2004).

O objetivo do CMMI é integrar os diversos CMM's numa estrutura única, todos com a mesma terminologia, processos de avaliação e estrutura. Além disso, o projeto também se preocupou em tornar o CMM compatível com a norma ISO/IEC 15504 - ISO (2004a), de modo que avaliações em um modelo sejam reconhecidas como equivalentes aos do outro e, naturalmente, incorporar ao CMM as sugestões de melhoria surgidas e sugeridas ao longo dos anos por CMU/SEI (1997).

O CMMI é um guia desenvolvido pela comunidade de software. É um modelo único que integra os CMM's. Incorpora as necessidades de melhorias identificadas pelo uso do CMM no mundo, é compatível com a norma ISO/IEC 15504, alinhado com o PMBOK e possui um direcionamento claro e objetivo para a interpretação das práticas, apresentando sub-práticas e produtos típicos de trabalho para cada prática, sendo apresentado em duas representações, vistas em SEI (2004): a estagiada e a contínua.

A representação estagiada pode ser utilizada para verificar o nível de maturidade da organização em geral. Cada área de processo pertence exclusivamente a um dos cinco níveis de maturidade. Para obter certificação em determinado nível, a organização precisa atender às metas de todas as áreas de processo definidas para o nível e às metas das áreas de processo definidas para os níveis anteriores.

A estrutura desta representação do modelo é composta de cinco níveis de maturidade. Cada nível de maturidade é composto de Áreas de Processo (PA´s – *Process Areas*), compostas de metas específicas e genéricas. As metas específicas possuem práticas específicas e as metas genéricas possuem características comuns compostas de práticas genéricas

A representação contínua pode ser utilizada para verificar o nível de capacidade dos processos. Nesta, cada área de processo possui características relativas a mais de um nível. Assim, uma área de processo que, na representação estagiada, pertence exclusivamente ao nível dois, no modelo contínuo pode ter características que a coloquem em outros níveis, ou seja, cada área de processo é classificada separadamente.

De acordo com o guia publicado em SEI (2004), a representação contínua do CMMI está organizada em seis níveis de capacidade, numerados de zero a cinco, que fornecem uma ordem recomendada aproximando a melhoria dos processos dentro de cada área. À medida que as metas específicas e genéricas são atingidas para uma área de processo em um determinado nível de capacidade, os benefícios da melhoria de processos são obtidos.

Os níveis de capacidade são focados no aumento da capacidade da organização em executar, controlar e melhorar o seu desempenho em uma área de processo. Ainda, os níveis de capacidade permitem monitorar, avaliar e demonstrar

a evolução de uma organização na melhoria dos processos associados a uma área, e no escopo deste trabalho será abordado o nível 2 desta representação.

- ✓ Nível 2: Gerenciado

Um processo com nível de capacidade 2 é caracterizado como um processo gerenciado, que além de ser executado também é planejado e executado de acordo com políticas, emprega pessoas capacitadas e recursos adequados para produzir resultados controlados, envolve *stakeholders* relevantes.

Um processo gerenciado é monitorado, controlado, revisado e avaliado quanto à aderência em relação a sua descrição. O processo pode ser iniciado por um projeto, grupo ou função organizacional específicos. A gerência do processo preocupa-se com a institucionalização da área de processo e com a realização de outros objetivos específicos estabelecidos para o processo, tais como custo, prazo e qualidade.

2.2.2.6 *Rational Unified Process* - RUP

O *Rational Unified Process* (RUP), segundo KRUTCHEN (2003), é um **processo de engenharia de software***,* fornecendo uma abordagem disciplinada para assumir tarefas e responsabilidades dentro de uma organização de desenvolvimento. Seu objetivo é assegurar a produção de software de alta qualidade que satisfaça as necessidades de seus usuários finais dentro do prazo e orçamento previsíveis.

O RUP é um **produto de processo**, e também uma **estrutura de processo***,* que pode ser adaptada e estendida para compor as necessidades de uma organização que o esteja adotando, segundo KRUTCHEN (2003), como é o caso da empresa *case*. É desenvolvido e mantido pela Rational Software, segundo RATIONAL UNIFIED PROCESS (2004) e integrado com seu conjunto de ferramentas de desenvolvimento de software.

O RUP captura muitas das melhores práticas no desenvolvimento moderno de software, de forma satisfatória para uma grande faixa de projetos e organizações, e em particular, cobre seis práticas:

1. Desenvolver software iterativamente;
2. Gerenciar exigências;
3. Usar arquiteturas baseadas em componente;
4. Modelar visualmente o software;
5. Verificar continuamente a qualidade do software;
6. Controlar mudanças no software.

O RUP adota as seguintes premissas básicas:

- Uso de iterações para evitar o impacto de mudanças no projeto,
- Gerenciamento de mudanças e
- Abordagens dos pontos de maior risco o mais cedo possível.

A figura 3 apresenta os elementos básicos do RUP. Nesta metodologia, o projeto passa por 4 fases básicas. Estas fases são:

- Iniciação (*Inception*) - entendimento da necessidade e visão do projeto; definição dos objetivos e viabilidade do projeto (a idéia do projeto) e do escopo de vários aspectos.
- Elaboração (*Elaboration*) - especificação e abordagem dos pontos de maior risco; eliminação dos elementos de maior risco do projeto através da criação de uma arquitetura coerente e consistente da solução.
- Construção (*Construction*) - desenvolvimento principal do sistema; desenvolvimento de todos os componentes e características não resolvidas nas fases anteriores, testando-as e integrando-as na forma de um produto.
- Transição (*Transition*) - ajustes, implantação e transferência de propriedade do sistema; realiza a transição do produto para a comunidade de usuários

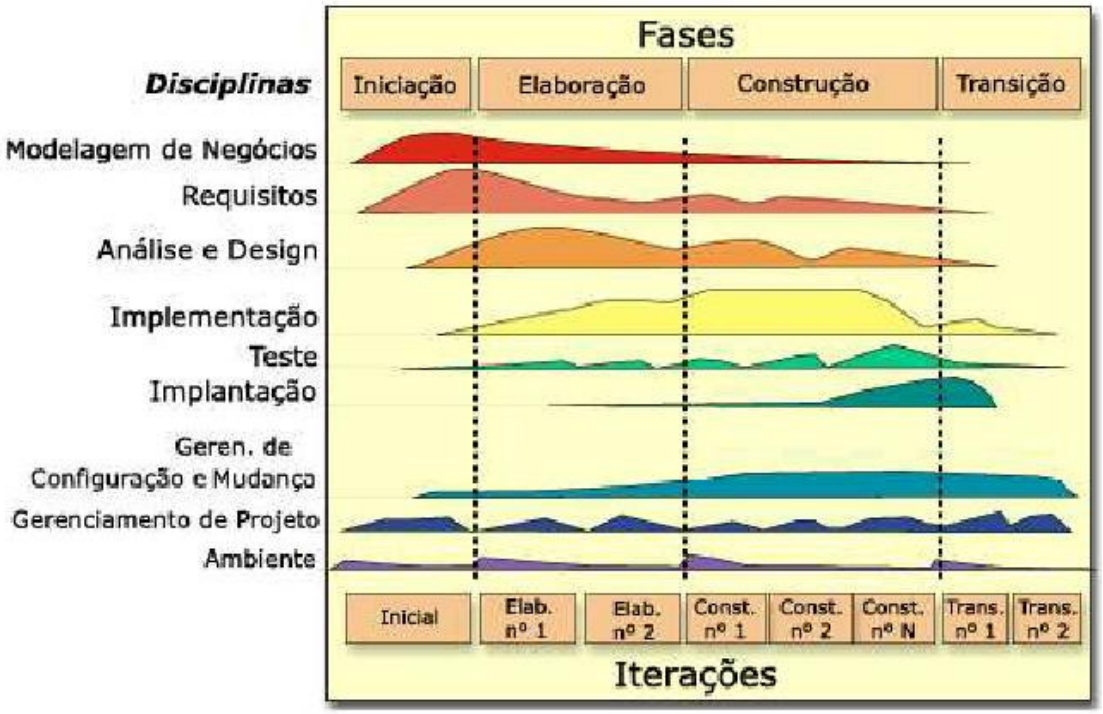


Figura 3 - Organização e Estrutura do RUP – extraído de KRUTCHEN (2003)

Apesar de parecer um modelo em cascata, na verdade cada fase é composta de uma ou mais iterações, o que se assemelha a um modelo em espiral. Estas iterações são em geral curtas (1-2 semanas) e abordam algumas poucas funções do sistema. Isto reduz o impacto de mudanças, pois quanto menor o tempo, menor a probabilidade de haver uma mudança neste período para as funções em questão.

Além das fases e iterações, o RUP oferece recursos adicionais para controle e desenvolvimento de projetos, que são os workflows, as tarefas, modelo de equipe e modelo de documento. Para maiores detalhes consulte KRUTCHEN (2003).

Com base nestes recursos oferecidos, a adoção do RUP pode ser feita de mais de uma maneira. Uma maneira seria usar o RUP à risca, ou seja, aplicar todos os métodos e processos exatamente como são propostos. A vantagem desta abordagem é que nada deve ser alterado.

Outra maneira seria adotar outro modelo de processo mais simples ou conhecido e utilizar o material do RUP como fonte de referência complementar para assuntos não abordados em outro modelo como, por exemplo, os modelos de documentos.

A primeira abordagem é interessante para empresas que precisam de uma grande formalização do processo de desenvolvimento de software e cujo método atual seja totalmente inadequado ou inexistente. A segunda abordagem seria interessante para quem já tem alguma metodologia que considera adequada, mas que tem deficiência em alguma área como, por exemplo, suporte a *Unified Modeling Language* - UML. Soluções intermediárias também são possíveis.

A empresa *case*, por estar numa fase de mudança de paradigmas, está se adequando para implementar a segunda abordagem nos seus processos de desenvolvimento, visto que há uma pequena formalização nos seus processos, e principalmente, está em fase de adoção de modelagem UML.

2.2.2.7 Métricas

As métricas referem-se às várias formas que existem para medir o tamanho de um Software; através destes métodos é possível determinar numericamente algum aspecto importante do software. As métricas ajudam a avaliar a qualidade do produto, metodologias e ferramentas que estão sendo utilizadas no processo, além de servirem de base para as estimativas de custo, tempo e recursos humanos necessários ao desenvolvimento. Assim, as medições auxiliam na tomada das melhores decisões possíveis.

Categorização das métricas

As medições podem ser divididas em duas categorias: medidas diretas e indiretas.

Medidas diretas - são todas aquelas em que se pode fazer convenções específicas para medição e que sejam estabelecidas antecipadamente.

Medidas indiretas - são parâmetros difíceis de serem avaliadas como qualidade e eficiência.

Na Engenharia de Software considera-se como medidas diretas: custo, esforço, linhas de código, velocidade de execução, tamanho de memória e outros. São definidas como medidas indiretas: funcionalidade, complexidade, confiabilidade, manutenibilidade, entre outras.

Podem ainda ser consideradas as seguintes categorias:

Métricas de Produtividades

Concentram-se na saída do processo de engenharia de software.

Métricas de Qualidade

Oferecem uma indicação de quanto o software se adequou às exigências implícitas e explícitas do cliente.

Métricas Técnicas

Concentram-se na característica do software (complexidade lógica e grau de manutenibilidade) e não no processo por meio do qual o software foi desenvolvido.

Métricas Orientadas ao Tamanho

Compilam as medições diretas da saída e da qualidade da engenharia de software.

Métricas Orientadas às pessoas

Compilam informações sobre a maneira como as pessoas desenvolvem software de computador e percepções humanas sobre a efetividade das ferramentas e métodos.

2.2.3 Estimativas

O poder de estimar constantemente exercitado no dia-a-dia: quanto tempo é necessário para chegar a determinado local, quantas pessoas havia em um evento, qual será a temperatura no final de semana, qual o placar do jogo, quanto custará o conserto do carro.

Para tudo isso existem respostas prontas, mesmo que com pouca ou nenhuma informação, disponíveis a quem quiser ouvir. Estimar, portanto, é um processo gerador de expectativas, que pode resultar em grandes frustrações, se não atender as expectativas ou se for mal elaborado.

Em projetos isso não é diferente. A figura 4 apresenta o processo de estimativa para projetos. Para um projeto ser planejado, os objetivos e escopo devem ser estabelecidos, devem existir soluções alternativas a ser consideradas e restrições administrativas e técnicas identificadas. Sem essas informações é muito difícil definir estimativas de custos razoáveis, de acordo com KIVAL (1999).

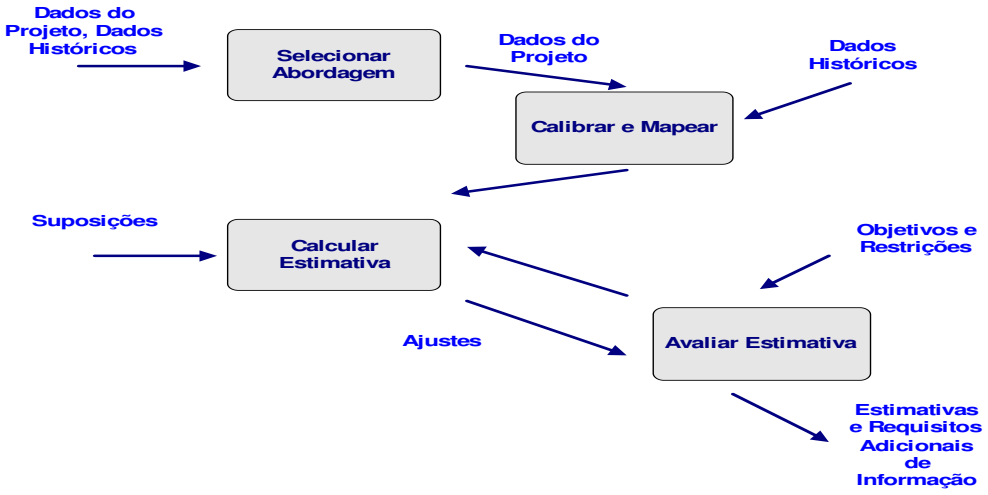


Figura 4 - O processo de estimativa

A realidade mostra que a grande maioria dos projetos não falha necessariamente por ter excedido o tempo ou ultrapassado o custo previsto Na verdade, excederam a expectativa de tempo ou estouraram a expectativa de custo, visto que a prática habitual de estimativa nas organizações resume-se em perguntar para alguém experiente qual o esforço necessário para a realização de determinada tarefa, trazendo quatro conseqüências imediatas:

1) por ser um processo subjetivo, não há um parâmetro racional para gerar as estimativas, isto é, se houver uma alteração nas premissas, dificilmente será possível reaproveitar ou readequar as estimativas;

2) tem-se a tendência a estimar baseados na experiência individual e produtividade de cada um, o que pode não refletir a experiência e produtividade da organização ou de um determinado time de projeto;

3) cria-se uma dependência e um gargalo nos especialistas em estimativa, pois só eles detêm alguma certa fórmula empírica e subjetiva para estimar;

4) não há um racional sólido para defender as estimativas quando questionados ou pressionados a reduzi-las (tudo se resume ao sentimento do especialista) e elas podem ser rotuladas de hipóteses, sugestões (mesmo que corretas) e tornarem-se sensíveis à pressão.

É inegável, no entanto, que várias organizações têm um bom registro de acerto médio em suas estimativas, observado nos projetos. Normalmente, há a tendência de não acertar esforço e custo de um projeto baseado somente na opinião de especialistas. O que se necessita nas estimativas realizadas, para que possuam qualidade, é aumentar o grau de previsibilidade (certeza) do que é estimado, e não necessariamente conseguir manter uma boa média. Isto é, a média pode estar perfeita, mas é para previsibilidade que o foco deve ser direcionado, segundo PMI (2004).

2.2.3.1 Métodos de Estimativa e Previsibilidade

Segundo PMI (2004), a adoção de um método formal de estimativa minimiza a subjetividade na elaboração desta. Os resultados esperados pela adoção de um método podem ser definidos por:

1) estabelecer uma forma técnica de estimar na organização;

2) possibilitar que não-especialistas também consigam estimar com um bom grau de certeza;

3) viabilizar a comparação e refino gradual do método de forma muito mais efetiva e objetiva que o aprendizado do especialista a partir de suas próprias estimativas;

4) registrar as premissas nas quais foram baseadas as estimativas, de modo que estas sejam repetíveis;

5) facilitar a defesa das estimativas quando questionados ou solicitados a reduzi-las, pois elas têm um racional por trás;

6) aumentar a previsibilidade, pois a um método e fórmula de estimativa é inerente a existência de limites para os resultados possíveis.

2.2.3.2 Refinando as Estimativas: Qualidade dos Dados e Histórico

O processo de estimativa tem o objetivo de modelar em um método a experiência do especialista, requerendo menor experiência para seu uso, pois esta estará refletida no próprio método, o que a tornará repetível. Um método de estimativa não é a verdade absoluta, mas constitui-se na melhor aproximação possível desta verdade em um determinado momento, devendo ser constante e incessantemente revisto e calibrado com base no histórico de estimativas que é coletado.

Um bom método de estimativa deve estar documentado, para ser repetido sempre da mesma forma por qualquer um na organização, ser simples para poder ser bem entendido e ser realimentado pelo histórico, visto que através dos dados históricos iremos convergir em nossas estimativas até o nível de qualidade exigido.

Na área de desenvolvimento de software, é possível citar Análise por Pontos de Função e Pontos de Caso de Uso como bons exemplos de métodos de estimativa. O primeiro é internacionalmente aceito e padronizado, mas exige mais informações e prática para seu uso.

Já Pontos de Caso de Uso é um método de estimativa que tem crescido muito nos últimos anos, por ser bastante simples e poder ser aplicado bem no início do projeto.

A adoção de um método de estimativa que modela a experiência do especialista de uma maneira simples e objetiva traz muitos benefícios à organização. Um método nivela o conhecimento sobre estimativas, possibilita a coleta de dados históricos e suas premissas para realimentar o método, resultando na diminuição do grau de incerteza nas estimativas realizadas.

2.2.3.3 Modelos de Estimativas

A literatura apresenta várias técnicas e modelos de estimativa para projetos de software, sendo que as mais conhecidas são apresentadas, resumidamente, a seguir.

Estimativa do Esforço PRESSMAN (1995) : é uma técnica de estimativa de esforço e custos de desenvolvimento de projeto considerando o número de profissionais-mês que executam as funções do projeto em cada nível de atividade do desenvolvimento (Análise, Projeto, Codificação, Teste).

Estimativa de Putnam PUTNAM (1978) : é um modelo dinâmico de múltiplas variáveis que pressupõe uma distribuição do esforço específico ao longo da existência de um projeto de desenvolvimento de software. O modelo relaciona o número de linhas de código ao tempo e esforço de desenvolvimento.

Pontos de Particularidade (*Feature Points*) JONES (1986) : é uma extensão da técnica Análise de Pontos-por-Função. A principal diferença é que a estimativa de Pontos de Particularidade acrescenta, ao cálculo dos Pontos-por-Função, o número de algoritmos do projeto. Este acréscimo permite uma estimativa que inclui uma análise da complexidade do projeto.

PSP (*Personal Software Process*) HUMPHREY (1995) : não é somente uma estimativa de custo, mas uma técnica derivada do SEI-CMM (*Software Engineering Institute – Capability Maturity Model*) que foi desenvolvida com a função de capacitação, melhoria e otimização do processo individual de trabalho. Essa técnica é dividida em sete etapas, sendo que nas etapas PSP0, PSP0.1 e PSP1 estima-se o tamanho e o tempo necessário para o desenvolvimento do produto.

PSM (*Pratical Software Measurement*) CHULANI (1999) : desenvolvido por profissionais da área de *Software Process Improvement* e formalizado através do padrão ISO/IEC 15939 – *Software Engineering – Software Measurement Process Framework*. Foi utilizada para a elaboração da *Process Área Measurement and Analysis* do CMMI. O PSM procura resolver dois problemas: especificar formalmente as medidas a serem utilizadas, através do Modelo de Informação, e como conduzir o processo de medição, através do Modelo de Processo.

Analisando-se os modelos e técnicas de estimativas apresentadas na literatura, verificou-se que a estimativa do esforço não atende aos requisitos definidos para este trabalho, por se basear no número de profissionais-mês. A estimativa de Putnam se baseia no número de linhas de código, medida que varia de acordo com a linguagem de programação. O PSP se trata de um processo completo, não tão voltado para estimativa, mas sim para o aprimoramento individual. A estimativa de Pontos de Particularidade utiliza como medida chave os pontos-por-função e, também, utiliza a complexidade dos algoritmos em seus cálculos. O PSM é um modelo para a estruturação da atividade de mensuração em um projeto de software como todo, que é o Modelo de Informação, que fornece um caminho para a seleção das medidas utilizadas, enquanto o Modelo de Processo serve de guia para a implementação do modelo.

A seguir serão vistos a Análise de Pontos de Função e o CoCoMo 81 de forma mais superficial, como uma forma de aproximação e embasamento para o estudo do CoCoMo II, que será o modelo a ser utilizado neste trabalho para elaboração do estudo de caso e da metodologia que será proposta para aplicação na empresa case.

2.3 Modelos

2.3.1 Análise de Pontos de Função (FPA)

Os conceitos gerais relacionados à técnica de Análise por Pontos de Função (FPA - *Function Point Analysis*), suas definições, identificações, exemplos e classificações, auxiliam o Gerente de Projetos, Analista de Sistemas ou Engenheiro de Software a estimar custo e prazo com maior precisão, através do dimensionamento de uma aplicação na perspectiva do usuário final, ou seja, considera como unidade de medida os aspectos externos do software ao invés das características técnicas da linguagem utilizada. A estimativa elaborada através de FPA retorna como resultado o tamanho do software em KDSI (*Kilo Delivered Source Instructions* – Milhares de Instruções Fontes Entregues).

De acordo com GONÇALVES (2004), FPA é uma "Técnica de dimensionamento de projetos de software, que considera como unidade de medida os aspectos externos do software, requisitados e visíveis ao usuário".

Segundo AZEVEDO (1998), a "FPA dimensiona software quantificando a funcionalidade que ele proporciona aos usuários.", e seus objetivos são: medir o que foi requisitado e recebido pelo usuário; medir independente da tecnologia utilizada para a implementação; prover uma métrica de medição para apoiar a análise de produtividade e qualidade; prover uma forma de estimar o tamanho do software e prover um fator de normalização para comparação de software.

A técnica pode ser aplicada tanto no dimensionamento de projetos de aplicações já implantadas quanto no dimensionamento de projetos de desenvolvimento ou manutenção de aplicações.

- **Dimensionamento de um projeto de desenvolvimento**

Utilizado para dimensionar em Pontos de Função o tamanho de um projeto de desenvolvimento de uma nova aplicação. Quantifica as funções solicitadas e entregues ao usuário pela nova aplicação, incluindo neste caso, as funções referentes ao processo de conversão de dados. Este valor, menos os Pontos por

Função associados às atividades de conversão de dados, torna-se o tamanho da aplicação, após sua implantação.

- **Dimensionamento de um projeto de manutenção**

Utilizado para dimensionar o tamanho de um projeto de manutenção em uma aplicação já existente. Mede todas as modificações de funções do usuário, incluindo, também, as funções providas pelo processo de conversão de dados. Ao final de um projeto de manutenção é necessário recalcular o tamanho da aplicação, para refletir as mudanças nas funções da aplicação.

- **Dimensionamento de uma aplicação**

Utilizado para dimensionar o tamanho real de uma aplicação em Pontos por Função. O valor encontrado representa a funcionalidade da aplicação do ponto de vista do usuário. Esse valor pode diferir da dimensão do projeto de desenvolvimento desta aplicação, uma vez que não inclui as funções do processo de conversão de dados.

- **Visão Geral de uma Aplicação**

Segundo PONTOS de FUNÇÃO (2004), "Uma aplicação, vista sob a ótica do usuário, é um conjunto de funções ou atividades do negócio que o beneficiam na realização de suas tarefas." Essas funções podem ser divididas em 5 agrupamentos:

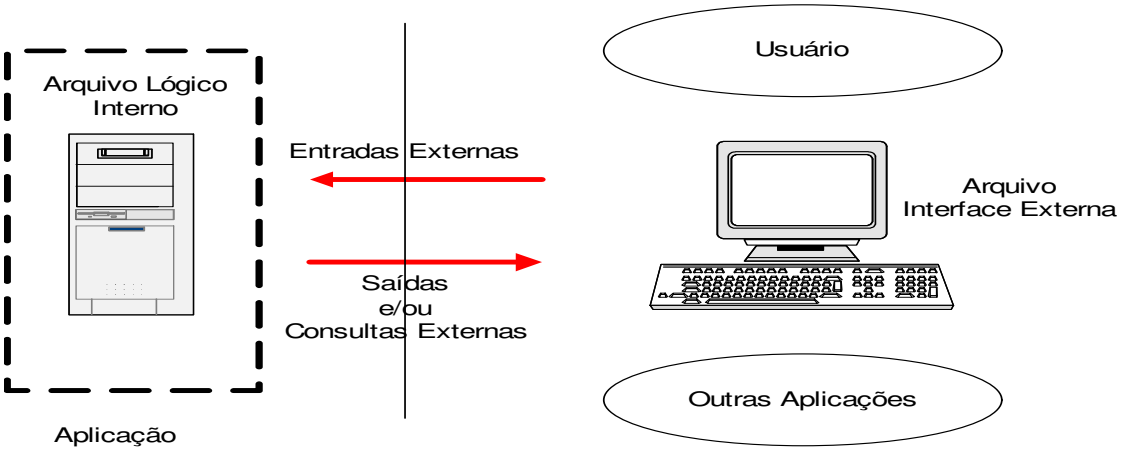


Figura 5 - Visão Geral de uma Aplicação

· ***Arquivo Lógico Interno***

Grupo lógico de dados do ponto de vista do usuário cuja manutenção é feita internamente pela aplicação.

Atribui-se um número de Pontos de Função para cada Arquivo Lógico, de acordo com a sua complexidade funcional relativa, apresentados a seguir: 7(sete) Pontos por Função (PF) se a complexidade for considerada simples; 10 (dez) PF se a complexidade for considerada média e 15 (quinze) PF se a complexidade for considerada complexa.

· ***Arquivo de Interface Externa***

Grupo lógico de dados utilizados na aplicação cuja manutenção pertence a outra aplicação.

Atribui-se um número de PF para cada Arquivo de Interface Externa de acordo com a complexidade funcional relativa apresentados a seguir: 5(cinco) PF se a complexidade for considerada simples; 7(sete) PF, se a complexidade for considerada média e 10 PF se a complexidade for considerada complexa.

· ***Entrada Externa***

Transações vindas diretamente do usuário que referenciam arquivos internos.

Atribui-se um número de PF para cada Entrada Externa, de acordo com a sua complexidade funcional relativa, apresentados a seguir: 3 (três) PF se a complexidade for considerada simples; 4(quatro) PF se a complexidade for considerada média e 6(seis) PF se a complexidade for considerada complexa.

· ***Saída Externa***

São dados extraídos da aplicação, tais como relatórios e mensagens do terminal de vídeo.

Atribui-se um número de PF para cada Saída Externa, de acordo com a sua complexidade funcional relativa, apresentados a seguir: 4 (quatro) PF se a complexidade for considerada simples; 5(cinco) PF se a complexidade for considerada média e 7(sete) PF se a complexidade for considerada complexa.

· ***Consulta Externa***

Combinação de uma entrada e uma saída de dados, isto é, uma requisição de dados que gera uma aquisição e exibição imediata de dados.

Atribui-se um número de PF para cada Consulta Externa, de acordo com a sua complexidade, apresentados a seguir: 3 (três) PF se complexidade for considerada simples; 4(quatro) PF se complexidade for considerada média e 6(seis) PF se complexidade for considerada complexa.

- **Cálculo de Pontos Por Função**

Segundo BFPUG (2004), determina-se os Pontos de Função de uma aplicação em três etapas de avaliação:

- ✓ Pontos de função Não Ajustados;
- ✓ Fator de Ajuste e
- ✓ Pontos de Função Ajustados.

  - **Pontos de Função Não Ajustados**

Refletem as funções específicas e mensuráveis do negócio, provida ao usuário pela aplicação. Para fazer o cálculo dos pontos de função não ajustados, uma função específica do usuário em uma aplicação é avaliada em termos do que é fornecido pela aplicação e não como é fornecido. Somente componentes solicitados pelo usuário e visíveis são contados.

  - **Fator de ajuste**

Representa a funcionalidade geral provida ao usuário pela aplicação. O fator de ajuste é calculado a partir de 14 características gerais dos sistemas apresentadas na Tabela 1, que permitem uma avaliação geral da funcionalidade da aplicação. As características gerais de um sistema são:

Tabela 1 – Níveis de influência para o fator de ajuste

Nível de influência	
1. Comunicação de Dados	8. Atualização On-line
2. Processamento de Dados Distribuídos	9. Processamento Complexo
3. Desempenho	10. Reutilização do código
4. Configuração Pesadamente Utilizada	11. Facilidade de Instalação
5. Taxa de Transação	12. Facilidade Operacional
6. Entrada de Dados On-line	13. Múltiplas Instalações
7. Eficiência do Usuário Final	14. Facilidades de Mudança

Fonte: BFPUG (2004)

Atribui-se um peso de 0 a 5 para cada característica, de acordo com o seu nível de influência (NI) na aplicação.

0 - nenhuma; 1 - mínima; 2 - moderada; 3 - média; 4 - significativa; 5 - grande.

- **Pontos de Função ajustados**

Reflete o fator de ajuste aplicado ao resultado apurado na primeira etapa. Para fazer o cálculo dos PF ajustados, e o total de PF da aplicação, multiplica-se o fator de ajuste pela quantidade de PF não ajustados; após esse cálculo, busca-se na tabela correspondente o peso que a linguagem a ser utilizada possui, e multiplica-se pelo valor dos pontos de função ajustados. Desta forma encontra-se o número de instruções fontes da aplicação, e dividindo-se por 1000, encontra-se o número de KDSI.

2.3.1.1 Como contar linhas de código

De acordo com COCOMO MANUAL (2004), existem algumas maneiras para contar e estimar linhas de código. A melhor fonte é possuir dados históricos. Assim podem haver dados que se converterão em Pontos de Função, por exemplo, componentes, para estimar linhas do código.

Linhas de código geralmente significam as instruções que são relativamente parte integrante do software, ou seja, excluir da contagem o que não será entregue, o que não faz parte da sustentação do software, tal como drivers de teste e comentários.

No CoCoMo II a indicação lógica da fonte foi escolhida como a linha padrão do código. Definir exatamente uma linha do código é difícil devido às diferenças conceituais envolvidas na contabilidade para declarações executáveis das indicações e dos dados em linguagens diferentes. O objetivo é medir a quantidade de trabalho intelectual posta no desenvolvimento do software, mas as dificuldades levantam-se ao tentar definir medidas consistentes através das linguagens diferentes. Para minimizar estes problemas, uma lista de verificação da definição de linha código pode ser encontrada em SEI (2004) para uma indicação lógica da fonte que é usada para definir uma linha do código.

2.3.2 *Constructive Cost Model* - CoCoMo 81

O *Constructive Cost Model*, (CoCoMo) , é um método que busca medir esforço, prazo, tamanho de equipe e custo necessário para o desenvolvimento do software, desde que se tenha a dimensão do mesmo., através de um modelo de estimativa de tamanho de software, como PFA.

O modelo CoCoMo foi proposto por BOEHM (1981), tendo sido construído e calibrado inicialmente a partir de informação de um número considerável de projetos concluídos, em torno de 83. Afirma-se que a sua utilização tem permitido estimativas com um erro inferior a 20% em cerca de 70% dos projetos.

O CoCoMo 81 considera três modos de desenvolvimento:

• Modo orgânico - aplicável a ambientes de desenvolvimento estáveis, com pouca inovação e a projetos com equipes de dimensão relativamente pequena;

• Modo semidestacado - aplicável a projetos com características entre o modo orgânico e o embutido

• Modo embutido - aplicável no desenvolvimento de sistemas complexos embutidos em hardware, com muita inovação, com restrições severas e/ou com requisitos muito voláteis.

O CoCoMo considera ainda três "estágios" do modelo, à medida que vai sendo introduzido mais detalhamento:

Modelo Básico

É um modelo estático de valor simples que computa o esforço e o custo de desenvolvimento de software como uma função do tamanho de programa expresso em linhas de código estimadas, de acordo com ABREU (1998a).

Modelo Intermediário

A fase seguinte de sofisticação do modelo; corresponde a considerar a influência de um conjunto de vários fatores, relativos quer ao sistema a produzir (produto) propriamente dito, quer ao suporte computacional (tecnologia utilizada), fator humano e organização do processo de desenvolvimento de software. A influência destes fatores, em número de 15 no modelo originalmente proposto, deve ser avaliada numa escala discreta e ponderada.

Modelo Completo

Na sua versão mais completa, o modelo CoCoMo introduz aspectos adicionais como a decomposição de um sistema de grande dimensão em subsistemas. Outros aspectos correspondem à distribuição das estimativas de

esforço e de prazo por fase e por atividade e à influência diferenciada de cada fator influenciador do custo por fase. Para mais detalhe sobre este modelo consulte ABREU (1998b).

As estimativas do CoCoMo são mais objetivas e repetíveis do que as feitas pelos métodos que confiam em modelos proprietários, tendo como vantagem ainda o fato de ser um modelo aberto. Pode ainda refletir um ambiente de desenvolvimento do software e produzir estimativas precisas.

- **Direcionadores de Custo**

São fatores de ajuste do esforço, baseados em 15 atributos direcionadores do custo que estão divididos em quatro categorias (produto, computador, pessoal e projeto).

Cada direcionador de custo é estimado com base em uma escala de seis pontos, variando de baixa para alta importância, conforme a Tabela 2. O produto de todos os multiplicadores de esforço é o Fator de Ajustamento de Esforço ou EAF (*Effort Adjustment Factor*).

Tabela 2 – Direcionadores de Custo CoCoMo 81

Direcionador de Custo	***Descrição***	***Muito Baixo***	***Baixo***	***Nominal***	***Alto***	***Muito Alto***	***Extra Alto***
Produto							
RELY	Confiabilidade do software	0,75	0,88	1,00	1,15	1,40	-
DATA	Tamanho do banco de dados	-	0,94	1,00	1,08	1,16	-
CPLX	Complexidade do produto	0,70	0,85	1,00	1,15	1,30	1,65
Computador							
TIME	Restrição de tempo de execução	-	-	1,00	1,11	1,30	1,66
STOR	Restrição memória principal	-	-	1,00	1,06	1,21	1,56
VIRT	Mudanças do ambiente	-	0,87	1,00	1,15	1,30	-
TURN	Velocidade processam. Computador	-	0,87	1,00	1,07	1,15	-
Pessoal							
ACAP	Capacidade do analista	1,46	1,19	1,00	0,86	0,71	-
AEXP	Experiência na aplicação	1,29	1,13	1,00	0,91	0,82	-
PCAP	Capacidade do programador	1,42	1,17	1,00	0,86	0,70	-
VEXP	Experiência com hardware	1,21	1,10	1,00	0,90	-	-
LEXP	Experiência na linguagem	1,14	1,07	1,00	0,95	-	-
Projeto							
MODP	Práticas de programação Modernas	1,24	1,10	1,00	0,91	0,82	-
TOOL	Ferramentas de Software	1,24	1,10	1,00	0,91	0,83	-
SCED	Cronograma de desenvolvimento	1,23	1,08	1,00	1,04	1,10	-

Fonte: BOEHM (1981)

2.3.3 *Constructive Cost Model* - CoCoMo II

Devido à idade dos projetos que embasaram o modelo anterior, assim como sua incapacidade de lidar com ciclos iterativos e com a utilização de componentes *Commercial-Off-The-Shelf* (COTS)[5], o CoCoMo 81 é atualmente considerado obsoleto, tendo sido substituído pela sua versão II, publicada inicialmente em 2000 e vista em AGUIAR (2004).

O CoCoMo II, assim como o seu predecessor, busca medir esforço, prazo, tamanho e custo, necessários para o desenvolvimento de software, desde que se tenha, como premissa, a dimensão do mesmo. Para o cálculo do custo deve-se conhecer o prazo e equipe de trabalho, para então chegar ao valor, sendo que para definir o tamanho do programa, torna-se necessário que se caracterize que medida será adotada (linhas de código, pontos por função ou pontos por caso de uso).

Este modelo para a mensuração de esforços, prazos e custos, é condizente com as tecnologias dos anos 90, trazendo adequações às características descritas desde que o CoCoMo original fora descrito em TRINDADE (1999), e pelo fato que normalmente nos faltam dados históricos que permitam a utilização de uma abordagem mais simplificada, torna-se interessante a utilização deste modelo paramétrico (baseado em parâmetros).

O CoCoMo II estima o custo e tempo baseado em pessoas/mês e meses, respectivamente, para a determinação do *baseline*[6] de exigências de um produto para a conclusão de uma atividade. O modelo ainda prevê um adicional de 20% ao tempo computado, como margem de erro (análise de risco).

A definição inicial do CoCoMo II vem sendo refinada com pesquisas e dados adicionais sobre projetos que são coletados e analisados pela equipe da USC. Podemos então definir os objetivos primários do CoCoMo II como sendo:

- Um modelo de estimativa de custo para o desenvolvimento de software de acordo com o modelo de ciclo de vida de software, vista em JONES (1986), e as práticas de desenvolvimento da última década.
- Criar ferramentas de suporte capazes de fornecer melhoramentos do modelo, através de manutenção de informações sobre o desenvolvimento de software em uma estrutura de base de dados.
- Fornecer um *framework* analítico, e um conjunto de ferramentas e técnicas para avaliação dos efeitos de melhoria na tecnologia e nos custos despendidos no ciclo de vida de desenvolvimento de software.

Estes objetivos suportam as necessidades primárias da estimativa de custo, realizadas pela equipe da USC. Como prioridade, essas necessidades devem

---

[5] *Comercial-off-the-shelf* (COTS) são componentes de software prontos para utilização, de terceiros, comercialmente disponíveis, e que se tornam importantes durante a criação do novo software, devendo ser utilizados preferencialmente durante a fase de pré-desenvolvimento do produto (software), segundo BOEHM (2000).

[6] Uma baseline é um conjunto de produtos aceitos e controlados, e que serão utilizados em atividades posteriores à sua aceitação, segundo LEITE (1997)

fornecer suporte para a etapa de planejamento de projetos, equipe de desenvolvimento necessária, preparação inicial do projeto, replanejamento, negociação de contratos, avaliações de propostas, necessidades e níveis de recursos (tecnológicos e humanos), exploração dos conceitos e avaliação de design. Para cada uma dessas necessidades, o CoCoMo II fornece mais atualizações para suporte do que o seu predecessor CoCoMo 81 e Ada CoCoMo, segundo ABREU (1998b).

Para obter o custo por fase, dado que o CoCoMo II gera o esforço/fase, é necessário gerar o custo médio por Pessoa-mês, calculado como custo e total para projeto inteiro (excluindo exigências) / Esforço Total em Pessoas-mês (excluindo exigências). A este cálculo é assumido um adicional de 7% do Esforço Total.

O custo do projeto inteiro é a soma dos custos das fases individuais. Considerando que estas são só estimativas, são usadas tabelas para exibir o custo atual, gerando apenas uma estimativa Otimista e Pessimista para o custo do projeto inteiro.

2.3.3.1 Calibração do Modelo

A calibração do modelo para um ambiente consiste no ajuste da variável "**A"**, que indica o percentual de reusode código e será vista na seção 3.5 deste trabalho (Medições pelo método CoCoMo II) da fórmula de esforço. O objetivo da calibração é obter a distribuição da produtividade e atividade de desenvolvimento para um ambiente específico. O modelo CoCoMo II foi originalmente calibrado com dados de 161 projetos. Os mesmos foram selecionados entre 2000 projetos candidatos. Para cada um dos 161 projetos escolhidos foram realizadas entrevistas e visitas, a fim de garantir a consistência das definições e suposições do modelo. O modelo nominal vem calibrado para esses projetos, cuja natureza pode diferir daquele que se deseja estimar.

Embora o CoCoMo II possa ser executado com os parâmetros nominais, sua correta utilização pressupõe a calibração para o ambiente-alvo. Com a calibração em projetos do mesmo ramo é possível modificar o valor das constantes das fórmulas padrões. Na ausência de dados históricos disponíveis para o ambiente-alvo em questão, devem ser selecionados projetos equivalentes para efetuar a calibração. Os dados históricos selecionados devem ser validados antes de sua utilização, calculando os coeficientes calibrados e, posteriormente, verificando se a diferença percentual (estimado – real)/estimado encontra-se compatível com o nível de erro pretendido para as estimativas. Devido ao seu impacto exponencial, não é recomendável calibrar o modelo quando houver menos de 10 projetos disponíveis para a calibração, conforme BOEHM (2000).

2.3.3.2 Fatores de Escala (*Scale Factors*)

Os modelos de estimativa de custo de software possuem um fator exponencial para considerar os gastos (despesas) e economias relativas as escala encontradas em projetos de software de tamanhos distintos. Para determinar se o projeto apresenta gastos ou economias, uma variável é utilizada como expoente da equação de esforço, **b = 0,91 + 0,01 x ΣSFj**. A equação como um todo será vista com mais detalhes na seção 3.5 deste trabalho (Medições utilizando o Modelo CoCoMo II).

A partir do valor de "**b"**, pode-se capturar os efeitos do projeto e chegar as seguintes premissas:

b < 1,0: O projeto apresenta economia de escala. Se por ventura o tamanho do produto dobra, o esforço relativo ao projeto é menor que o dobro. A produtividade do projeto aumenta à medida que aumenta o tamanho do produto. Podem-se alcançar algumas economias de escala do projeto com ferramentas de projeto específicas. Para projetos pequenos, fixar custos de saída, tais como reuniões periódicas e ainda relatórios administrativos, são a rigor uma fonte de economia.

b = 1,0: As economias e gastos estão em um nível de equilíbrio. Este modelo linear é utilizado a rigor para a estimativa de custo de projetos pequenos.

b > 1,0: O projeto apresenta gastos de escala. Isto se deve normalmente a dois fatores principais: o crescimento do gasto em comunicações e o gasto em crescimento da integração de um grande sistema. Os projetos maiores alocam mais recursos humanos, e conseqüentemente uma maior comunicação e iteração interpessoal produzindo despesas. Integrar um produto pequeno como parte de um maior não requer apenas esforço de desenvolver o produto pequeno como também a despesa adicional de esforço para projetar, manter, integrar e testar suas interfaces com o resto do produto.

O expoente "**b"** é obtido através dos fatores de escala. A seleção dos fatores de escala é embasada na razão de que são recursos significativos de variação exponencial de esforço ou variação da produtividade do projeto. Cada fator tem um intervalo de níveis de valores que vão desde Muito Baixo até Extra Alto, conforme é apresentado na Tabela 3. Cada nível de valores possui um peso, SF, e o valor específico do peso é chamado Fator de Escala. Um fator de escala de um projeto é calculado realizando o somatório de todos os fatores e é utilizado para determinar o expoente "**b"**.

Tabela 3 – Fatores de escala do modelo CoCoMo II

Fatores de Escala (SF)	Muito Baixo	Baixo	Nominal	Alto	Muito Alto	Extra Alto
PREC	Completa-mente sem Precedentes	Praticamente sem Precedentes	Quase sem Precedentes	Algo Familiar	Muito Familiar	Completa-mente Familiar
FLEX	Rigoroso	Relaxamentos Ocasionais	Relaxamentos Periódicos	Conformidade geral	Algo de conformidade	Metas gerais
RESL	Pouca (20%)	Alguma (40%)	A rigor (60%)	Geralmente (75%)	Em sua maior parte (90%)	Por completo (100%)
TEAM	Iterações muito complicadas	Alguma dificuldade nas iterações	Iterações basicamente cooperativas	Bastante cooperativos	Altamente cooperativos	Iterações completas
PMAT	Peso médio de respostas "Sim" para o questionário de maturidade CMM					

Fonte: COCOMO MANUAL (2004)

- **PREC e FLEX:** Precedência e Flexibilidade de Desenvolvimento

Estes dois fatores de escala capturam em grande parte as diferenças entre os modos Orgânico, Semidestacado e Embutido do modelo original CoCoMo 81.

- **RESL:** Arquitetura/Resolução de Riscos

Este fator combina dois fatores de medida do modelo AdaCoCoMo "Minúcia do projeto pela revisão do projeto do produto (PDR)" e "Eliminação de riscos por PDR".

- **TEAM:** Coesão da Equipe

Este fator explica os recursos de turbulência e entropia do projeto devido a dificuldades na sincronização dos implicados no projeto, usuários, clientes, desenvolvedores, responsáveis pelo suporte e assim por diante. Essas dificuldades podem aparecer pelas diferentes culturas e objetivos dos implicados; dificuldades em concilias objetivos e a falta de experiência e de familiaridade dos implicados em trabalhar como uma equipe.

- **PMAT:** Maturidade do Processo

O procedimento para determinar PMAT é obtido através do modelo *Capability Maturity Model* (CMM), do *Software Engineering Instiute* (SEI). O período de tempo para medir a maturidade do processo é o momento em que o projeto se inicia. Existem duas formas de medir a maturidade do processo. A primeira toma o resultado de uma avaliação organizada embasada no CMM. A segunda está organizada em função das 18 áreas principais de processo (KPA's – *Key Process Area*) do modelo CMM. O procedimento para determinar PMAT é decidir o percentual de conformidade para cada uma das KPA's. O nível embasado em meta (objetivo) de conformidade é determinado realizando a média das metas de cada área principal do processo. Para maiores informações sobre as KPA's consulte SEI (2004). Pesquisas realizadas em diversas bibliografias não ofereceram subsídios suficientes para avaliar adaptações ou os impactos do CMMI sobre o modelo CoCoMo II.

2.3.3.3 Multiplicadores de Esforço (*Effort Multipliers*)

Os drivers de custo são utilizados para capturar características do desenvolvimento de software que afetam o esforço para completar o projeto. Os drivers de custo possuem um nível de medida que expressa o impacto do driver no esforço de desenvolvimento. Esses valores variam desde Extra Baixo até Extra Alto. Para o propósito de análise quantitativa, o nível de medida de cada driver de custo possui um peso associado. A este peso se dá o nome de Multiplicador de Esforço (EM). A medida atribuída a um driver de custo é 1.0 e o nível de medida associado com este peso é chamado de nominal. Se um nível de medida produz mais esforço de desenvolvimento de software, então seu correspondente EM possui valor acima de 1.0. Reciprocamente se o nível de medida reduz o esforço, então o correspondente EM possui valor abaixo de 1.0.

Os EM são utilizados para ajustar o esforço Pessoas/Mês nominal. Existem 7 multiplicadores de esforço para o modelo *Early Design* e 17 para o modelo *Post-*

Architecture, e a sua influência dentro da fórmula de esforço se dá da seguinte forma: **PM = A X (Size)^b X ΠEMi[7].**

Os drivers de custo do modelo *Early Design* são obtidos através da combinação dos drivers de custo do modelo *Post-Architecture*, como será apresentado na Tabela 4. Sempre que uma avaliação de um driver de custo estiver entre níveis, deve-se arredondar para o valor mais próximo do Nominal. Por exemplo, se o valor de um driver de custo estiver entre Muito Baixo e Baixo, deve ser selecionado o nível Baixo, por estar mais próximo de Nominal.

Tabela 4 – Multiplicadores de Esforço dos Modelos *Early Design* e *Post-Architecture*

Drivers de Custo do modelo Early Design	Drivers de custo combinados, homólogos do modelo Post-Architecture
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PERS	ACAP, PCAP, PCON
PREX	AEXP, PEXP, LTEX
FCIL	TOOL, SITE
SCED	SCED

Fonte: COCOMO MANUAL (2004)

- **Drivers de custo para o modelo *Early Design***
    - **RCPX:** Confiabilidade e Complexidade do Produto
    - **RUSE:** Reutilização Necessária
    - **PDIF:** Dificuldade da Plataforma
    - **PERS:** Capacidade do Pessoal
    - **PREX :** Experiência do Pessoal
    - **FCIL:** Facilidades
    - **SCED:** Planificação Temporal
- **Drivers de custo para o modelo *Post-Architecture***

Os 17 drivers de custo estão divididos entre os contextos produto, plataforma, pessoal e projeto e são usados para obter características do desenvolvimento do software que afetam o esforço para concluir o projeto.

O contexto de produto se embasa em informações referentes à ênfase na confiabilidade do software, sua documentação durante todo o ciclo de vida do produto, sua complexidade e a medida (ou tamanho) da base de dados.

O contexto plataforma refere-se à complexidade do hardware e à infra-estrutura do software da máquina a ser utilizada (também chamada de máquina virtual), para desenvolvimento e execução da aplicação desenvolvida. Devem ser

[7] Π significa produtório, que simboliza o total de diversas mutlplicações.

considerados alguns fatores adicionais da plataforma, como a distribuição, paralelismo e operações em tempo real.

O contexto pessoal combina as informações que trata das habilidades pessoais dos integrantes do projeto, como a sua experiência, experiência na plataforma utilizada, por exemplo o sistema operacional, e a experiência na linguagem e ferramentas adjuntas utilizadas no desenvolvimento do projeto de software, e também de fatores que abrangem o nível de rotatividade da equipe.

O contexto de projeto abrange os fatores referentes ao uso das ferramentas de software, ou seja, o tipo de ferramenta utilizada para o desenvolvimento do projeto de software, quanto ao tipo de tecnologia para comunicação que será necessária no projeto e informações sobre o calendário do projeto, que indicam as restrições de tempo impostas à equipe de projeto que desenvolve o software.

Contexto Produto

- **RELY: Confiabilidade Requerida do Software**

  Esta é a medida de até que ponto o software deve realizar sua função esperada durante um período de tempo. Se o efeito de um fracasso é apenas um ligeiro mal-estar então RELY é baixo. Se uma falha do sistema arriscar vidas humanas, então RELY é muito alto.

- **DATA: Medida de Volume de Dados**

  Esta medida tenta capturar o quanto afetam no desenvolvimento do produto algumas requisições de volumes grandes de dados. A medida se determina calculando D/P. A razão pela qual é importante considerar o tamanho da base de dados é pelo esforço necessário para gerar dados de teste que serão utilizados para executar o programa.

  D – *Database Size* (*Bytes*) / P – *Program Size* (SLOC)

  O multiplicador DATA se define como baixo se D/P é menor que 10, e muito alto se é maior que 1000.

- **CPLX: Complexidade do Produto**

  A complexidade se decide em 5 (cinco) áreas: Controle de Funcionamento, Funcionamento Computacional, Funcionamento de Dispositivos Dependentes, Funcionamento do Setor de Dados e Funcionamento do Gestor de Interface de Usuário. Será selecionada a área ou combinação que caracteriza o produto ou um subsistema do produto. A medida de complexidade é a média subjetiva destas áreas.

- **RUSE: Reutilização Requerida**

  Este driver de custo explica o esforço adicional necessário para construir componentes pensados para serem reutilizados em projetos presentes ou futuros. Este esforço consiste em criar um projeto mais genérico do software, documentação mais detalhada e testes mais extensos, para assegurar que os componentes estarão prontos para que sejam utilizados em outras aplicações.

- **DOCU: Documentação Associada às Necessidades do Ciclo de Vida**

  Vários modelos de custo de software possuem um driver de custo para o nível de documentação necessária. No CoCoMo II a escala de medida para este driver de custo se avalia em termos de adequação da documentação do projeto às necessidades de seu ciclo de vida. A escala de valores varia desde muito baixo (muitas necessidades do ciclo de vida sem cobertura) até muito alto (excesso para as necessidades do ciclo de vida).

Contexto Plataforma

- **TIME: Restrição do Tempo de Execução**

  Esta é uma medida da restrição do tempo de execução imposta em um sistema de software. As medidas são expressas em termos de porcentagem de tempo de execução disponível que é esperado que seja usado pelo subsistema ou sistema que consome o recurso de tempo de execução. Os valores vão desde o nominal, menos de 50% de recursos de execução utilizados, até extra alto, 95% do recurso do tempo de execução consumido.

- **STOR: Restrição de Armazenamento Principal**

  Esta medida representa o grau de restrição de armazenamento principal imposto a um sistema ou subsistema de software. Dado o notável aumento no tempo de execução disponível do processador e do armazenamento principal, pode ser questionado se essas variáveis de restrição ainda são pertinentes. Os valores vão desde o nominal, menos de 50%, até extra alto, 95%.

- **PVOL: Volatilidade da Plataforma**

  O termo “Plataforma” aqui é usado para significar a complexidade do *Hardware* e do *Software* (Sistema Operacional, Sistema Gerenciador de Banco de Dados), que o produto software necessita para realizar suas tarefas. Se o software a desenvolver é um sistema operacional, então a plataforma é o hardware do servidor. Se estiver sendo desenvolvido um Gerenciador de Banco de Dados, então as plataformas são o *hardware* e o sistema operacional. Se estiver sendo desenvolvido um browser de texto pela intranet, então a plataforma é a rede, o *hardware* do servidor, o sistema operacional e os repositórios de informações distribuídos. A plataforma inclui qualquer compilador que suporte o desenvolvimento do sistema software. Os valores vão desde baixo, onde a cada 12 meses há uma alteração importante, até muito alto, onde existe uma alteração importante a cada 2 semanas.

Contexto Pessoal

- **ACAP: Habilidade do Analista**

  Os analistas são o pessoal que trabalha nos requisitos de projeto de alto nível e no projeto detalhado. Os atributos principais que devem ser considerados nesta medida são a habilidade para realizar análise e projeto, a eficiência e minúcia, e habilidade para se comunicar e

cooperar. A medida não deve considerar o nível de experiência do analista; para avaliar este nível se utiliza o driver **AEXP**. Os analistas podem possuir uma avaliação que varia desde muito baixo, 15% a muito alto, 90% do nível de habilidade.

- **PCAP: Habilidade do Programador**

  As tendências atuais continuam dando ênfase à capacidade dos analistas. Entretanto, o crescente papel dos pacotes complexos COTS, e a relevante influência associada à capacidade dos programadores para tratar com esses pacotes, indicam também uma tendência a dar maior importância à capacidade do programador. Esta avaliação deve ser embasada na capacidade dos programadores como uma equipe, mais do que individualmente. A habilidade do programador não deve ser considerada aqui, este fator é avaliado com **AEXP.** Os valores variam de muito baixo, 15%, até muito alto, 90% do nível de habilidade.

- **AEXP: Experiência nas Aplicações**

  Essa medida depende do nível experiência em aplicações da equipe de projeto ao desenvolver sistemas ou subsistemas de software. Os valores definem termos de nível de experiência da equipe de projeto no tipo de aplicações. Um valor muito baixo para experiência em aplicações é menor que 2 meses. Um valor muito alto para experiência está em torno de 6 anos ou mais.

- **PEXP: Experiência na Plataforma**

  O modelo *Post-Archtecture* amplia a influência em produtividade de **PEXP**, reconhecendo a importância de entender o uso de plataformas mais poderosas, incluindo mais interfaces gráficas de usuários, redes e capacidade de processamento distribuído.

- **LTEX: Experiência na Ferramenta e na Linguagem**

  Esta é uma medida de nível de experiência na linguagem de programação e na ferramenta de software da equipe de projeto que desenvolve o sistema ou subsistema de software. O desenvolvimento de software inclui o uso de ferramentas que realizam a representação dos requisitos e projeto, análise, gestão da configuração, origem dos documentos, gestão das bibliotecas, estilo de programas e estruturas, verificação de consistência e etc. Além da experiência programando em uma linguagem específica, as ferramentas que dão suporte também influem no tempo de desenvolvimento. Possuir uma experiência de menos de 2 meses implica em um valor muito baixo, e se for de 6 anos ou mais, implica em um valor muito alto.

- **PCON: Continuidade do Pessoal**

  A escala de valores PCON é avaliada em termos de movimentação e rotatividade do pessoal do projeto anualmente, variando de muito baixo a muito alto.

Contexto Projeto

- **TOOL: Utilização das Ferramentas de Software**

  As ferramentas de software têm melhorado significativamente desde os 70 projetos utilizados inicialmente para calibrar o modelo CoCoMo II. Os valores para a ferramenta vão desde edição e código simples, muito baixo, até ferramentas integradas com a gestão do ciclo de vida, muito alto.

- **SITE: Desenvolvimento Multilocal**

  Dada a freqüência crescente de desenvolvimento em vários lugares e indicações de que os efeitos do desenvolvimento multilocal são significantes, foi adicionado ao CoCoMo II o driver de custo SITE. Determinar a medida do driver de custo inclui o cálculo e a medida de dois fatores: localização do lugar (desde totalmente localizado até distribuição internacional) e suporte de comunicação (desde correio convencional e algum acesso telefônico até multimídia totalmente interativo).

- **SCED: Calendário de Desenvolvimento Requerido**

  Este valor mensura as restrições de horário impostas à equipe de projeto que desenvolve o software. Os valores são definidos em termos de porcentagem de aceleração ou alargamento sobre o calendário nominal para um projeto que requer uma quantidade de esforço dada. Os calendários acelerados tendem a produzir mais esforço nas fases mais adiantadas do desenvolvimento porque se deixa para solucionar uma quantidade maior de problemas devido à falta de tempo resolvê-los anteriormente. Uma compactação do calendário de aproximadamente 74% é avaliada como muito baixo. Uma aceleração do calendário produz mais esforço nas fases iniciais do desenvolvimento, onde há mais tempo para planejar, realizar especificações e validar de forma mais detalhada. Uma aceleração de aproximadamente 160% é avaliada como muito alta.

2.3.3.4 Inovações do Modelo CoCoMo II

Além da capacidade de lidar com a utilização de componentes *Commercial-Off-The-Shelf* (COTS), o modelo CoCoMo II foi desenvolvido para ser utilizado por projetos que estejam empregando os processos do tipo espiral ou *waterfall*. Para que sejam razoavelmente compatíveis com o modelo, este tipo de desenvolvimento necessita ser fortemente dirigido a riscos, para evitar que haja alta quantia de retrabalho, que não esteja incluído na estimativa realizada pelo modelo, de acordo com COCOMO MANUAL (2004).

O CoCoMo II inclui atualizações significativas em seu modelo, como disponibilizar análises de melhoria de processo, aquisição de ferramentas, alterações na arquitetura, decisões entre desenvolvimento e aquisição de componentes, assim como diversas outras decisões sobre retorno do investimento e acompanhamento da evolução da tecnologia dos algoritmos, servindo de base a estimativas mais modernas.

A implementação do modelo espiral utilizada pelo modelo CoCoMo II também necessita adicionar um aspecto: a definição de um conjunto de marcos, que podem

servir como final entre os pontos os quais o CoCoMo II utiliza para os atuais cálculos das estimativas para este tipo de implementação. O resultado são os Pontos Âncoras: Objetivos do Ciclo de Vida (*Life Cycle Objectives* – LCO), Arquitetura do Ciclo de Vida (*Life Cycle Architecture* – LCA) e Capacidade Operacional Inicial (*Initial Operational Capability* – IOC). Estes marcos asseguram a compatibilidade do CoCoMo II com o RUP (visto na seção 2.2.2.4 deste trabalho) e são uma boa maneira de adaptar os pontos dos ciclos de vida a projetos encerrados para ter um embasamento de como utilizá-los projetos novos ou em andamento. Os marcos LCO e LCA envolvem concorrências anteriores à elaboração e desenvolvimento seqüencial do conceito de sistemas operacionais, necessidades, arquitetura, protótipo e planejamento do ciclo de vida.

- LCO – *Lyfe Cycle Objectives* – ponto no qual é escolhida uma possível arquitetura para o projeto (não sendo necessariamente aquela que será de fato escolhida), ocorre ao final da Inception, apropriado para projetos que estejam sendo estimados pelo modelo Early Design.
- LCA – *Life Cycle Architecture* – definição da arquitetura que será utilizada no desenvolvimento do projeto.
- IOC – *Initial Operational Capability* – ponto no qual é concluído o desenvolvimento do software, estando o sistema pronto para entrega e testes finais. Este marco é mais adequado para projetos onde já se tenham todos os requisitos e especificações concluídas, sua arquitetura definida, ou seja, é apropriado para projetos que estejam sendo estimados pelo modelo *Post- Architecture*.

A figura 6 apresenta o relacionamento das fases do modelo *Waterfall* e RUP e a probabilidade do modelo CoCoMo II ser utilizado em estimativas. Os marcos possuem algumas variações próprias para serem utilizadas na distribuição de esforço e estimativas entre os dois tipos de metodologias.

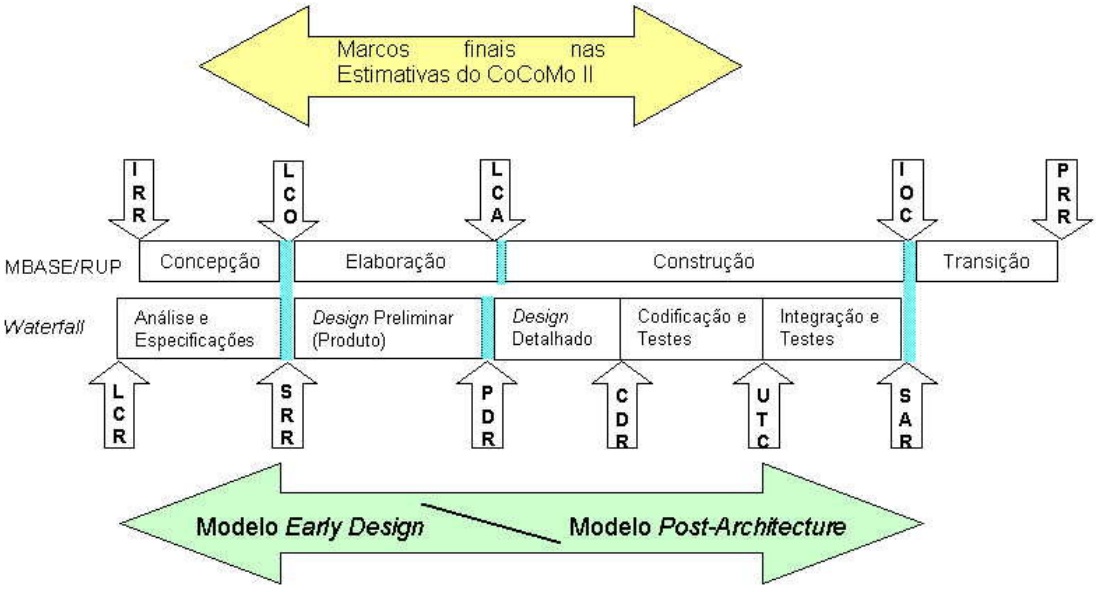


Figura 6 - Fases dos ciclos de vida – extraído de COCOMO MANUAL (2004)

- **Modelo de Prototipação de Software em Cascata (Tradicional, *Waterfall* e Espiral – Incremental)**
    - **Proposta para estimar e planejar iterações**

Esta proposta se apóia em um ciclo de vida iterativo incremental, que compreende duas etapas, sendo que a primeira é determinar casos de uso ou funções a serem realizadas na iteração, e posteriormente, estimar o esforço para cada iteração.

- **Determinação do esforço para iterações**

Para descobrir o esforço causado em cada iteração, em primeiro lugar é preciso estimar cada um dos multiplicadores de esforço do modelo CoCoMo II, correspondentes a cada iteração.

A partir desses valores e conhecendo-se os pontos de função sem ajustar correspondentes a cada iteração, se obtêm o esforço em meses de cada uma das iterações, e, com esse valor, é possível determinar o tempo, pessoas e custo necessários para concluir a iteração, e por conseqüência, de todo o projeto.

É importante ressaltar que o contexto do projeto pode ser alterado ao passar de uma iteração para outra (seja através do conhecimento da plataforma de desenvolvimento, ou integração da equipe de desenvolvimento, etc), sendo que pode ser necessário realizar novamente a estimativa do esforço requerido para as iterações subseqüentes, revisando as especificações em caso de alteração de requisitos, ou recalculando os multiplicadores de esforço no caso de alterações contextuais ou organizacionais que afetem algum dos drivers de custo. As figuras 7, 8 e 9 apresentam alguns ciclos iterativos que podem ser medidos pelo CoCoMo II.

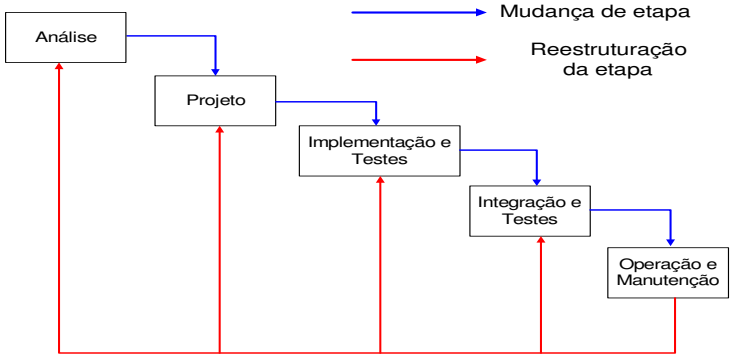


Figura 7 - Modelo de prototipação Cascata

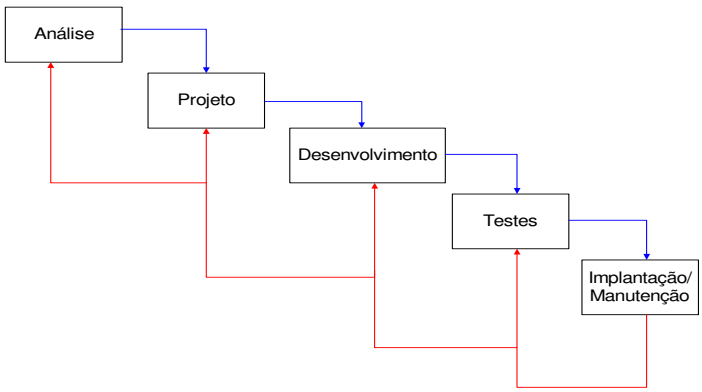


Figura 8 - Modelo de prototipação *Waterfall*

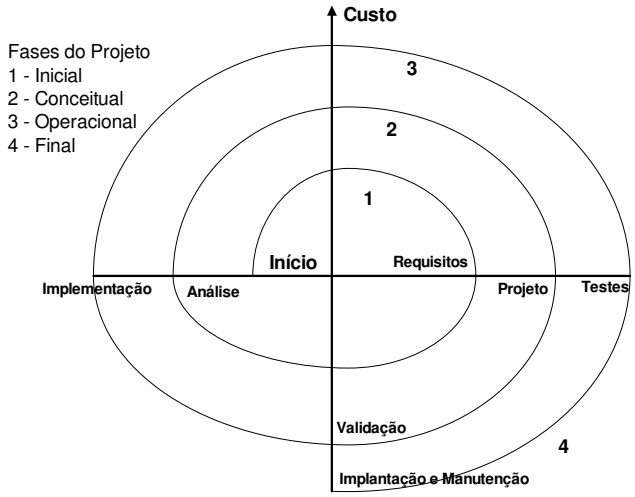


Figura 9 - Modelo de Prototipação Espiral

- **Inclusão de Pontos de Objeto / Pontos de Caso de Uso**

Uma evolução significativa aplicada ao modelo CoCoMo II é a possibilidade de realizar estimativas para softwares Orientados a Objetos (OO), dada a crescente utilização deste paradigma de desenvolvimento de software. Para isto, o método utiliza-se do conceito de Pontos Objeto ou Pontos de Caso de Uso.

Os Pontos Objeto são derivados da estrutura de classes, de mensagens e de processos ou casos de uso e ponderados pelos fatores de ajuste. Utilizam como estimativa inicial de tamanho a contagem do número de telas, relatórios e componentes de terceira geração que serão usados na aplicação. Cada objeto na aplicação contém uma classificação, de acordo com o número de instâncias[8] que possui. Já Pontos de Caso de Uso tratam classes como arquivos e métodos como transações, fazendo uma clara alusão aos Pontos de Função. Um método alternativo para a estimativa, com base em casos de uso, prevê a quantidade de pontos de função em cada caso de uso, com o posterior uso da técnica de Pontos de Função, como ilustra a figura 10.

[8] Uma instância é a criação, dentro da aplicação, de um novo objeto, ou classe, que contém as mesmas características, métodos e atributos do objeto ou classe original.

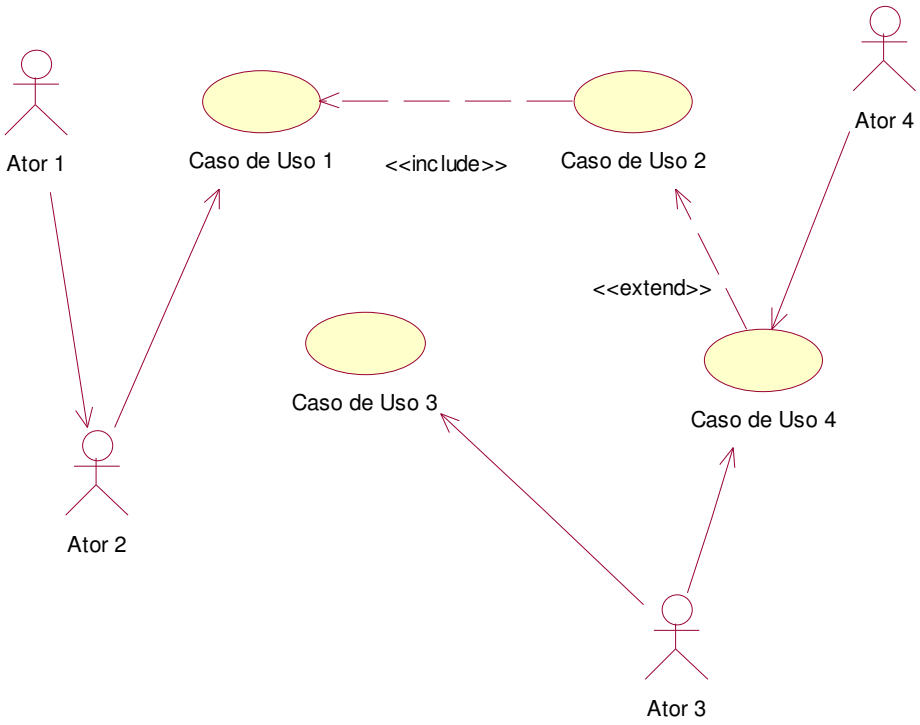
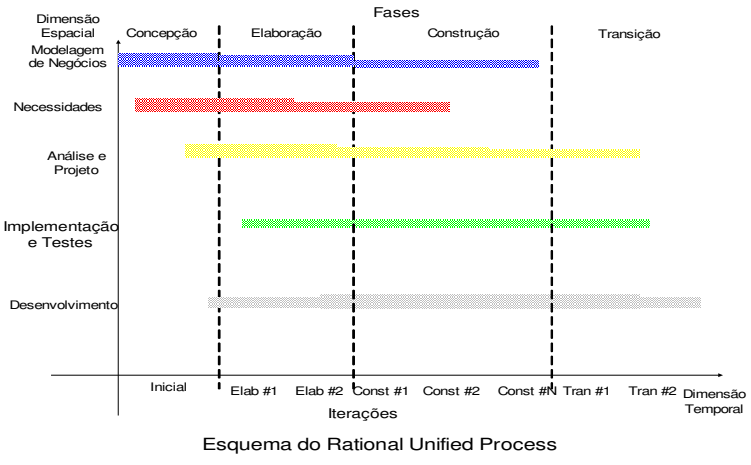


Figura 10 - Pontos de Caso de Uso

- **Suporte a metodologia RUP**

O CoCoMo II, quando aplicado ao RUP, estima o custo, esforço, prazo e equipe média para as fases de *Elaboration e Construction* (as fases do RUP podem ser vistas na seção 2.2.2.6 deste trabalho (*Rational Unified Process*)). As fases *Inception e Transition* são estimadas como percentuais da soma *Elaboration* + *Construction*, segundo AGUIAR (2004), como ilustra a figura 11.

O CoCoMo II produz estimativas para a região do ciclo de vida do RUP situada entre os marcos LCO e IOC.



Esquema do Rational Unified Process

Figura 11 - Fases da metodologia RUP

- **Aplicação em projetos com desenvolvimento iterativo.**

Segundo KRUTCHEN (2003) é aconselhada a utilização de uma abordagem iterativa no desenvolvimento de projetos por algumas razões:

A abordagem iterativa em conta alterações de exigências, na fase de análise de requisitos, visto que normalmente as exigências mudam, que podem conduzir a perda de prazos;

Ela permite identificar os riscos cedo, porque a integração é geralmente o único momento em que os riscos são descobertos ou endereçados;

Proporciona o gerenciamento com meios de realizar alterações táticas ao produto por qualquer razão, como por exemplo, para competir com produtos existentes;

Facilita a reutilização porque é fácil de identificar partes comuns, como elas são projetadas parcialmente ou implementadas, em vez de identificar no início, antes que qualquer coisa seja projetada ou implementada;

Resulta em uma arquitetura mais robusta, porque os erros são corrigidos em cima de várias iterações. Falhas são detectadas até mesmo nas iterações prematuras, e não em uma fase de teste massiva no final;

Desenvolvedores podem aprender durante o projeto, e suas habilidades e especialidades podem ser direcionadas e empregadas mais completamente durante todo o ciclo de vida;

O processo de desenvolvimento é melhorado e refinado durante todo o ciclo. A avaliação ao término de uma iteração não só olha para o status do projeto de uma perspectiva produto/prazo, como também analisa o que deveria ser alterado na organização e no processo, para melhorar na próxima iteração. A figura 12, na página seguinte, exemplifica esse ciclo.

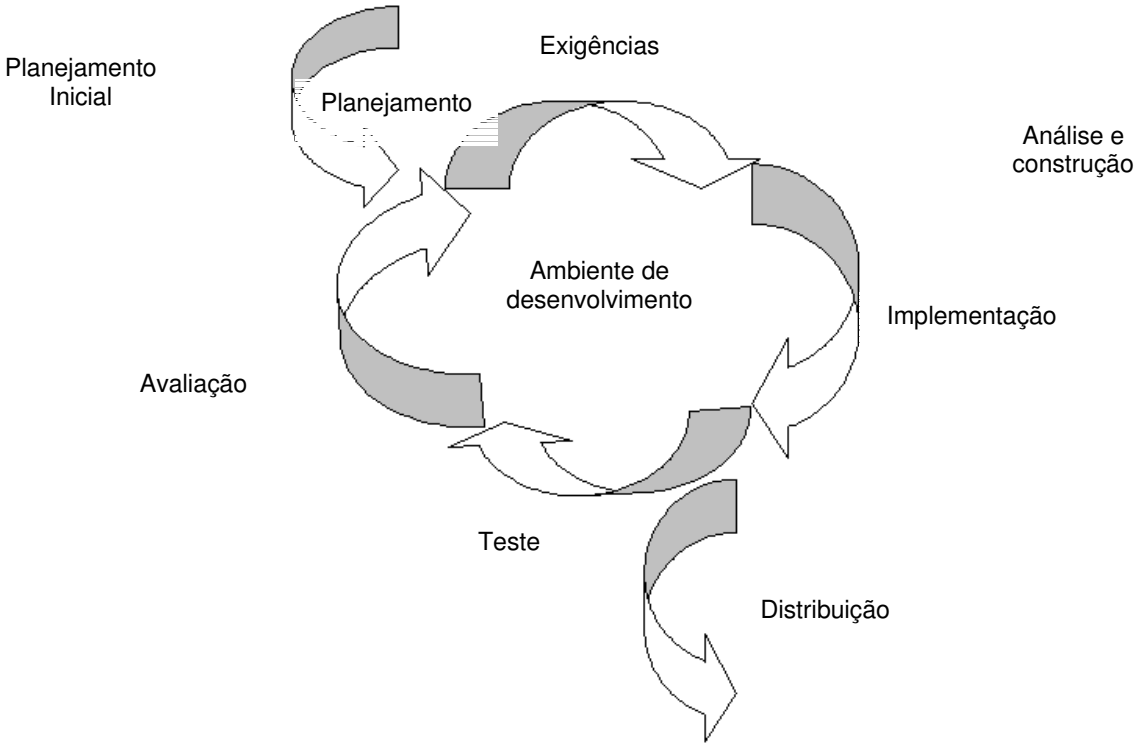


Figura 12 - Fases de um desenvolvimento iterativo

Integrado ao modelo CoCoMo II, o RUP pode calcular o custo e esforço necessário a cada iteração realizada, como apresentado na figura 13, podendo desta forma estimar de forma mais precisa os fatores necessários para o desenvolvimento de um projeto ou produto.

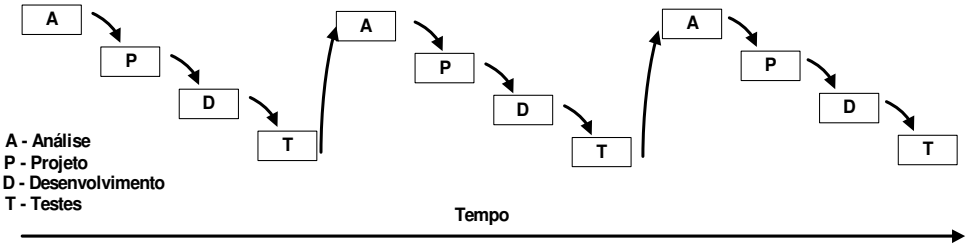


Figura 13 - Fases iterativas do RUP integradas com CoCoMo II

2.3.3.5 Limitações do Modelo CoCoMo II

Segundo BOEHM (2000), o modelo CoCoMo II apresenta algumas limitações em sua utilização, sendo que as principais são apresentadas a seguir:

A calibração do modelo é realizada em projetos concluídos, mas não garante que a aplicação em projetos de contexto não semelhante seja extremamente bem sucedida; pode ser necessário realizar alguma adaptação na calibração para o ambiente ou projeto específico.

O fator *Schedule* (SCED), que já foi um guia de custo para CoCoMo81, continua sendo uma área comum de dificuldade e limitação do modelo. Por exemplo, se um projeto é planejado para 10 pessoas durante um ano, entao é necessário um esforço de 120 pessoas meses. Se uma nova situação surgir, sendo necessário agora 9 meses para desenvolver uma aplicação, quantas pessoas são necessárias?

Poderia ser dito que 14 (120/9) pessoas seria o bastante. Porém, tal diminuição do *Schedule* aumentaria o esforço em 22%. Se o esforço fosse aumentado em 22%, então o projeto deveria requerer 16 (120*1.22/9) pessoas. Desta forma, BOEHM (2000) recomenda que a estimativa seja refeita com as novas variáveis, caso haja alguma alteração no tempo ou número de pessoas do projeto.

2.3.3.6 Diferenças entre os modelos CoCoMo 81 e CoCoMo II

A tabela 5 apresenta as diferenças entre o modelo CoCoMo 81 e CoCoMo II.

Tabela 5 – Diferenças entre os modelos

	CoCoMo 81	***CoCoMo II***
Estrutura do Modelo	Um modelo único onde se assume que o início com alguns requisitos atribuídos ao software	Três modelos que assumem que se progride através do desenvolvimento do tipo espiral para consolidar os requisitos e a arquitetura, reduzindo os riscos
Fórmula Matemática da Equação do Esforço	Esforço = A(q) (SIZE)^expoente	Esforço = A(q) (SIZE)^expoente
Expoente	Constante fixa selecionada como uma função do modo: • Orgânico = 1.05 • Semidestacado = 1.12 • Embutido = 1.20	Variável estabelecida em função de uma medida de 5 fatores de escala: • PREC: Precedência • FLEX: Flexibilidade de desenvolvimento • RESL: Resolução de Arquitetura/Riscos • TEAM: Coesão da Equipe • PMAT: Maturidade do Processo
Medida	Linhas de código fonte (com extensões por pontos de função)	• Pontos Objeto, • Pontos de Função ou • Linhas de Código Fonte
Drivers de Custo	15 drivers, cada um com o que deve ser estimado: • RELY – Confiabilidade • DATA - Tamanho da Base de Dados • CPLX – Complexidade • TIME - Restrição de tempo de execução • STOR - Restrição de armazenamento principal • VIRT - Volatilidade da máquina virtual • TURN - Tempo de Resposta • ACAP - Capacidade do Analista • PCAP - Capacidade do Programador • AEXP - Experiência em aplicações • VEXP - Experiência na plataforma (máquina virtual) • LEXP - Experiência na ferramenta (linguagem) • TOOL - Uso das ferramentas de software • MODP - Uso de técnicas modernas de programação • SCED - Planejamento requerido	17 drivers, cada um com o que deve ser estimado: • RELY – Confiabilidade • DATA - Tamanho da Base de Dados • CPLX – Complexidade • RUSE - Reutilização Requerida • DOCU – Documentação • TIME - Restrição de tempo de execução • STOR - Restrição de armazenamento principal • PVOL - Volatilidade da plataforma • ACAP - Capacidade do Analista • PCAP - Capacidade do Programador • AEXP - Experiência em aplicações • PEXP - Experiência na plataforma • LTEX - Experiência na linguagem e ferramenta • PCON - Continuidade da equipe • TOOL - Uso das ferramentas de software • SITE - Desenvolvimento multilocal • SCED - Planejamento requerido

Outras Diferenças entre os Modelos	Modelo baseado em • Fórmula de reutilização linear • Requisitos razoavelmente estáveis	Possui várias melhoras, dentre as quais: • Fórmula de reutilização não-linear • Modelo de reutilização que considera o esforço necessário para entender e assimilar • Medidas de ruptura usadas para abordar a volatilidade dos requisitos • Características de autocalibração

Fonte: COCOMO MANUAL (2004)

CAPÍTULO III – IDENTIFICAÇÃO DO PROBLEMA/MODELO PROPOSTO

3.1 Identificação do Problema

Os principais problemas enfrentados por um Gerente de Projetos de Software estão relacionados aos processos de estimativa dos custos e prazos de entrega dos projetos.

Influenciando esses prazos e suas estimativas têm-se, ainda, a composição da equipe de projeto e desenvolvimento e o acompanhamento das etapas efetuadas pela mesma. Some-se a isso as crescentes mudanças no mercado e as novas tecnologias disponíveis para se projetar e desenvolver softwares. Eis aí um ambiente competitivo, extremamente dinâmico, ao qual as empresas que desenvolvem software, mesmo que não seja o seu foco, devem se adaptar para continuarem competitivas e principalmente manter sua fatia de mercado, duramente conquistada.

O Gerente de Projetos de Software torna-se, portanto, fundamental na sobrevivência e crescimento dessas empresas no mercado. Porém estimar o prazo, custo e esforço de projeto e desenvolvimento de um software é um processo delicado e que envolve muita sensibilidade do Gerente de Projetos de Software e, principalmente, uma excelente capacidade de estimativa.

Por maior que seja a experiência do profissional atuando como Gerente de Projetos de Software, ele fatalmente irá deparar-se com projetos de software com os quais ainda não tenha trabalhado, nem mesmo com algo similar ou ainda, iniciar trabalhos com uma equipe da qual ele não conhece os seus membros e nem suas capacidades. A Engenharia de Software fornece, como ferramentas possíveis de auxilio, as métricas de software e métricas para projetos, ferramentas capazes de gerar dados para serem analisados e utilizados como apoio pelos Gerentes de Projetos de Software.

Dentro deste contexto, o problema que se apresenta na unidade de análise da empresa “case” a ser estudada é o fato que, com relação aos projetos de software e das atividades que fazem parte do projeto, as mensurações e estimativas são realizadas seguindo um modelo empírico, baseado e embasado na percepção e experiência dos responsáveis pela fase de análise dos requisitos e também pelo conhecimento de quem realiza a atividade. Isso dificulta a obtenção de informações precisas sobre os projetos realizados, para efeitos de comparação do que foi estimado com o que foi realizado, bem como para a realização de estimativas para os novos projetos que serão realizados.

Para estes novos projetos, hoje é utilizada uma estimativa por semelhança, seguindo a percepção dos gerentes e analistas; de maneira totalmente empírica o gerente ou analista, ao receberem as especificações do projeto, levam-nas ao projetista e conjuntamente avaliam e discutem a melhor maneira de realizá-las, sendo que a definição da estimativa é realizada nesse momento.

3.2 Etapas de Construção do Modelo

Conforme depoimento do coordenador de projetos, THOMAS (2005), o primeiro obstáculo a ser superado para que estas ferramentas, as métricas de software e as métricas de projetos, sejam utilizadas é a efetiva implementação

dessas métricas nas empresas que desenvolvem projetos de software para poderem gerar os dados necessários e essenciais.

O segundo obstáculo a superar é a obtenção de uma ferramenta que auxilie o Gerente de Projetos de Software na compilação e análise dos dados gerados, que não são poucos. Seria impossível tudo isso se não existissem métricas de software e métricas para o projeto pelo meio qual o software é desenvolvido.

Basicamente, as métricas mostram dois indicadores básicos muito importantes:

- **Produtividade:** A produtividade depende de inúmeros fatores como a dimensão e complexidade dos sistemas a desenvolver, as linguagens de programação, o grau de reutilização ou a experiência e motivação dos participantes no processo de desenvolvimento. Para constatar uma variação de produtividade resultante da influência desses fatores é necessário medir e quantificar a produtividade.
- **Qualidade:** A qualidade dos produtos de software é traduzida através de características como a correção, eficiência, confiabilidade, portabilidade ou facilidade de manutenção. A obtenção desses dados quantitativos relativos a essas características é assim fundamental para introduzir melhorias no processo de desenvolvimento.

Para realizar a modelagem do projeto, ou da atividade a ser realizada no projeto, a melhor maneira é utilizar as ferramentas e metodologias apropriadas, que nos permitem realizar essas estimativas de forma mais precisa. Dessa forma será possível obter melhores resultados e contar com um histórico para atividades subseqüentes ou futuras. A seguir, será apresentado como o problema de estimativas pode ser resolvido utilizando as metodologias estudadas.

Conhecendo:

- Os modelos que são utilizados para estimar o custo do software;
- O nível de produtividade da equipe;
- O esforço de desenvolvimento de software;
- A taxa de produção e de manutenção de software.

Será apresentado um roteiro sobre como utilizar essas metodologias para resolver problemas de realização de estimativas de um projeto ou das suas atividades, através dos componentes de um modelo de custo, de acordo com a Figura 14, apresentada na página a seguir, sem entrar em detalhes funcionais do projeto ou da atividade.

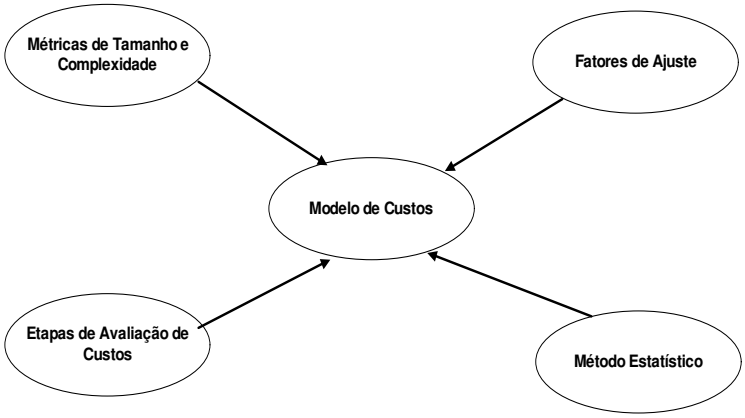


Figura 14 - Componentes de um modelo de custo

3.3 Medições pelo método de Análise de Pontos de Função

O método de Análise de Pontos de Função permite estimar o "tamanho" do software, em linhas de código, sendo o ponto inicial para se realizar as estimativas de custo e prazo de um projeto ou atividade.

Na figura 15, é ilustrado o esquema de utilização e são apresentados os passos a serem seguidos para utilização do método PFA e se obter o tamanho do software:

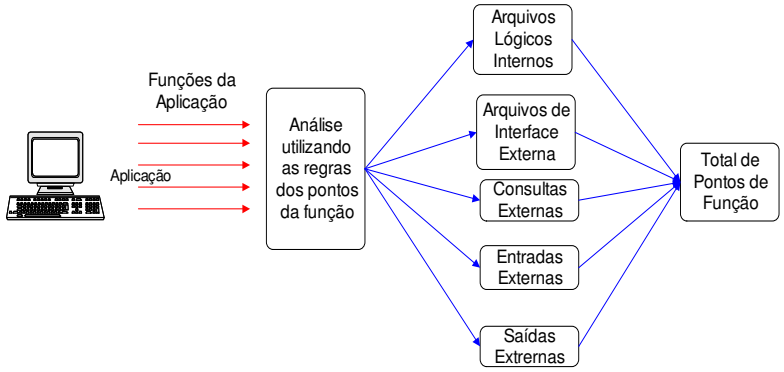


Figura 15 - Esquema geral de utilização de Análise de Pontos de Função

1° Passo - Determinar o tipo da contagem

Projeto de desenvolvimento;

Projeto de melhoria;

Aplicação já entregue e em produção

2° Passo - Identificar as fronteiras da contagem

Interfaces entre o projeto sendo medido e o usuário e outras aplicações externas que interagem com o projeto sendo medido

3° Passo - A partir das especificações recebidas, realizar um levantamento para determinar o número de:

- Arquivos lógicos internos
- Arquivos de interface externa
- Entradas externas

- Saídas externas
- Consultas externas

4° Passo - Categorizar cada um dos tipos de agrupamentos de acordo com sua complexidade e atribuir o peso correspondente. Encontrar o número de Pontos de Função não Ajustados para cada um dos tipos de agrupamentos descritos acima através da totalização dos valores correspondentes multiplicado pelo peso atribuído a cada agrupamento, através da equação:

Pontos de Função não Ajustados para cada agrupamento = somatório (número de agrupamentos em cada complexidade X peso de cada complexidade);

5° Passo - Aplicar o Fator de Ajuste para os Pontos de Função não Ajustados acima, verificando através da tabela de ***Características Nível de influência Peso*** que são adequadas ao propósito, este Fator de Ajuste é encontrado a partir da equação :

Fator de Ajuste = (NI * 0,01) + 0,65, onde :

NI = Somatório das Características Nível de influência Peso

Encontrar os **Pontos de Função Ajustados** através da equação :

PF Ajustados = PF não ajustados * Fator de Ajuste

6° Passo - Estimar o número de Instruções Fonte (**em KDSI**)

3.4 Medições pelo método CoCoMo 81

Aproveitando resultado do PFA, como ilustrado na figura 16, que permite obter o tamanho do software em linhas de código, o modelo CoCoMo 81 estima o Custo, Prazo, e Esforço (tamanho da equipe e recursos tecnológicos necessários) para projetos de software, com uma margem de imprecisão de cerca de 30 a 40% nas estimativas realizadas para cada projeto.

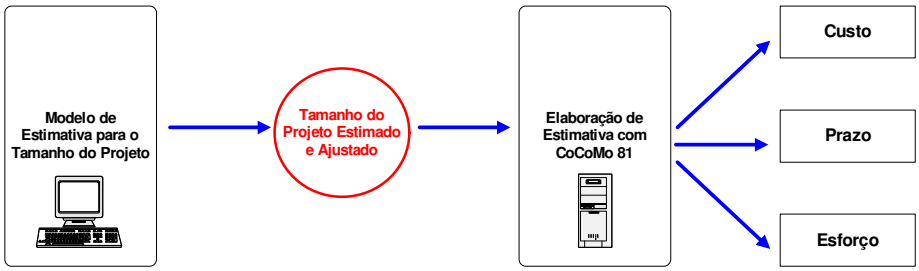


Figura 16 - Esquema de estimativa realizada no CoCoMo 81

A utilização deste modelo segue algumas premissas básicas a definir para saber qual tipo do modelo será utilizado (premissas dependentes da estrutura da organização):

- CoCoMo Básico = projetos pequenos e in-house;
- CoCoMo Intermediário;
- CoCoMo Detalhado

Exemplo de estimativa utilizando o **CoCoMo Básico:**

1° Passo - Classificar o projeto

Modo Orgânico (Convencional) = equipes pequenas, "in-house", altamente estruturado, algoritmos simples, até 50 KDSI.

2° Passo - Determinar o nº total de KDSI

(Via metodologia PFA vista acima anteriormente, ou outro modelo)

3° Passo - Utilizar a equação do Esforço

Orgânico → H/M = 2,4 (KDSI)**1,05

Difuso → H/M = 3,0 (KDSI)**1,12

Restrito → H/M = 3,6 (KDSI)**1,2

4° Passo - Utilizar a equação do Prazo

Orgânico → Prazo = 2,5 (H/M)**0,38

Difuso → Prazo = 2,5 (H/M)**0,35

Restrito → Prazo = 2,5 (H/M)**0,32

5° Passo - Estimar Pessoas

H = (H/M) : M = Esforço / Prazo = total de pessoas necessárias

A partir do número de pessoas necessárias no projeto e tendo conhecimento do valor da hora de cada integrante, é possível conhecer o custo da atividade ou projeto.

As estimativas com os outros tipos do modelo seguem o mesmo roteiro do CoCoMo Básico, com a alteração de alguns valores na equação do esforço e do prazo visto que a estrutura organizacional é diferente.

3.5 Medições pelo método CoCoMo II

O modelo CoCoMo II permite estimar e determinar o Custo, o Esforço e o Tempo de um projeto ou atividade de software a partir de um modelo de estimativa de tamanho do software. A diferença em relação ao modelo CoCoMo 81, que utiliza o resultado final que o modelo PFA fornece, é que no CoCoMo II os pontos de função são utilizados quando ainda não foram ajustados (passo 4 da utilização do método PFA visto na seção correspondente), dado que na maioria dos casos é difícil determinar exatamente o número de linhas de código que o projeto ou a atividade conterá, isto é apresentado na figura 17. Isto acarreta que ambos os modelos – de estimativa de tamanho e CoCoMo II, sejam perfeitamente compatíveis e complementares, de acordo com COCOMO MANUAL (2004).

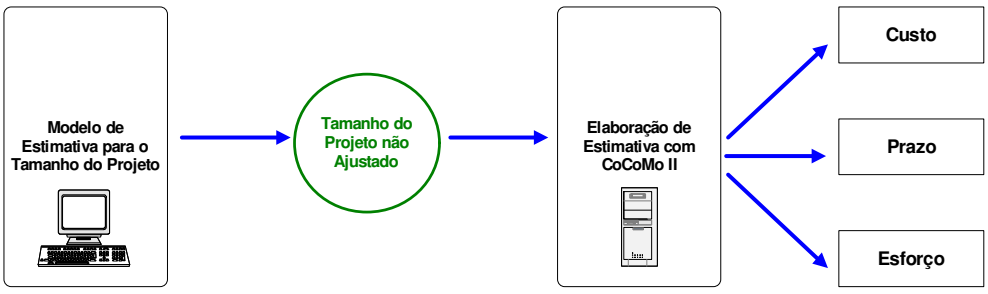


Figura 17 - Esquema de estimativa realizada no CoCoMo II

A seguir serão apresentados os passos a serem seguidos para utilização do método CoCoMo II para se obter a estimativa do Custo, Esforço e Tempo para o projeto ou atividade a ser desenvolvida, o que, de acordo com a fase em que o projeto se encontra, utiliza-se o modelo adequado.

1° Passo - Calibrar o CoCoMo II para a base histórica de projetos obtida, salvando os coeficientes em um modelo; - esta etapa é comum a todos os tipos de projeto e aos 3 modelos.

2° Passo - Aplicar a fórmula para o cálculo do esforço de desenvolvimento, esta etapa é dependente do modelo de desenvolvimento que estiver sendo utilizado, e isto é definido de acordo com o tipo de projeto.

Utilizando o modelo *Application Composition*

- ✓ Computar o número de Pontos de Objeto (OP – *Object Points*) para o sistema inteiro;
- ✓ Estimar o percentual de reutilização de código e computar o número de Novos Pontos de Objeto (NOP – *New Object Points*) requeridos;
- ✓ Determinar a taxa de produtividade (PROD) , que é o número de novos pontos objetos por mês que a equipe de projeto pode produzir;
- ✓ Estimar o esforço através da equação **E = NOP/PROD**

Utilizando o modelo *Early Design*

- ✓ Calcular pontos de função não ajustados (UFP) para o sistema;
- ✓ Converter os UFP para KSLOC, usando o fator de ajuste específico do ambiente de desenvolvimento;
- ✓ Ajustar a estimativa inicial através do conjunto de drivers de custo;
    1. Complexidade e confiabilidade do produto
    2. Reutilização requerida
    3. Dificuldade da plataforma
    4. Experiência da equipe
    5. Capacidade da equipe
    6. Facilidades disponíveis
    7. Programação
- ✓ Estimar o esforço **"E"** como : **E = A x KSLOC^b X** Π (drivers de custo)

Utilizando o modelo *Post-Architecture*

- ✓ A fórmula de estimativa para este modelo é dada pela equação:

    E = a x A x KSLOC^b x Π (drivers de custo);
- ✓ O fator **"a"** é determinado pelo tipo de projeto;
- ✓ O fator **"A"** determina o percentual de reutilização de código; e possui o valor de 2,94, conforme a calibração do CoCoMo II,
- ✓ O expoente **"b"** é derivado da soma dos 5(cinco) fatores de escala utilizando a fórmula: **b = 1,01 + 0,01 x** $\sum$ **Wi**,

Onde cada fator **"W"** possui um intervalo de valores entre 5 (baixo) e 0 (alto). Os fatores são:

Precedência

Flexibilidade de desenvolvimento

Resolução do risco/arquitetura

Coesão da equipe

Maturidade do processo

3° Passo - Aplicar o cálculo do tempo de desenvolvimento, e estimar o prazo.

TDEV = C x {PM ^ F}

Onde:

C é uma constante multiplicativa para tempo de desenvolvimento.

(C = 3.67 conforme calibragem do CoCoMo II – COCOMO MANUAL (2004))

F = [D + 0.2 x (B - 1.01)]

D é uma constante multiplicativa para tempo de desenvolvimento.

(D = 0.28 conforme calibragem do COCOMO II – COCOMO MANUAL (2004))

B assume o mesmo valor da fórmula para cálculo de esforço – Somatório dos Fatores de Escala

PM é o esforço calculado

"TDEV" é expresso em quantidade de meses.

4° Passo - A Equipe Média é obtida através da divisão do Esforço pelo Prazo. O CoCoMo II considera um mês de trabalho equivalentes a 152 horas de trabalho, excluídos finais de semana e feriados. Este valor pode ser alterado para um ambiente específico.

P = PM/TDEV

Onde:

PM é o esforço calculado

TDEV é o prazo calculado

"P" é expresso em quantidade de homens

5° Passo - Calculo do custo.

Com o prazo (**TDEV**) e Equipe média (**P**) é possível estimar o custo do software, conhecendo, além dessas variáveis, o valor hora de cada integrante envolvido no projeto.

CAPÍTULO IV – APRESENTAÇÃO E APLICAÇÃO DO ESTUDO DE CASO

4.1 Descrição da Empresa "*Case*"

A Digicon, através de seu fundador, Joseph Elbling, deu início às suas atividades no ano de 1977 com a fabricação de sistemas de medição de alta precisão para máquinas operatrizes. Hoje, o grupo Digicon, que é formado por quatro empresas, tem sua área de atuação bastante ampliada, projetando e desenvolvendo equipamentos para automação industrial e controle de processos.

Entre seus principais produtos estão: Controladores de Acesso (Catracas), Controladores de Tráfego Viário, Sistemas de Controle de Estacionamento Rotativo (Parquímetro), Sistemas de Bilhetagem Eletrônica para Transporte Urbano, Periféricos para Indústria de Transformação de Plástico, e componentes para o mercado aeronáutico.

Além disso, a Digicon comercializa equipamentos para impressão e escaneamento de grandes formatos.

A Digicon começou a atuar na área de bilhetagem em 1996. Está situada na cidade de Gravataí – RS. Inicialmente as atividades restringiam-se ao fornecimento de equipamentos, mas em pouco tempo a empresa desenvolvia também os softwares necessários para a implantação de um completo Sistema de Bilhetagem Eletrônico.

Atualmente, o Sistema de Bilhetagem da Digicon está sendo utilizado em diversos municípios de pequeno, médio e grande porte, instalado em mais de 3.000 veículos de transporte de passageiros, de acordo com DIGICON (2005).

Os benefícios da implantação desse sistema são sentidos por todos os segmentos envolvidos no processo, desde o gestor público e empresas de ônibus até os usuários. Com o Sistema de Bilhetagem da Digicon há uma redução significativa do tempo de embarque e, conseqüentemente, do tempo de viagem, e uma vez que ele utiliza tanto bilhetes magnéticos como cartões inteligentes sem contato, há menos dinheiro no ônibus, resultando em mais segurança para passageiros e tripulação.

4.2 Descrição do Caso

4.2.1 Metodologia atual utilizada para elaboração das estimativas

Nesta seção é apresentada a maneira como são desenvolvidas e realizadas atualmente as estimativas de custo e prazo dos projetos e atividades desenvolvidas na empresa "case" estudada.

As ações descritas neste contexto seguem a ordem cronológica dos fatos como ocorrem na empresa.

1) O Gerente de Projetos de Software recebe do departamento de vendas, ou diretamente do cliente, através de um contato realizado, uma solicitação para criação de um novo projeto, ou solicitação de uma nova funcionalidade, ou ainda, uma alteração de funcionalidade de um projeto existente, sendo que estas opções não são excludentes, podendo ocorrer mais de uma solicitação, conforme ilustrado na figura 18.

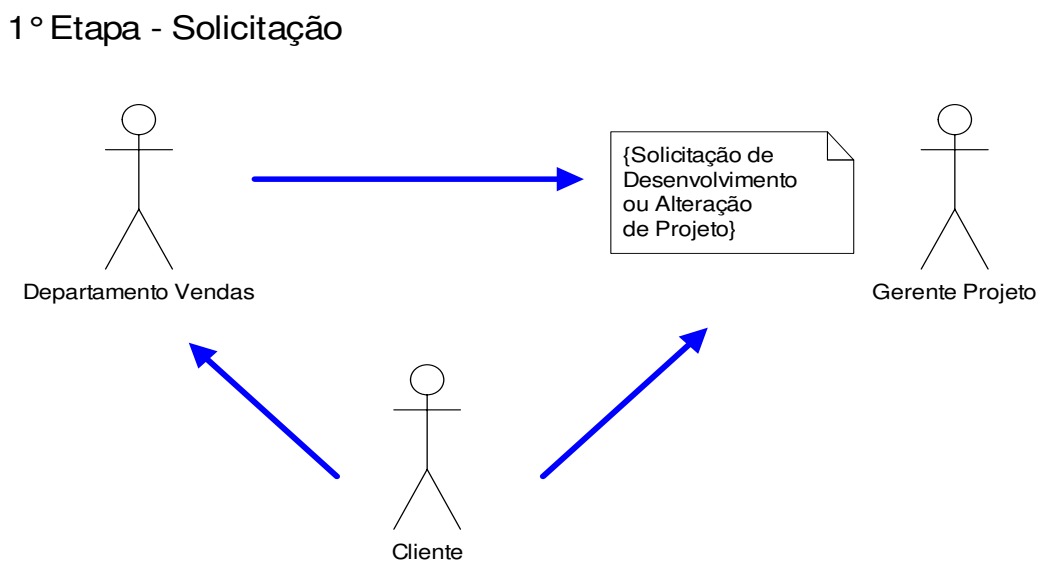


Figura 18 - Solicitação de Desenvolvimento ou Alteração de Projeto

2) O Gerente de Projetos, após analisar a viabilidade técnica da solicitação, repassa-a ao Analista de Sistemas, que, de posse dessas informações, elabora uma especificação funcional mais detalhada da solicitação, repassando-a ao Projetista de Softwares, para que elabore o projeto da solução e apresente o mesmo para o Analista, conforme ilustra a figura 19.

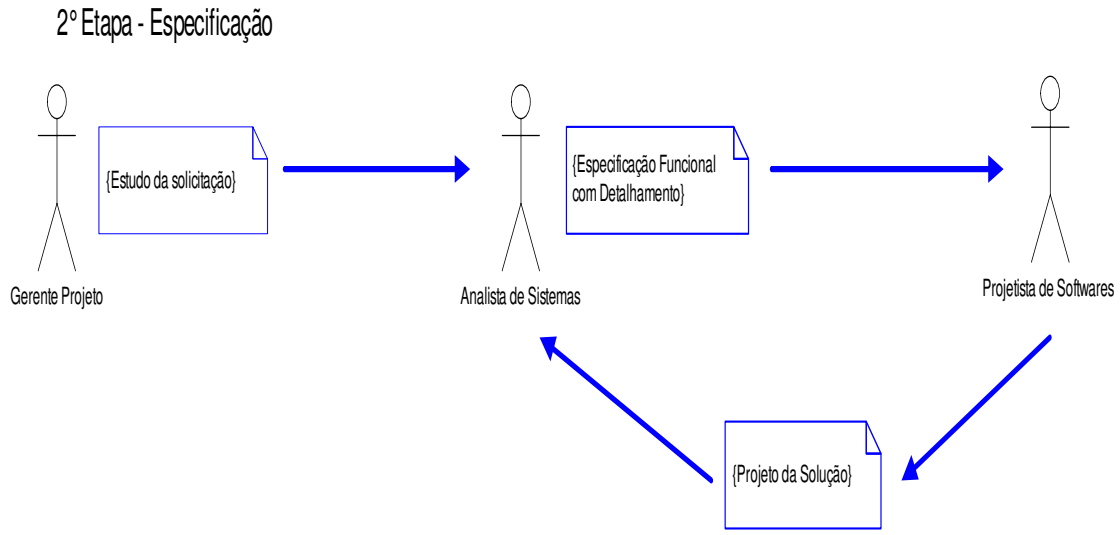


Figura 19 - Especificação da solicitação recebida

3) O Gerente de Projetos, o Analista e o Projetista em comum acordo verificam quem dos recursos humanos disponíveis (essa disponibilidade não significa que o integrante não esteja realizando alguma atividade no momento, e sim em relação ao número de atividades e complexidade que os integrantes estejam desenvolvendo no momento), teria capacidade de realizar a atividade, definindo assim o número de pessoas que realizará a atividade, com ilustra a figura 20.

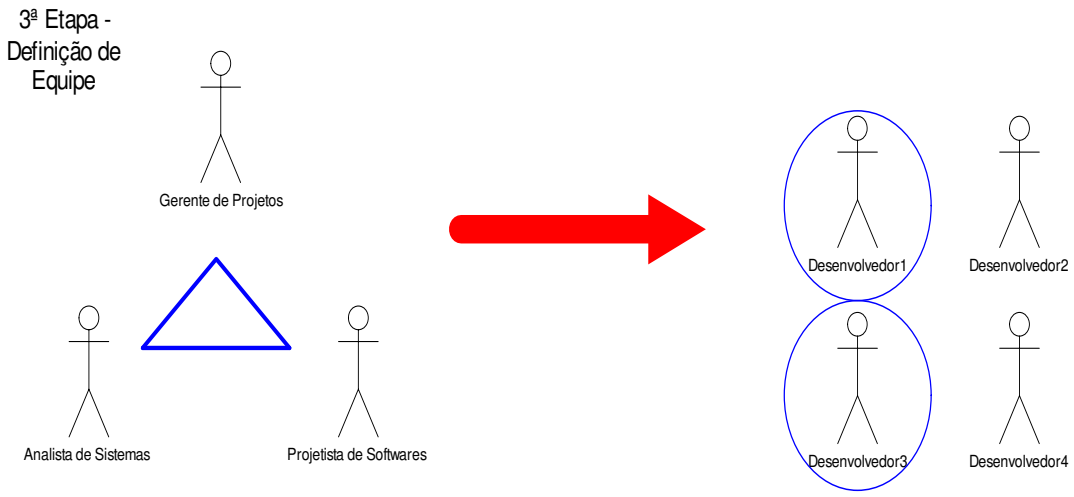


Figura 20 - Definição da equipe a ser utilizada

4) De maneira informal, as pessoas envolvidas na atividade, Analista, Projetista e desenvolvedor(es), discutem as premissas necessárias para a atividade, como ilustra a figura 21, que foram especificadas e definidas pelo projetista na etapa 2 do modelo, como tecnologia a ser utilizada, alterações e/ou adequações necessárias nas aplicações, bibliotecas e componentes adjuntas ao projeto e que podem interferir de alguma forma na realização da nova atividade.

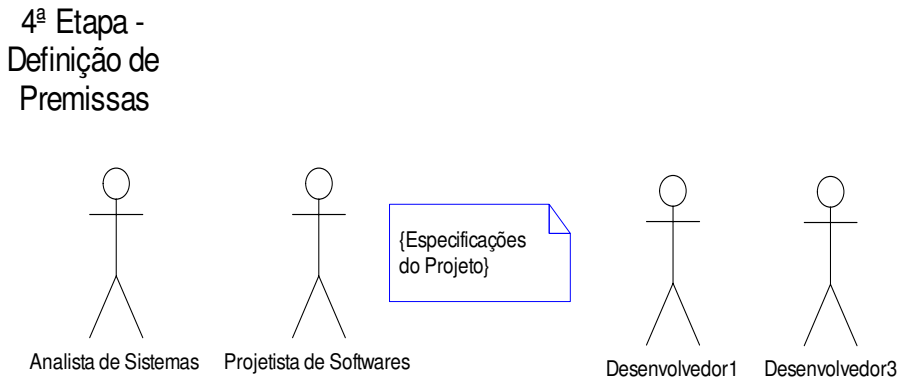


Figura 21 - Definição das premissas necessárias

5) A estimativa de definição de tempo é passada para o(s) desenvolvedor(es) envolvidos, que podem ou não concordar com a estimativa realizada e estipular um novo tempo de acordo com a sua percepção e repassar para o Gerente de Projetos e Analista, que vão analisar e retornar essa perspectiva, sendo que este ciclo permanece até que se chegue a um denominador comum entre todas as partes envolvidas nesse processo, ilustrado na figura 22.

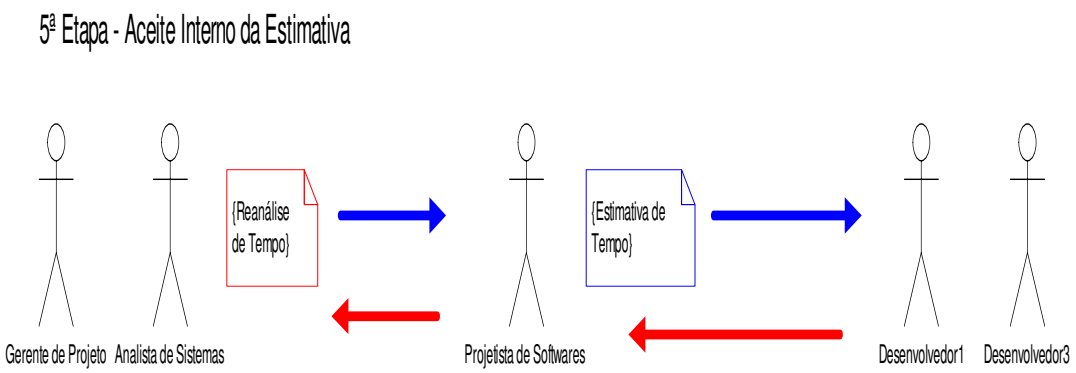


Figura 22 - Elaboração das Estimativas

6) Após essa discussão, há a definição do custo do desenvolvimento, o qual pode ser definido seja por algum projeto ou atividade anteriormente realizado e que possa ter alguma similaridade, mas que geralmente são definidas através da percepção do Gerente de Projetos e do Analista, sendo que a estimativa de tempo é repassada para o Projetista para que seja repassada ao(s) desenvolvedor(es), enquanto a estimativa de custo é utilizada para realização de orçamento para o cliente, ilustrado na figura 23

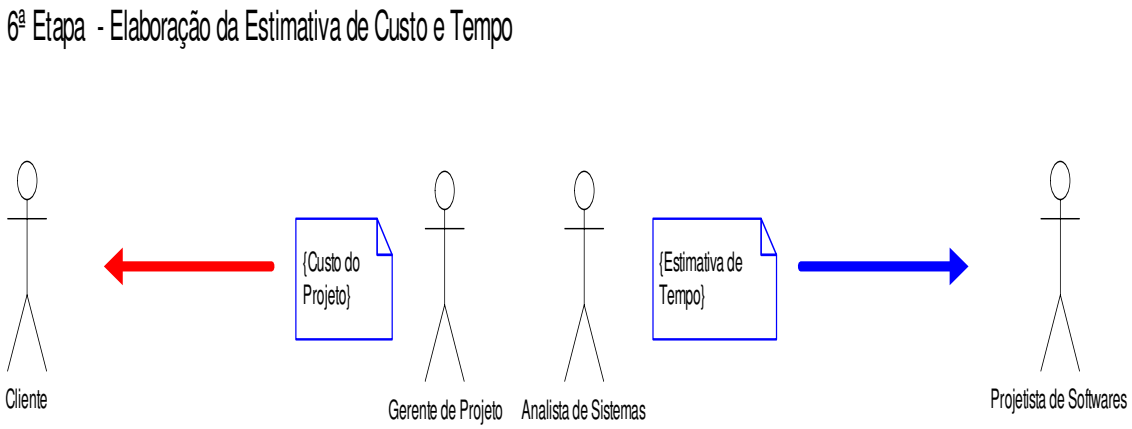


Figura 23 - Aceite da estimativa por parte da equipe

Para melhor ilustrar e exemplificar como são feitas as estimativas na empresa "case", será apresentado um breve resumo de uma solução desenvolvida e comercializada pela empresa e logo a seguir um caso de estimativa da forma como é realizada atualmente para uma solicitação recebida pela equipe de desenvolvimento de software da empresa.

- **Resumo da solução**:

Os produtos da empresa-case a serem utilizados no presente trabalho são responsáveis pelo controle de gestão e automação de transporte viário, sendo que a solução engloba desde a solicitação e venda de créditos para que os usuários possam utilizar nos veículos correspondentes, cadastro de usuários que possuem cartões para utilização e o controle e coleta dos dados dos equipamentos nos quais são utilizados os créditos que estão armazenados nos cartões.

Os cartões pré-pagos utilizados contêm créditos armazenados em sua memória para utilização nos ônibus que possuem equipamentos adequados, sendo que o mesmo pode ser recarregado mensalmente ou quando não há mais limite disponível com créditos para utilização nos equipamentos.

- **Exemplo com dados reais de estimativa da forma como é realizada atualmente:**

1) O Gerente de Projetos recebeu do cliente na cidade de Ribeirão Preto, através de um contato realizado diretamente pelo cliente, a solicitação de uma alteração no sistema para que o mesmo passe a vender créditos com casas decimais para que possam ser carregados nos cartões de utilização, ao invés do que ocorre atualmente, que considerava apenas valores inteiros na venda e utilização de créditos.

2) O Gerente de Projetos, verificando e aprovando a viabilidade da solicitação, repassa-a ao Analista para que este possa verificar os módulos que serão afetados, que vão desde as alterações em banco de dados, os softwares de venda de créditos, de controle de débito dos créditos e os componentes que realizam a coleta e sincronização dos dados e assim elabora uma especificação mais completa da solicitação. Após a elaboração dessa especificação, a mesma é repassada ao Projetista de Software, que elabora o projeto da especificação, através de diagramas, que representarão as entradas, mensagens e saídas das alterações a implementar.

3) O Gerente, o Analista e o Projetista, a partir das especificações elaboradas, verificam os projetos e suas atividades que estão em desenvolvimento no momento, tentam definir os *trade-offs* para a realização da atividade e assim definem quem serão o(s) desenvolvedor(es) que irão realizar as alterações solicitadas, que neste caso foi definido que seria utilizado um desenvolvedor, visto que a solicitação foi considerada de complexidade média.

4) Reunidos de maneira informal, que pode ser na mesa de alguns dos integrantes citados, são discutidas e definidas como as alterações serão implementadas. Como neste caso era uma alteração, a linguagem utilizada já estava definida, *Visual Basic* 6.0 que foi utilizada para implementação do software, mas houve a definição das colunas do banco de dados que haveriam de ser criados ou alterados, das rotinas dos projetos que deveriam ser reescritas para se adequar a solicitação, de acordo com o projeto elaborado anteriormente.

5) Neste caso, como o desenvolvedor estava na fase de finalização de um outro projeto, este concordou com o prazo estabelecido, apenas com a premissa de que começaria a trabalhar nesta alteração após a conclusão do projeto em que trabalhava no momento.

6) Depois desta reunião, o Gerente e o Analista reunidos definiram que o custo desta alteração seria de R$ 2.400,00 (dois mil e quatrocentos reais), o qual foi repassado para o cliente na forma de orçamento, visto que havia uma alteração conceitual no sistema, com a utilização de três pessoas como recursos humanos, e o tempo definido de 20 (vinte) dias (160 horas) para a realização, conclusão e testes da solicitação, sendo repassada para o Projetista para repassar e discutir com o desenvolvedor se o mesmo estaria de acordo com esse prazo.

Segundo THOMAS (2005), a fórmula utilizada para realizar as estimativas atualmente, é a seguinte:

Nº de dias estimados x nº de pessoas utilizadas na atividade x valor hora dos recursos x 8 horas diárias.

O valor de produção interno do software para a estimativa apresentada acima é calculado desta forma:

20 (nº dias) x 3 (nº pessoas) x 5 (valor/hora) x 8 (horas/dia) = 2.400

Com o estudo do modelo CoCoMo II, pretende-se que as estimativas deixem de ser realizadas de forma empírica para que passem a ser elaboradas a partir de um modelo científico, possuindo uma base histórica para os projetos seguintes, e auxiliando nas próximas atividades e projetos a ser desenvolvidos, permitindo que as estimativas sejam elaboradas de maneira mais precisa, como ilustrado na figura 24.

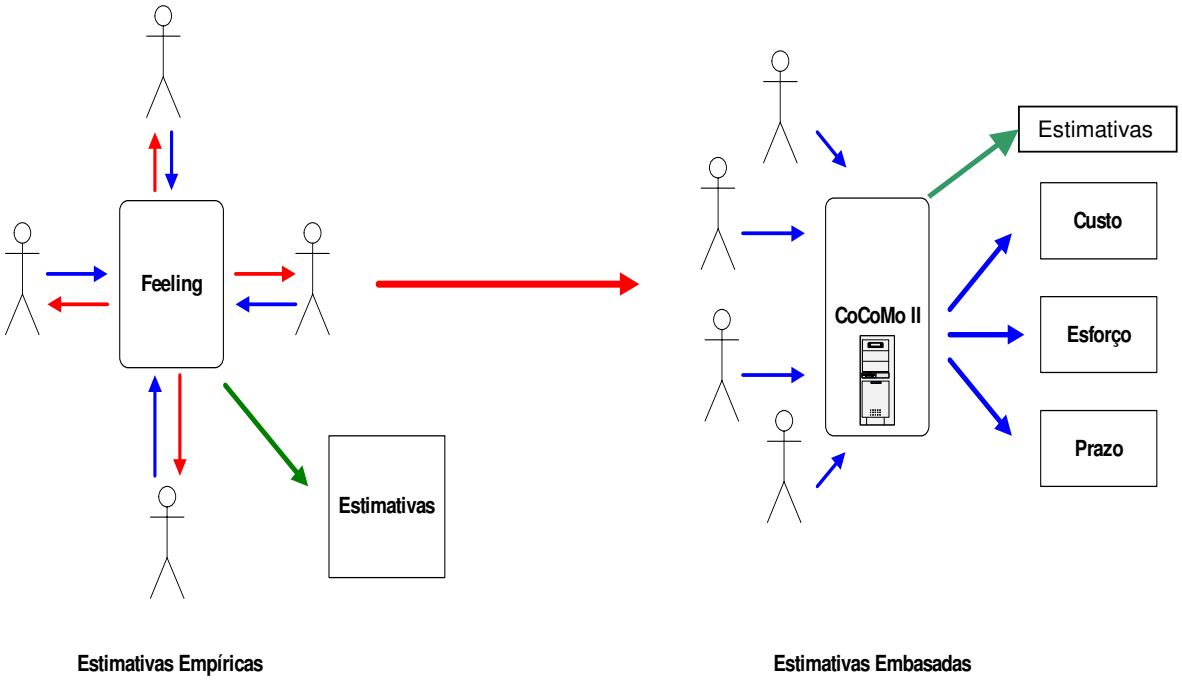


Figura 24 - Proposta de mudança do paradigma de elaboração de estimativas

4.3 Escopo das estimativas a serem realizadas

Qualidade de Projeto é um conceito diferente de qualidade do produto. A literatura comumente aborda a qualidade do produto em TI ao passo que a qualidade do projeto é um tópico mais difícil de ser pesquisado. Qualidade de Projeto é um conceito amplo que engloba muitos aspectos da qualidade do produto. O que se deve ter em mente quando se considera a qualidade do produto é que a mesma se torna intangível sem um bom gerenciamento da qualidade do projeto, segundo PRADO (2000)

Partindo desta premissa, os produtos que serão estimados neste estudo de caso, fazem parte de um projeto chamado Sistema de Bilhetagem Eletrônica. Assim, as estimativas serão referentes ao **custo dos produtos em questão**, e não da solução completa; não faz parte do escopo da estimativa o custo de todo o Sistema de Bilhetagem e sim apenas dos módulos, ou produtos, que serão citados logo a seguir, na seção 4.4 deste trabalho (Descrição dos produtos a serem analisados).

O custo da solução completa engloba outros aspectos que estão além do propósito deste trabalho, como por exemplo, as visitas realizadas, as etapas de análise, testes, implantação e manutenção do projeto, o custo do hardware que acompanha os produtos, materiais de marketing e materiais de treinamento, confecção e elaboração de manuais, os deslocamentos de pessoal a clientes para consultorias, visitas ou outras atividades, eventuais treinamentos que tenham de ser realizados para aprendizado de uma metodologia ou tecnologia, ou ainda, a contratação de pessoal ou terceirização de serviços.

É relevante ressaltar, também, que o custo estimado, não é o que está sendo repassado ao cliente final, e sim o custo que a empresa terá para o desenvolvimento do produto, este custo do produto será adicionado aos demais custos citados no parágrafo acima, sendo que este cálculo de custo total do projeto não está no escopo do presente trabalho.

4.4 Descrição dos Produtos de Software a serem analisados

4.4.1 Software estimado pelo modelo *Application Composition*

Para realizar as estimativas com este modelo, foi utilizado um projeto que ainda está em fase de planejamento e levantamento de requisitos, cuja descrição encontra-se abaixo.

O Digicon Network (DGNet), é um produto dentro do Sistema de Bilhetagem Eletrônica que tem como objetivo principal a geração e distribuição de créditos de forma automatizada e com maior segurança para utilização nos diferentes tipos de cartões existentes no sistema. Visto que a geração de créditos para vendas atualmente é realizada pelo Sistema Central de Processamento (SCP), o DGNet é uma forma de descentralizar algumas funcionalidades que são consideradas críticas no aspecto performance e segurança.

O DGNet contém em suas especificações, a utilização de uma placa criptográfica para segurança dos dados que trafegam pela rede (local e comunicação com aplicações externas), passando a ser esta a maneira com que se realizará a comunicação com os PDVs (Pontos de Venda), podendo ser operado apenas por usuários autorizados, ao contrário do que ocorre hoje no SCP, onde todos os usuários cadastrados podem realizar esta operação.

O produto DGNet, por estar em uma fase de andamento, foi avaliado e estimado pelo modelo CoCoMo II em 2 momentos distintos de seu ciclo de produção:

1. No primeiro momento, o produto estava em fase de planejamento, e por esta razão foi avaliado utilizando o modelo *Application Composition*, fase que transcorreu no instante em que foi elaborada a primeira versão da metodologia, e a fase de planejamento do produto estava concluída;
2. O segundo momento, no instante em que o presente trabalho encontrava-se na sua fase final de elaboração, e decorriam-se 130 dias de trabalho de desenvolvimento do produto, a contar do final da fase de planejamento, sendo que o mesmo encontrava-se, para efeitos de estimativas, adequado para ser avaliado pelo modelo *Early Design*.

4.4.2 Software estimado pelo modelo *Early Design*

Para realizar as estimativas com este modelo, foi utilizado um projeto que ainda está em fase de desenvolvimento, cuja descrição encontra-se abaixo.

O Terminal de Acerto de Contas (TAC) é um produto integrante da solução de transporte de passageiros que tem como objetivo principal a conferência de valores e número de passageiros que o cartão bordo liberou durante um serviço para que seja possível determinar o quanto em dinheiro o cobrador deve prestar contas, e quantos passageiros, dos diversos tipos existentes em todo o sistema, passaram durante determinado serviço, na Garagem na qual o seu veículo pertence.

O cartão do tipo bordo é de propriedade única e exclusiva do cobrador de ônibus, que o utiliza quando um passageiro que embarca não possui cartão, ou seu cartão não possui saldo suficiente para pagar a passagem, então esta é paga a dinheiro para o cobrador, que libera a roleta com o seu cartão para que o usuário possa realizar seu deslocamento, sendo que o tipo de cartão é contabilizado para ser totalizado ao final de cada viagem. Um serviço se caracteriza pelo itinerário bairro – centro, centro – bairro, realizado pelo veículo.

Além desta funcionalidade, é de responsabilidade do Terminal do Acerto de Contas recuperar e restaurar as informações do cartão bordo para o seu estado original, para que possa continuar sendo utilizado nos veículos nas próximas viagens.

O Terminal de Acerto de Contas é aplicado somente a cartões do tipo Bordo, não sendo possível a apresentação de outro tipo de cartão.

O produto TAC, por também estar em uma fase de andamento, foi avaliado e estimado pelo modelo CoCoMo II em 2 momentos distintos de seu ciclo de produção:

1. No primeiro momento, o produto estava em fase inicial de desenvolvimento, com os requisitos iniciais finalizados, e por esta razão sua avaliação inicial foi utilizando o modelo *Early Design*. Esta fase transcorreu no instante em que foi elaborada a primeira versão da metodologia, em que se iniciava a fase final de especificações do desenvolvimento do produto,
2. O segundo momento, no instante em que o presente trabalho encontrava-se com seus requisitos defindos e e aprovados pelo cliente final, e decorriam-se 125 dias de trabalho de desenvolvimento do produto, sendo que o mesmo encontrava-se, para efeitos de estimativas, adequado para ser avaliado pelo modelo *Post-Architecture*.

4.4.3 Softwares estimados pelo modelo *Post-Architecture*

Para realizar as estimativas com este modelo, foram utilizados dois projetos já concluídos, cujas descrições encontram-se abaixo.

- ✓ Sistema Central de Processamento

O Sistema Central de Processamento (SCP) é o aplicativo responsável pela gestão e configuração do Sistema de Bilhetagem Eletrônica. É este aplicativo que possibilita o cadastramento das linhas do sistema de transporte público que operam na cidade, os tipos de cartões de usuários que são aceitos no sistema de

bilhetagem, as restrições aplicáveis a estes cartões, definição do tipo de créditos que vão utilizar , que podem especiais ou subsidiados, a emissão de cartões Másteres, que são os responsáveis pela geração e distribuição dos créditos para os PDVs (Pontos de Venda),

O SCP também é o aplicativo responsável pela comunicação com e entre os PDVs, e também pela comunicação com o SCP e o SGG (Sistema Gerenciador de Garagem), que é outro módulo do Sistema de Bilhetagem Eletrônica.

- ✓ Sistema de Cadastramento e Atendimento

O Sistema de Cadastramento e Atendimento (SCA), responde por toda a operação de cadastros e necessidades do usuário e funcionários das empresas que utilizam o Sistema de Bilhetagem Eletrônica.

Como objetivo principal, o SCA destina-se ao atendimento do usuário final, nas etapas de cadastramento, solicitação de cartão e disponibilidade de informações de caráter geral.

Além das tarefas relativas ao cadastramento, o SCA também disponibiliza ferramentas para o processamento de solicitações de créditos para empresas, devidamente cadastradas, que fornecem o benefício de utilização de créditos para os seus funcionários.

Os produtos SCP e SCA, por estarem finalizados ao momento em que foi realizado o presente trabalho, foram avaliados e estimados pelo modelo CoCoMo II em um único momento, utilizando o modelo *Post-Architecture* para elaboração das estimativas.

4.5 Aplicação Prática do Modelo CoCoMo II nos Softwares

4.5.1 Produto de Software em planejamento: estimativa utilizando o modelo *Application Composition*

- ✓ **Software DGNet**
- **Cálculo dos Pontos Objetos do produto**

Interfaces – Telas

	Total na Categoria	Complexidade	Total
Simples	23	1	23
Médias	20	2	40
Complexas	11	3	33
Total	**54**		**96**

Total de Interfaces = 96

Classes

As classes, e os demais componentes 3GL devem ser totalizados e este total multiplicado por 10.

Total de Classes e Componentes 3GL = 29

Total de Classes : 29 x 10 = 290

Relatórios 10 Relatórios

	Total na Categoria	Complexidade	Total
Simples	3	2	6
Médias	2	5	10
Complexas	5	8	40
Total	10		56

Total de Relatórios = 56

Total de Pontos Objetos = 96 + 290 + 56 = 442

- **Porcentagem de Reuso do Software**

30% – porcentagem de reutilização de módulos - estimada.

NOP (*New Object Points* – Novos Pontos de Objeto) = (OP x (100 - reuse)) / 100

NOP = (442 x (100 - 30))/100

NOP = 309,40

PROD - através da tabela correspondente

Experiência e capacidade dos desenvolvedores.

Alta = PROD = 25

- **Esforço de Desenvolvimento (Computacional)**

PM = NOP/PROD

PM = 309,40 / 25

PM = 13 meses

- **Tempo de Desenvolvimento**

TEDV = 3.67 x PM ^ (0,28 + 0,2 x (b – 1,01))

TEDV = 3.67 x 13 ^ (0,28 + 0,2 x (1.0 – 1,01))

TEDV = 8 meses

- **Estimativa do RH**

P = PM / TDEV

P = 13 / 8

P = 2 pessoas

- **Custo do produto – fase de levantamento de requisitos**

8 meses de especificações e projeto

152 horas mensais (de acordo com COCOMO MANUAL (2004))

8 horas diárias

19 dias úteis mensais, descontando finais de semana e eventuais feriados.

Total de dias = 152 dias

Total de horas = 1216 horas por integrante

1.216 x 2 integrantes = 2.432

Custo aproximado da hora de desenvolvimento de cada integrante = 5

2.432 x 5 = 12.160

Custo estimado = R$ 12.160,00 (Doze mil e cento e sessenta reais)

Onde:

b = 1.0 (de acordo com COCOMO MANUAL (2004))

4.5.2 Produtos de Software em andamento: estimativa utilizando o modelo *Early Design*

- ✓ **Software TAC**

1ª fase – Estimativa de Tamanho por Pontos de Função Não Ajustados

- **Tipo de projeto –** Projeto de desenvolvimento
- **Classificação de cada elemento da aplicação**

Arquivos Lógicos Internos

	Total no Produto	Complexidade	Total
Simples	4	7	28
Médio	2	10	20
Complexo	3	15	45
Total	9		93

Arquivos de interface externa

	Total no Produto	Complexidade	Total
Simples	5	5	25
Médio	4	7	28
Complexo	2	10	20
Total	11		73

Entradas externas

	Total no Produto	Complexidade	Total
Simples	2	3	6
Médio	3	4	12
Complexo	3	6	18
Total	8		36

Saídas externas

	Total no Produto	Complexidade	Total
Simples	3	4	12
Médio	2	5	10
Complexo	1	7	7
Total	6		29

Consultas externas

	Total no Produto	Complexidade	Total
Simples	3	3	9
Médio	2	4	8
Complexo	5	6	30
Total	10		47

Total de Pontos de Função não Ajustados = 278

- **Estimando o tamanho do produto**

DSI = PFNA x Peso da Linguagem Utilizada para desenvolvimento

DSI = 278 x 32

DSI = 8.896

KDSI = DSI /1000

KDSI = 8.896 / 1000 = **8,896**

2ª fase – Estimativa de Custo através do modelo CoCoMo II

- **Esforço de Desenvolvimento (Computacional)**

$PM_{(nominal)}$ = A x (SIZE)^b

$PM_{(nominal)}$ = 2,94 x (8,94048) ^ 1,1455

$PM_{(nominal)}$ = 37 meses

$PM_{(ajustado)} = PM_{(nominal)} \times \Pi EM$

$PM_{(ajustado)}$ = 37 x 1.0955087 = 41 meses

- **Tempo de Desenvolvimento**

TEDV = 3.67 x $PM_{(ajustado)}$ ^ (0,28 + 0,2 x (b – 1,01))

TEDV = 3.67 x 41 ^ (0,28 + 0,2 x (1.1455 – 1,01))

TEDV = 12 meses

- **Estimativa do RH**

P = $PM_{(ajustado)}$ / TDEV

P = 41 / 12

P = 4 pessoas

- **Custo do produto – fase de revisão dos requisitos e implementação**

12 meses de desenvolvimento

152 horas mensais (de acordo com COCOMO MANUAL (2004))

8 horas diárias

19 dias úteis mensais, descontando finais de semana e eventuais feriados.

Total de dias = 228 dias

Total de horas = 1824 horas por integrante

1824 X 4 integrantes = 7.296

Custo aproximado da hora de desenvolvimento de cada integrante = 5

7.296 x 5 = 36.480

Custo estimado = R$ 36.480,00 (Trinta e seis mil e quatrocentos e oitenta reais)

Onde:

A = 2.94 - calibração Cocomo II

SIZE = size x (1 + (BRAK[9]/100))

DSI = 8.896

size (KDSI) = 8,896

BRAK = 500 = 0,5

SIZE = 8.896 x (1 + (0,5/100))

SIZE = 8.940,48 = 8,94048

b = 0.91 + 0.01 x somatório SF

Calculo dos SF (Scale Factors)

Fator	Conceito	Valor
PREC	Muito Baixo	6.20
FLEX	Baixo	4.05
RESL	Nominal	4.24
TEAM	Baixo	4.38
PMAT	Nominal	4.68
TOTAL		23.55

$\sum$SF = 23,55

b = 0.91 + 0.01 x 23.55 = 1.1455

Cálculo dos EM (Effort Multipliers)

Multiplicador	Conceito	Valor
RCPX	Nominal	1
RUSE	Nominal	1
PDIF	Nominal	1
PERS	Alto	0.83
PREX	Muito Alto	0.71
FCIL	Baixo	1.30
SCED	Baixo	1.43
TOTAL		1.0955087

[9] BRAK é uma porcentagem que o modelo CoCoMo II utiliza para ajustar o tamanho do software. Essa porcentagem reflete a volatilidade dos requisitos em um produto, em resumo, é a porcentagem de código desperdiçado devido à volatilidade dos requisitos, segundo COCOMO MANUAL (2004).

- ✓ **Software DGNet**

1ª fase – Estimativa de Tamanho por Pontos de Função Não Ajustados

- **Tipo de projeto –** Projeto de desenvolvimento
- **Classificação de cada elemento da aplicação**

Arquivos Lógicos Internos

	Total no Produto	Complexidade	Total
Simples	4	7	28
Médio	3	10	30
Complexo	0	15	0
Total	7		58

Arquivos de interface externa

	Total no Produto	Complexidade	Total
Simples	5	5	25
Médio	7	7	49
Complexo	6	10	60
Total	18		134

Entradas externas

	Total no Produto	Complexidade	Total
Simples	7	3	21
Médio	5	4	20
Complexo	2	6	12
Total	14		53

Saídas externas

	Total no Produto	Complexidade	Total
Simples	7	4	28
Médio	4	5	20
Complexo	5	7	35
Total	16		83

Consultas externas

	Total no Produto	Complexidade	Total
Simples	4	3	12
Médio	6	4	24
Complexo	3	6	18
Total	12		54

Total de Pontos de Função não Ajustados = 382

- **Estimando o tamanho do produto**

DSI = PFNA x Peso da Linguagem Utilizada para desenvolvimento

DSI = 382 x 32

DSI = 12.224

KDSI = DSI /1000

KDSI = 12.224 / 1000 = **12,224**

2ª fase – Estimativa de Custo através do modelo CoCoMo II

- **Esforço de Desenvolvimento (Computacional)**

$PM_{(nominal)}$ = A x (SIZE)^b

$PM_{(nominal)}$ = 2,94 x (12,28512) ^ 1.1331

$PM_{(nominal)}$ = 51 meses

$PM_{(ajustado)} = PM_{(nominal)} \times \Pi EM$

$PM_{(ajustado)}$ = 51 x 1.00199186 = 52 meses

- **Tempo de Desenvolvimento**

TEDV = 3.67 x $PM_{(ajustado)}$ ^ (0,28 + 0,2 x (b – 1,01))

TEDV = 3.67 x 52 ^ (0,28 + 0,2 x (1.1331– 1,01))

TEDV = 13 meses

- **Estimativa do RH**

P = $PM_{(ajustado)}$ / TDEV

P = 52 / 13

P = 4 pessoas

- **Custo do produto – fase de revisão dos requisitos e implementação**

13 meses de desenvolvimento

152 horas mensais (de acordo com COCOMO MANUAL (2004))

8 horas diárias

19 dias úteis mensais, descontando finais de semana e eventuais feriados.

Total de dias = 247 dias

Total de horas = 1976 horas por integrante

1976 x 4 integrantes = 7.904

Custo aproximado da hora de d\esenvolvimento de cada integrante = 5

7.904 x 5 = 39.520

Custo estimado = R$ 39.520,00 (Trinta mil, quinhentos e vinte reais)

Onde:

A = 2.94 – (de acordo com COCOMO MANUAL (2004))

SIZE = size x (1 + (BRAK/100))

DSI = 12.224

size (KDSI) = 12,224

BRAK = 500 = 0,5

SIZE = 12.224 x (1 + (0,5/100))

SIZE = 12.285,12 = 12,28512

b = 0.91 + 0.01 x somatório SF

Cálculo dos SF (Scale Factors)

Fator	Conceito	Valor
PREC	Baixo	4.96
FLEX	Baixo	4.05
RESL	Nominal	4.24
TEAM	Baixo	4.38
PMAT	Nominal	4.68
TOTAL		22.31

$\sum$SF = 22.31

b = 0.91 + 0.01 x 22.31 = 1.1331

Cálculo dos EM (Effort Multipliers)

Multiplicador	Conceito	Valor
RCPX	Baixo	0.98
RUSE	Baixo	0.95
PDIF	Nominal	1
PERS	Alto	0.83
PREX	Muito Alto	0.71
FCIL	Baixo	1.30
SCED	Baixo	1.43
TOTAL		1.0199186

4.5.3 Produtos de Software em fase final: estimativa utilizando o modelo *Post-Architecture*

- ✓ **Software TAC**

1ª fase – Estimativa de Tamanho por Pontos de Função Não Ajustados

- **Tipo de projeto –** Projeto em desenvolvimento
- **Classificação de cada elemento da aplicação**

Arquivos Lógicos Internos

	Total no Produto	Complexidade	Total
Simples	17	7	119
Médio	9	10	90
Complexo	11	15	165
Total	37		374

Arquivos de interface externa

	Total no Produto	Complexidade	Total
Simples	14	5	70
Médio	18	7	126
Complexo	13	10	130
Total	45		326

Entradas externas

	Total no Produto	Complexidade	Total
Simples	10	3	30
Médio	16	4	64
Complexo	8	6	48
Total	34		142

Saídas externas

	Total no Produto	Complexidade	Total
Simples	7	4	28
Médio	7	5	35
Complexo	10	7	70
Total	24		133

Consultas externas

	Total no Produto	Complexidade	Total
Simples	8	3	24
Médio	9	4	36
Complexo	8	6	48
Total	25		108

Total de Pontos de Função não Ajustados = 1083

- **Estimando o tamanho do produto**

DSI = PFNA x Peso da Linguagem Utilizada para desenvolvimento

DSI = 1083 x 32

DSI = 30.112

KDSI = DSI /1000

KDSI = 34.656 / 1000 = **34,656**

2ª fase – Estimativa de Custo através do modelo CoCoMo II

- **Esforço de Desenvolvimento (Computacional)**

$PM_{(nominal)}$ = A x (SIZE)^b

$PM_{(nominal)}$ = 2,94 x (34,82928) ^ 1,1129

$PM_{(nominal)}$ = 153 meses

$PM_{(ajustado)} = PM_{(nominal)} \times \Pi EM$

$PM_{(ajustado)}$ = 153 x 0.363343772 = 56 meses

- **Tempo de Desenvolvimento**

TEDV = 3.67 x $PM_{(ajustado)}$ ^ (0,28 + 0,2 x (b – 1,01))

TEDV = 3.67 x 56 ^ (0,28 + 0,2 x (1.1129 – 1,01))

TEDV = 13 meses

- **Estimativa do RH**

P = $PM_{(ajustado)}$ / TDEV

P = 56 / 13

P = 5 pessoas

- **Custo do produto**

Total de horas estimadas

13 meses

152 horas mensais (de acordo com a definição do modelo CoCoMo II)

8 horas diárias

19 dias úteis mensais, descontando finais de semana e eventuais feriados.

Total de dias = 247

Total de horas = 1976 horas por integrante

1976 x 5 integrantes = 9.880

Custo aproximado da hora de desenvolvimento de cada integrante = 5

9.880 x 5 = 49.400

Custo estimado = R$ 49.400,00 (Quarenta e nove mil e quatrocentos reais)

Onde:

A = 2.94 – (de acordo com COCOMO MANUAL (2004))

SIZE = size x (1 + (BRAK/100))

DSI = 34.656

size (KDSI) = 34,656

BRAK = 500 = 0,5

SIZE = 34.656 x (1 + (0,5/100))

SIZE = 34.829,28 = 34,82928

b = 0.91 + 0.01 x somatório SF

Cálculo dos SF (Scale Factors)

Fator	Conceito	Valor
PREC	Baixo	4.96
FLEX	Alto	2.03
RESL	Nominal	4.24
TEAM	Baixo	4.38
PMAT	Nominal	4.68
TOTAL		20.29

∑SF = 20.29

b = 0.91 + 0.01 x 20.09 = 1.1129

Cálculo dos EM (Effort Multipliers)

Multiplicador	Conceito	Valor
RELY	Muito Baixo	0.82
DATA	Alto	1.14
CPLX	Baixo	0.87
RUSE	Nominal	1
DOCU	Muito Baixo	0.81
TIME	Nominal	1
STOR	Nominal	1
PVOL	Baixo	0.87
ACAP	Baixo	1.19
PCAP	Alto	0.88
PCON	Alto	0.90
AEXP	Alto	0.88
PEXP	Alto	0.91
LTEX	Muito Alto	0.84
TOOL	Nominal	1
SITE	Nominal	1
SCED	Nominal	1
TOTAL		0.363343772

- ✓ **Software SCA**

1ª fase – Estimativa de Tamanho por Pontos de Função Não Ajustados

- **Tipo de projeto** – Projeto já entregue e em produção
- **Classificação de cada elemento da aplicação**

Arquivos Lógicos Internos

	Total no Produto	Complexidade	Total
Simples	8	7	56
Médio	12	10	120
Complexo	9	15	135
Total	29		311

Arquivos de interface externa

	Total no Produto	Complexidade	Total
Simples	7	5	35
Médio	10	7	70
Complexo	10	10	100
Total	27		205

Entradas externas

	Total no Produto	Complexidade	Total
Simples	8	3	24
Médio	3	4	12
Complexo	5	6	30
Total	16		66

Saídas externas

	Total no Produto	Complexidade	Total
Simples	5	4	20
Médio	7	5	35
Complexo	5	7	35
Total	17		90

Consultas externas

	Total no Produto	Complexidade	Total
Simples	3	3	9
Médio	2	4	8
Complexo	5	6	30
Total	10		47

Total de Pontos de Função não Ajustados = 1030

- **Estimando o tamanho do produto**

DSI = PFNA x Peso da Linguagem Utilizada para desenvolvimento

DSI = 1030 x 32

DSI = 32.960

KDSI = DSI /1000

KDSI = 32.960 / 1000 = **32,96**

2ª fase – Estimativa de Custo através do modelo CoCoMo II

- **Esforço de Desenvolvimento (Computacional)**

$PM_{(nominal)}$ = A x (SIZE)^b

$PM_{(nominal)}$ = 2,94 x (33,1248) ^ 1,1129

$PM_{(nominal)}$ = 145 meses

$PM_{(ajustado)} = PM_{(nominal)} \times \Pi EM$

$PM_{(ajustado)}$ = 145 x 0.363343772 = 53 meses

- **Tempo de Desenvolvimento**

TEDV = 3.67 x $PM_{(ajustado)}$ ^ (0,28 + 0,2 x (b – 1,01))

TEDV = 3.67 x 53 ^ (0,28 + 0,2 x (1.1129 – 1,01))

TEDV = 13 meses

- **Estimativa do RH**

P = $PM_{(ajustado)}$ / TDEV

P = 53 / 13

P = 5 pessoas

- **Custo do produto**

Total de horas estimadas

13 meses

152 horas mensais (de acordo com COCOMO MANUAL (2004))

8 horas diárias

19 dias úteis mensais, descontando finais de semana e eventuais feriados.

Total de dias = 247

Total de horas = 1976 horas por integrante

1976 x 5 integrantes = 9.880

Custo aproximado da hora de desenvolvimento de cada integrante = 5

9.880 x 5 = 49.400

Custo estimado = R$ 49.400 (Quarenta e nove mil e quatrocentos reais)

Onde:

A = 2.94 – (de acordo com COCOMO MANUAL (2004))

SIZE = size x (1 + (BRAK/100))

DSI = 32.960

Size (KDSI) = 32,96

BRAK = 500 = 0,5

SIZE = 32.960 x (1 + (0,5/100))

SIZE = 33.124,80 = 33,1248

b = 0.91 + 0.01 x somatório SF

Cálculo dos SF (Scale Factors)

Fator	Conceito	Valor
PREC	Baixo	4.96
FLEX	Alto	2.03
RESL	Nominal	4.24
TEAM	Baixo	4.38
PMAT	Nominal	4.68
TOTAL		20.29

∑SF = 20.29

b = 0.91 + 0.01 x 20.09 = 1.1129

Cálculo dos EM (Effort Multipliers)

Multiplicador	Conceito	Valor
RELY	Muito Baixo	0.82
DATA	Alto	1.14
CPLX	Baixo	0.87
RUSE	Nominal	1
DOCU	Muito Baixo	0.81
TIME	Nominal	1
STOR	Nominal	1
PVOL	Baixo	0.87
ACAP	Baixo	1.19
PCAP	Alto	0.88
PCON	Alto	0.90
AEXP	Alto	0.88
PEXP	Alto	0.91
LTEX	Muito Alto	0.84
TOOL	Nominal	1
SITE	Nominal	1
SCED	Nominal	1
TOTAL		0.363343772

- ✓ **Software SCP**

1ª fase – Estimativa de Tamanho por Pontos de Função Não Ajustados

- **Tipo de projeto** – Projeto já entregue e em produção
- **Classificação de cada elemento da aplicação**

Arquivos Lógicos Internos

	Total no Produto	Complexidade	Total
Simples	8	7	56
Médio	12	10	120
Complexo	9	15	135
Total	29		311

Arquivos de interface externa

	Total no Produto	Complexidade	Total
Simples	7	5	35
Médio	10	7	70
Complexo	10	10	100
Total	27		205

Entradas externas

	Total no Produto	Complexidade	Total
Simples	8	3	24
Médio	3	4	12
Complexo	5	6	30
Total	16		66

Saídas externas

	Total no Produto	Complexidade	Total
Simples	5	4	20
Médio	7	5	35
Complexo	5	7	35
Total	17		90

Consultas externas

	Total no Produto	Complexidade	Total
Simples	3	3	9
Médio	2	4	8
Complexo	5	6	30
Total	10		47

Total de Pontos de Função não Ajustados = 1029

- **Estimando o tamanho do produto**

DSI = PFNA x Peso da Linguagem Utilizada para desenvolvimento

DSI = 1029 x 32

DSI = 32.928

KDSI = DSI /1000

KDSI = 32.928 / 1000 = **32,928**

2ª fase – Estimativa de Custo através do modelo CoCoMo II

- **Esforço de Desenvolvimento (Computacional)**

$PM_{(nominal)}$ = A x (SIZE)^b

$PM_{(nominal)}$ = 2,94 x (33,09264) ^ 1,1129

$PM_{(nominal)}$ = 145 meses

$PM_{(ajustado)} = PM_{(nominal)} \times \Pi EM$

$PM_{(ajustado)}$ = 145 x 0.911075246 = 133 meses

- **Tempo de Desenvolvimento**

TEDV = 3.67 x $PM_{(ajustado)}$ ^ (0,28 + 0,2 x (b – 1,01))

TEDV = 3.67 x 133 ^ (0,28 + 0,2 x (1,1129– 1,01))

TEDV = 16 meses

- **Estimativa do RH**

P = $PM_{(ajustado)}$ / TDEV

P = 133 / 16

P = 8 pessoas

- **Custo do produto**

Total de horas estimadas

16 meses

152 horas mensais (de acordo com COCOMO MANUAL (2004))

8 horas diárias

19 dias úteis mensais, descontando finais de semana e eventuais feriados.

Total de dias = 304

Total de horas = 2432 horas por integrante

2432 x 8 integrantes = 19.456

Custo aproximado da hora de desenvolvimento de cada integrante = 5

19.456 x 5 = 97.280

Custo estimado = R$ 97.280 (Noventa e sete mil e duzentos e oitenta reais)

Onde:

A = 2.94 – (de acordo com COCOMO MANUAL (2004))

SIZE = size x (1 + (BRAK/100))

DSI = 32.928

size (KDSI)= 32,928

BRAK = 500 = 0,5

SIZE = 32.928 x (1 + (0,5/100))

SIZE = 33.092,64 = 33,09264

b = 0.91 + 0.01 x somatório SF

Cálculo dos SF (Scale Factors)

Fator	Conceito	Valor
PREC	Baixo	4.96
FLEX	Alto	2.03
RESL	Nominal	4.24
TEAM	Baixo	4.38
PMAT	Nominal	4.68
TOTAL		20.29

$\sum$SF = 20,29

b = 0.91 + 0.01 x 20.09 = 1.1129

Cálculo dos EM (Effort Multipliers)

Multiplicador	Conceito	Valor
RELY	Muito Baixo	0.82
DATA	Alto	1.14
CPLX	Nominal	1
RUSE	Nominal	1
DOCU	Baixo	0.91
TIME	Normal	1
STOR	Nominal	1
PVOL	Nominal	1
ACAP	Muito Baixo	1.42
PCAP	Baixo	1.15
PCON	Alto	0.90
AEXP	Alto	0.88
PEXP	Alto	0.91
LTEX	Alto	0.91
TOOL	Nominal	1
SITE	Nominal	1
SCED	Nominal	1
TOTAL		0.911075246

CAPÍTULO V - ANÁLISE DE RESULTADOS

Este capítulo está desdobrado em dois momentos. Em um primeiro, é feita a comparação dos resultados obtidos à luz das estimativas realizadas pelo modelo CoCoMo II, com as estimativas realizadas da forma atual na empresa *case*, observando e analisando a aproximação ou diferença entre os resultados obtidos.

Em um segundo momento, é realizada uma simulação de proposta para a implantação da metodologia na empresa, explicitando o custo previsto e questões como: tempo de treinamento, usuários designados para utilização, itens que viabilizem e justifiquem a mudança de paradigma nos processos de estimativa da empresa.

5.1 Análise dos Resultados Obtidos

A tabela 6 apresenta um comparativo do produto de software DGNet estimado através do modelo *Application Composition*, estimativa realizada ao final da fase de planejamento de desenvolvimento do software, etapa iniciada em agosto de 2004 e concluída em janeiro de 2005.

Tabela 6 – Estimativa do produto DGNet através do modelo *Application Composition*

	Estimado Empresa	**Estimado CoCoMo II**	**Realizado**
Status produto			Fase concluída
Tempo (dias)	140	152	160
Esforço (nº pessoas)	4	2	3
Tamanho (KDSI)	Não aplicado	Não aplicado	Não aplicado
Custo (em reais)	22.400 [*]	12.160	19.200

De acordo com as estimativas realizadas pela empresa, com 140 dias úteis de trabalho e 4 pessoas seria possível concluir a fase de planejamento do produto de software DGNet, a um custo de R$ 22.400,00.

Pela estimativa elaborada no modelo, esta mesma fase do trabalho poderia ser realizada com 152 dias úteis de trabalho, e 2 pessoas a um custo de R$ 12.160,00.

Na verdade, esta etapa foi finalizada com 160 dias de trabalho, muito próximo do que o modelo estimou e com um intervalo relevante se comparado com o que foi estimado pela empresa.

Comparando a estimativa no momento da conclusão da atividade (coluna "Realizado" da Tabela 6) com o resultado que o modelo apresenta, a diferença seria de 8 dias, ou 5,2% de atraso em relação ao poderia ter sido repassado ao cliente como expectativa de conclusão desta fase, é algo que poderia ser facilmente negociado em termos de prazo. Em relação ao estimado pela empresa, a diferença

[*] De acordo com THOMAS (2005), a estimativa de custo atual é realizada através da seguinte equação:

Nº de dias da atividade x nº de pessoas alocadas na atividade x valor/hora de cada recurso humano x 8 horas/dia

se encontra em 8%, ou 12 dias, o que além de ocasionar o descontentamento por parte do cliente pelo atraso, gera um atraso na estimativa final do projeto, por ser esta fase a mais importante, a de entendimento das necessidades do cliente.

Em relação ao fator custo, na comparação da estimativa da empresa com a do modelo, haveria uma redução de aproximadamente 84,2% na estimativa elaborada pelo modelo, que estima o custo em R$ 12.160,00 contra R$ 22.400,00, ou seja, em valores uma redução de quase R$ 10.000,00 que impactaria de forma relevante no custo final do projeto. Em relação ao término desta atividade, o estimado pela empresa para esta fase de planejamento apresenta um acréscimo de 16,6% em relação ao que foi realizado, dos R$ 22.400,00 estimados contra os R$ 19.200,00 que efetivamente é o custo interno desta etapa.

Em relação à estimativa de esforço, a empresa previu que 4 pessoas seria o número ideal para a realização da atividade, sendo que foi realizada com 3 pessoas, representando um decréscimo de 33,3%, sendo que se a comparação for realizada entre a estimativa da empresa e do modelo, haveria uma redução de 50% do pessoal alocado para o desenvolvimento deste produto de software, sendo que estes recursos poderiam estar participando de outra atividade ou projeto.

A justificativa para a estimativa do modelo prever 2 pessoas é, que se houver dados históricos, certamente haverá maior agilidade nos processos de levantamentos e planejamento das atividades, diferente do que é realizado atualmente.

A tabela 7 apresenta um comparativo do produto de software DGNet estimado em um segundo momento, durante a fase de especificações e projeto do software, através do modelo *Early Design*, estimativa esta realizada em maio de 2005, com 130 dias de andamento do desenvolvimento do produto de software.

Tabela 7 – Estimativa do produto DGNet através do modelo *Early Design*

	Estimado Empresa	**Estimado CoCoMo II**	**Realizado**
Status produto			Em andamento
Tempo (dias)	300	247	130
Esforço (nº pessoas)	5	4	5
Tamanho (KDSI)	Não aplicado	12.28	6
Custo (em reais)	60.000 *	39.520	26.000

De acordo com as estimativas realizadas pela empresa, com 300 dias úteis de trabalho e 5 pessoas seria possível concluir esta fase do produto de software DGNet, a um custo de R$ 60.000,00.

Pela estimativa elaborada no modelo, esta etapa do desenvolvimento do produto de software poderia ser realizada em 247 dias úteis de trabalho, com uma equipe de 4 pessoas a um custo de R$ 39.520,00.

De acordo com o andamento desta etapa do desenvolvimento do produto, e verificando as informações de tempo, tamanho e custo até o presente momento, pode-se estimar, que o desenvolvimento do produto encontra-se aproximamente na sua metade, e indica que, se for seguido o padrão até o momento, conforme visto na coluna “Realizado” da Tabela 7, os valores finais estariam, de maneira aproximada, em 260 dias de desenvolvimento, com 5 pessoas e custo de R$ 52.000,00.

Comparando os valores estimados pela empresa com os previstos pelo modelo para esta etapa, o modelo prevê uma economia de 34,13% no fator custo, enquanto que se essa comparação for entre o modelo e o "provável" realizado, o modelo prevê uma economia de 24%. Em termos de prazo desta etapa de desenvolvimento do software, o modelo prevê uma economia de 18%, enquanto que se a comparação for realizada com o "provável" realizado, a economia proposta pelo modelo é de 5%. Verifica-se um equilíbrio entre a estimativa do modelo e o "provável realizado", e se este for cumprido sem muitas e relevantes exceções, além da diminuição no prazo de término de desenvolvimento e custo, poderia gerar mais e melhores testes antes da integração ao produto final, o que em termos de qualidade agregaria maiores benefícios e valor ao cliente e à empresa.

A tabela 8 apresenta um comparativo do produto de software TAC estimado em um primeiro momento através do modelo *Early Design*, estimativa realizada ao final da fase de especificações iniciais no desenvolvimento do software, e que foi iniciada em março de 2004, sendo concluída em janeiro de 2005.

Tabela 8 – Estimativa do produto TAC através do modelo *Early Design*

	Estimado Empresa	**Estimado CoCoMo II**	**Realizado**
Status produto			Fase concluída
Tempo (dias)	300	228	250
Esforço (nº pessoas)	5	4	5
Tamanho (KDSI)	Não aplicado	8.94	10
Custo (em reais)	60.000 *	36.480	50.000

De acordo com as estimativas realizadas, a elaborada pela empresa estima que com 300 dias úteis de trabalho e 5 pessoas será possível concluir a fase de especificações e projeto do produto TAC, a um custo de R$ 60.000,00.

Pela estimativa elaborada no modelo neste primeiro momento, o desenvolvimento do produto pode ser realizado com 228 dias úteis de trabalho, e 4 pessoas a um custo de R$ 36.480,00.

Comparando a estimativa no momento da conclusão da atividade (coluna realizado da Tabela 6) com o resultado do modelo, a diferença de tempo é de 22 dias, o que geraria uma economia de 8,8% no tempo de desenvolvimento desta etapa.

Em relação ao fator custo, na comparação da estimativa da empresa com a do modelo, haveria uma redução de aproximadamente 64,4% na estimativa elaborada pelo modelo, que estima o custo em R$ 36.480,00 contra R$ 60.000,00, ou seja, em valores uma redução de mais de R$ 20.000,00 que impactaria de forma relevante no custo final do projeto. E em relação ao término desta atividade, o estimado pela empresa para esta fase de planejamento apresenta um acréscimo de 20% em relação ao que foi realizado, dos R$ 60.000,00 estimados contra os R$ 50.000,00 que efetivamente é o custo interno desta etapa.

Em relação à estimativa de esforço, foram efetivamente utilizadas as 5 pessoas previstas para esta atividade, mas em relação à estimativa do modelo, que prevê 4 pessoas como número suficiente, a economia seria de 20% em recursos humanos alocados.

A justificativa para a estimativa do modelo prever 4 pessoas é, que se houver dados históricos, certamente haverá maior agilidade nos processos de levantamentos e planejamento das atividades, diferente do que é realizado atualmente.

Verifica-se um ponto de desequilíbrio entre os valores da estimativa do modelo e o "realizado", pelo fato de não haver uma metodologia mais adequada de estimativas, o que ocasiona retrabalho no desenvolvimento das atividades, impactando no custo final, visto que o tempo e o tamanho do produto nesta fase são relativamente compatíveis.

A tabela 9 apresenta um comparativo do produto de software TAC estimado em um segundo momento, através do modelo *Post-Architecture*, estimativa esta realizada em maio de 2005, com 125 dias de andamento do desenvolvimento do produto de software.

Tabela 9 – Estimativa do produto TAC através do modelo *Post-Architecture*

	Estimado Empresa	**Estimado CoCoMo II**	**Realizado**
Status produto			Em andamento
Tempo (dias)	350	247	125
Esforço (nº pessoas)	5	5	5
Tamanho (KDSI)	Não aplicado	34.82	14
Custo (em reais)	70.000 *	49.400	25.000

De acordo com as estimativas realizadas, a elaborada pela empresa estima que com 350 dias úteis de trabalho e 5 pessoas será possível concluir a fase de especificações e projeto do produto TAC, a um custo de R$ 70.000,00.

Pela estimativa elaborada no modelo, o ideal para o desenvolvimento do produto poderia ter sido realizado com 247 dias úteis de trabalho, e 5 pessoas a um custo de R$ 49.400,00.

Visto que este produto já está com todas as suas especificações concluídas, é possível ter uma idéia mais concreta de como a utilização do modelo pode beneficiar as estimativas.

A estimativa que teve maior precisão em todas as comparações é a de pessoal necessário, visto que 5 foi considerado o número ideal e suficiente de pessoas para o desenvolvimento do produto, e este número de pessoas é o que realmente está sendo utilizado no desenvolvimento do produto.

Em relação ao tempo de desenvolvimento, se for considerada a estimativa da empresa, o desenvolvimento está aproximadamente em um terço do tempo, sendo que se for considerada a estimativa do modelo, estaria aproximadamente na metade do seu tempo.

No aspecto tamanho do produto, visto que a empresa não estima atualmente esta característica, não há maneira de comparar o estimado pelo modelo com o estimado pela empresa. Porém, se verificarmos em termos do que foi realizado até o momento, pode-se perceber uma aproximação com a estimativa do modelo, com alguma defasagem em relação à estimativa elaborada pelo modelo, sendo que uma

das razões principais é que há muita repetição de código nos diversos módulos do produto, o que justifica o fato de que não foi estimada a reutilização de código.

Em termos de custo, verificando os dados obtidos, verifica-se que a diferença entre o estimado pela empresa e o estimado pelo modelo é de 30%. Porém, se considerarmos da mesma maneira feita com o tempo de desenvolvimento, se o produto de software estiver na metade do seu ciclo de desenvolvimento, o custo será de aproximadamente de 50.000, a diferença é de aproximadamente 1,2%, ou seja, o resultado estimado pelo modelo é praticamente preciso em relação ao realizado. Se considerarmos que o produto está em um terço do seu ciclo, a estimativa do modelo acarretaria uma economia de 35%, mas ficaria um tanto distante do que foi efetivamente realizado, mas haveriam fatores que justificariam essa diferença, como falta de padronização de desenvolvimento, retrabalho e não reutilização de código.

A tabela 10 apresenta um comparativo do produto de software SCA estimado através do modelo *Post-Architecture*.

Tabela 10 – Estimativa do produto SCA através do modelo *Post-Architecture*

	Estimado Empresa	**Estimado CoCoMo II**	**Realizado**
Status produto			Concluído
Tempo (dias)	450	247	300
Esforço (nº pessoas)	5	5	5
Tamanho (KDSI)	Não aplicado	33.12	50
Custo (em reais)	90.000 *	49.400	60.000

De acordo com as estimativas realizadas, a empresa estimou que com 450 dias úteis de trabalho e 5 pessoas seria possível concluir o desenvolvimento do produto SCA, a um custo de R$ 90.000,00.

Pela estimativa elaborada no modelo, o ideal para o desenvolvimento do produto poderia ter sido realizado com 247 dias úteis de trabalho, e 5 pessoas a um custo de R$ 49.400,00.

Visto que este produto já está concluído, é possível ter uma idéia mais concreta de como a utilização do modelo pode beneficiar as estimativas a realizar.

A estimativa que teve maior precisão em todas as comparações é a de pessoal necessário, visto que 5 foi considerado o número ideal e suficiente de pessoas para o desenvolvimento do produto, e este número de pessoas é o que realmente foi utilizado no desenvolvimento do produto.

Em relação ao tempo de desenvolvimento, a defasagem entre estimado e realizado pela empresa foi de 50%, visto que o produto levou menos tempo de desenvolvimento do que o previsto, entretanto na comparação com a estimativa pelo CoCoMo II, a diferença entre estimado e realizado é de 18%, tempo este a mais que pode ter sido causado, por exemplo, pela alocação de recurso em outro produto, problemas pessoais, enfim, alguma eventualidade não prevista no cronograma.

No aspecto tamanho do produto, visto que a empresa não estima atualmente esta característica, não há maneira de comparar o estimado com o realizado, mas em relação à estimativa elaborada pelo modelo, a diferença é de 21%, uma das

razões principais é que há muita repetição de código nos diversos módulos do produto, o que justifica o fato de que não foi estimada a reutilização de código.

Em termos de custo, verificando os dados obtidos, verifica-se que a defasagem entre o estimado pela empresa e o efetivamente gasto com o produto foi de 50%, a empresa gastou menos (R$ 60.000,00) do que estimou (R$ 90.000,00) para o desenvolvimento, entretanto na comparação com a estimativa pelo CoCoMo II, a diferença entre estimado e o que foi gasto é de 16,7%, que seria uma economia considerável de custo se o projeto tivesse seu curso normal estimado, esta diferença de custo é em decorrência dos fatores tempo e tamanho do produto, que influenciaram de forma diretamente proporcional.

A tabela 11 apresenta um comparativo do produto de software SCP estimado através do modelo *Post-Architecture*.

Tabela 11 – Estimativa do produto SCP através do modelo *Post-Architecture*

	Estimado Empresa	**Estimado CoCoMo II**	**Realizado**
Status produto			Concluído
Tempo (dias)	500	304	450
Esforço (nº pessoas)	8	10	8
Tamanho (KDSI)	Não aplicado	33.09	56.8502
Custo (em reais)	160.000 *	129.200	144.000

De acordo com as estimativas realizadas, a empresa previu que com 500 dias úteis de trabalho e 8 pessoas teria sido possível concluir o desenvolvimento do produto SCP, a um custo de R$ 160.000,00.

Pela estimativa elaborada no modelo, o ideal para o desenvolvimento do produto poderia ter sido realizado com 323 dias úteis de trabalho, e 10 pessoas a um custo de R$ 129.200,00.

Visto que este produto já está concluído, é possível ter uma idéia mais concreta de como a utilização do modelo pode beneficiar as estimativas

A estimativa de esforço indicava que 10 pessoas realizariam o trabalho no tempo indicado, mas efetivamente foram utilizadas 8 pessoas, o que indica uma defasagem de 25% em relação à estimativa elaborada pelo modelo, o que não significa que a estimativa realizada pela empresa tenha sido precisa pelo acerto, mas pode ter sido ocasionada pela disponibilidade de pessoal ao momento do desenvolvimento.

Em relação ao tempo de desenvolvimento, a defasagem entre estimado e realizado pela empresa foi de 11%, visto que o produto levou menos tempo de desenvolvimento do que o previsto, entretanto na comparação com a estimativa pelo CoCoMo II, a diferença entre estimado e realizado é de 10,3%, tempo este a mais que pode ter sido causado, por exemplo, pela alocação de recurso em outro produto, problemas pessoais, enfim, alguma eventualidade não prevista no cronograma.

No aspecto tamanho do produto, visto que a empresa não estima atualmente esta característica, não há maneira de comparar o estimado com o realizado, mas em relação à estimativa elaborada pelo modelo, a diferença é de 30%, uma das razões principais, como citado na análise anterior, é que há muita repetição de

código nos diversos módulos do produto, o que justifica o fato de que não foi estimada a reutilização de código.

Em termos de custo, verificando os dados obtidos, verifica-se que a defasagem entre o estimado pela empresa e o efetivamente gasto com o produto foi de 11%, a empresa gastou menos do que estimou para o desenvolvimento, entretanto na comparação com a estimativa pelo CoCoMo II, a diferença entre estimado e o que foi gasto é de 29%, que seria uma economia considerável de custo se o projeto tivesse seu curso normal estimado, esta diferença de custo é em decorrência dos fatores tempo e tamanho do produto, que influenciaram de forma diretamente proporcional.

5.2 Perspectiva de Implantação na Empresa

Analisando os itens que são relevantes para implantação do modelo na empresa estudada, além da mudança cultural que será necessária, surge a importância de haver um conhecimento dos itens de controle dos processos de desenvolvimento, através do quadro 5W1H, definindo desta forma atores, tarefas e momentos para cada fase de desenvolvimento.

Segundo UFRGS (2005), os itens de controle 5W1H podem ser definidos desta maneira:

- ***WHAT* (o que)** - Quais os itens de controle em qualidade, custo, entrega, moral e segurança? Qual a unidade de medida?
- ***WHEN* (quando)** - Qual a freqüência com que devem ser medidos? Quando atuar?
- ***WHERE* (onde)** - Onde são conduzidas as ações de controle?
- ***HOW* (como)** - Como exercer o controle. Indique o grau de prioridade para ação de cada item.
- ***WHY* (por que)** - Em que circunstancias o "controle" será exercido.
- ***WHO* (quem)** - Quem participará das ações necessárias ao controle.

Adaptando ao contexto da empresa estudada, a tabela 12 apresenta os itens 5W1H de acordo com a realidade e as necessidades para implantação do modelo CoCoMo II nos processos de desenvolvimento.

Tabela 12 – Atribuições pessoais dentro da equipe

WHAT	Planejamento e desenvolvimento de produtos de software
WHEN	A cada novo produto desenvolvido, ou alterações de produtos existentes
WHERE	Internamente com a equipe de desenvolvimento, para as definições de estimativas
HOW	Através de estimativas e armazenamento de dados sobre os produtos concluídos para obtenção de histórico para futuros produtos
WHY	Pelo aumento do custo interno de desenvolvimento de produtos de software, em decorrência de retrabalho e pouca reutilização de código
WHO	Gerentes de projeto, analistas de sistemas, projetistas e desenvolvedores

Outro item que é considerado de extrema importância pela empresa e pelo setor de desenvolvimento de projetos de software é o custo previsto para implantação. A tabela 13 apresenta os itens que são considerados importantes e o

tempo que deverá ser gasto, assim como o custo de implantação da metodologia de estimativas embasadas, e o impacto que pode causar pela mudança de paradigma.

Tabela 13 – Componentes do custo de implantação

Item	Tempo	N° de pessoas	Valor Hora (em reais)	Valor Total (em reais)
Obtenção de software para elaboração da estimativa	5 horas – durante 3 dias = 15 horas	1	5	75
Instrutor	14 horas	1	20	280
Treinamento de pessoal	2horas – durante 7 dias = 14 horas	16 (divididas em duas turmas de 8)	5	560
Instalação do software	1 hora	1	5	5
Avaliação dos dados existentes para calibração do modelo	7 dias – 56 horas	3	5	840
TOTAL	17 dias = 136 horas	16 pessoas		1760

Para o calculo do valor, considera-se o valor-hora de cada integrante da equipe de desenvolvimento, que é R$ 5,00. A hora do instrutor, de custo R$ 20,00, foi o valor repassado em contato feito com a empresa TI Métricas <http://www.timetricas.com.br>, para este tipo de treinamento na modalidade *in-company*. Outra consideração relevante, é que o total de integrantes do setor é de 16 pessoas; por esta razão o total de pessoas na coluna **N° de pessoas** na tabela 13 é 16 e não a soma de todas as linhas.

Este estudo realizado com todas as variáveis, resultados das estimativas dos produtos de software e valores necessários para implantação da metodologia, foi repassado aos gestores da empresa e do setor, para análise e estudo do custo-benefício com a adoção de uma metodologia para a elaboração de estimativas.

Na opinião dos gestores da empresa, os dois fatores mais relevantes no processo de desenvolvimento de um produto de software – o custo interno do desenvolvimento de software e o tempo de desenvolvimento – apresentaram resultados satisfatórios e animadores principalmente no aspecto que diz respeito aos procedimentos de desenvolvimento, que conforme o estudo de caso pode acarretar uma economia média de 20% a 30% em custo e tempo de desenvolvimento.

Tais valores são capazes de convencer que a mudança de metodologia possa ser benéfica aos processos e que a sua adoção acarretaria competitividade em relação ao mercado, visto que estes fatores influenciaram diretamente nos custos e tempos que são repassados aos clientes finais.

Ainda na opinião dos gestores, a principal questão que deve ser avaliada para adoção da metodologia, é a questão treinamento para utilização, que pode ser treinando parte da equipe e estes disseminando o conhecimento através do restante da equipe de trabalho. Esta seria a melhor maneira de adaptar esta mudança como parte da alteração de todos os processos internos da empresa; para que não influencie no desenvolvimento dos softwares que estão sendo realizados e não haja algum tipo de atraso no prazo de entrega. A tendência é pela adoção da metodologia, pois acredita-se válida a proposta, principalmente pela questão economia de custo de pessoal e de desenvolvimento que o estudo aponta, de acordo o relato do coordenador de projetos da empresa case, THOMAS (2005):

> *"A experiência adquirida dará condições para uma alta produtividade. Se o ambiente for organizado e controlado, esta produtividade dos colaboradores vai se somar à produtividade do ambiente. Não vejo as possibilidades de forma excludente. Se o objetivo é controle e garantia de atendimento de prazos, cronogramas e qualidade e principalmente a diminuição de custos internos, (e se eu tivesse que escolher uma) escolheria o approach metodológico, com embasamento teórico, pois garante resultados de forma mais previsível, mesmo que eventualmente mais demorado em alguns casos (constante, previsível e mensurável x incontrolável, surpreendente (positiva ou negatvaimente) e estressante)"*

CONCLUSÃO

No decorrer deste trabalho procurou-se explorar o modelo CoCoMo II, bem como as características e variáveis necessárias e que influenciam no sucesso da mudança de paradigma e implantação do modelo CoCoMo II. À luz de recomendações levantadas pelos autores na revisão bibliográfica, foram formulados os aspectos necessários para elaboração de estimativas com embasamento, que foram investigadas e aplicadas em um estudo de caso em uma empresa com perfil pré-definido e, finalmente, relatadas e discutidas no capítulo de análise de resultados.

Sobre o estudo de caso, pode-se inferir que grande parte das diferenças nos valores encontrados para as variáveis estudadas entre os produtos nas estimativas realizadas e nos resultados encontrados, justifica-se pelo fato de que os processos de desenvolvimento de software ainda estão sofrendo os efeitos da padronização ora sendo adotada. Assim, por exemplo, reutilização e modularização, podem acarretar a repetição de instruções escritas nos diferentes produtos estudados, ao invés de um encapsulamento de rotinas comuns a todos os produtos.

Os fatores levantados na revisão bibliográfica e as evidências colhidas no estudo de caso, levam concluir que o sucesso para implantação de uma metodologia para elaboração de estimativas com embasamento teórico, neste caso o modelo estudado, o CoCoMo II, depende, dentre outros aspectos, das seguintes variáveis levantadas no referencial teórico:

- Comparação de custo estimado pela empresa, o custo realmente alcançado e o custo proposto pela metodologia em projetos concluídos. Tal comparação é importante pois desta forma existe a possibilidade de verificar se há e onde há problemas nos processos de desenvolvimento de software, e desta forma ter a possibilidade de encontrar maneiras de melhorá-los ou atualizá-los.
- Custo de implementação da metodologia. Este fator é relevante e deve ser levado em conta, visto que em toda e qualquer mudança de paradigma ou metodologia, a questão custo e o impacto causado por essa mudança devem ser avaliados para verificar a sua viabilidade financeira.
- No aspecto recursos humanos, considerar o tempo de treinamento da equipe que utilizará o modelo, visto que é importante saber, além da capacidade da equipe em utilizar o modelo e elaborar estimativas, qual será a adaptabilidade dos integrantes da equipe de desenvolvimento a uma mudança. Desta maneira haverá maiores subsídios para gerentes e coordenadores controlar o andamento dos projetos, e do desenvolvimento dos produtos e das atividades realizadas pelos desenvolvedores.
- No aspecto tecnológico e de projetos, comparação do custo estimado de projetos em andamento ou em planejamento, não apenas em projetos concluídos, para justificar a mudança de paradigma na elaboração de estimativas. Isso viabiliza a possibilidade de aplicar a metodologia e o modelo CoCoMo II como forma de comparação e em projetos que estejam em andamento, possibilitando que haja uma

noção real de como a estimativa pode trazer benefícios aos processos de estimativa e elaboração do custo dos produtos e projetos, pois estando em andamento é possível direcionar o andamento do desenvolvimento em relação à estimativa, ou seja, se for detectado algum desvio, pode-se solucionar em meio ao desenvolvimento (iteração), e não trabalhar apenas com hipóteses, que é o resultado da aplicação em projetos concluídos.

Essas foram as variáveis que se apresentaram como as mais relevantes na elaboração do estudo de caso. Entretanto, é possível haver outras variáveis, como por exemplo, a receptividade da equipe à essa mudança, que podem influenciar nos resultados de planejamento e desenvolvimento de software, apresentando desta forma resultados diferentes e estimativas alteradas.

6.1 RESTRIÇÕES DO ESTUDO

Algumas dificuldades no decorrer da pesquisa, devem ser aqui relatadas.

- O modelo CoCoMo II é uma metodologia que pode ser capaz de gerar alguma vantagem competitiva às empresas que o adotam; sendo assim, a organização que serviu como caso no estudo, está inserida em um patamar e segmento altamente competitivo. Com isso, houve algumas restrições à coleta de dados, principalmente algumas informações que eram importantes para a elaboração das comparações entre as estimativas, como por exemplo, o custo efetivamente gasto no desenvolvimento dos produtos de software que estavam concluídos.
- Por ser um assunto relativamente novo e com recente repercussão nas empresas e na academia, alguns fatores que poderiam fazer parte das variáveis que podem ser consideradas na implantação do modelo CoCoMo II estão sendo abordados nos últimos meses e não constam neste trabalho. Por exemplo, pode-se citar a elaboração de estimativas para o desenvolvimento de softwares voltados à WEB como plataforma e canal de comunicação, abstraindo a cultura de cliente/servidor.

Com isso, salienta-se que o quadro de variáveis apresentado não é estático e que a cada momento podem ser levantadas novas variáveis com suas peculiaridades setoriais.

6.2 TRABALHOS FUTUROS

Acredita-se que o objetivo principal deste estudo foi atingido, uma vez ter sido possível elaborar uma metodologia para realizar estimativas de forma embasada, através da utilização do modelo CoCoMo II e descrever as variáveis que influenciam no sucesso da implantação desse modelo como metodologia nos processos de desenvolvimento de software em uma empresa. Sabe-se que por ser esse estudo de caráter exploratório, ficam abertas algumas discussões, que podem ser consideradas lacunas no conhecimento e que podem ser objeto de estudos futuros.

Um aspecto que pode gerar pesquisas futuras, em relação ao modelo CoCoMo II, é o estudo do impacto do CMMI sobre a maturidade da equipe de desenvolvimento de software, visto que hoje apenas o impacto do CMM sobre este aspecto é abordado.

Como o trabalho se propôs a descrever de maneira ampla uma metodologia de utilização e implantação do modelo CoCoMo II a um determinado perfil de

empresas, cabe ressaltar a importância da elaboração de metodologias específicas para implantação do modelo CoCoMo II em empresas que desenvolvam software, com perfis diferentes ao da empresa estudada, nos diversos segmentos da economia, adaptando-se aos processos de desenvolvimento que estas possuam.

Outra questão que pode ser aprofundada é a elaboração de um *framework*, para que o modelo estudado, a partir da metodologia elaborada, possa ser aplicado em empresas do mesmo perfil da empresa case.

Outro aspecto que pode ser abordado é em relação às variáveis para implantação de uma nova metodologia, como no caso do CoCoMo II, que podem alterar de organização para organização, de acordo com o histórico nos processos de desenvolvimento que cada uma contempla e implementa.

Estimativas em produtos e projetos que sejam desenvolvidos através da WEB é uma questão que poderia ser abordada, pela particularidade e metodologia de desenvolvimento em relação aos produtos cliente/servidor.

Ao final deste estudo percebe-se que o uso da estratégia de realizar o planejamento antes do efetivo desenvolvimento de um produto, deve ser a direção a ser seguida pelas empresas.

Também se pode inferir que para uma empresa mudar a metodologia dos seus processos de desenvolvimento e, no caso do presente trabalho, implantar o CoCoMo II, é necessário que se considere não apenas a tecnologia, mas a estratégia e as pessoas. Acredita-se que o presente trabalho seja um dos primeiros esforços no sentido de descrever uma metodologia para implantação do modelo de custos CoCoMo II para o desenvolvimento de software, contribuindo não somente à academia, mas também às organizações que possam vir a implantar tal metodologia.

REFERÊNCIAS BIBLIOGRÁFICAS

ABREU, Fernando Brito : **Modelo COCOMO: das origens à actualidade**, InterFace, nº6, Abril de 1998 a.

_____. **Do ADA COCOMO ao COCOMO 2**, InterFace, nº7, Maio de 1998 b.

AGUIAR, Maurício – **Estimando os Projetos com COCOMO II**. TIMétricas. Disponível em <http://www.timetricas.com.br> - Acesso em: 15 mai. 2004

AZEVEDO, Douglas José Peixoto de.; **FPA – Function Point Analysis – Sistema de Métrica**. Califórnia. Sage. 1998.

BFPUG. **Pontos de Função.** Disponível em <www.bfpug.com.br>. Acesso em 01/10/2004 às 15:30

BOEHM, Barry W.; **Software Engineering Economics**. Prentice Hall, 1981.

BOEHM, Barry et al.; **Software Cost Estimation With COCOMO II**. Prentice Hall, 2000.

COCOMO MANUAL. **Cocomo II Model Definition Manual.** Disponível em <http://sunset.usc.edu/research/COCOMOII>. Acesso em 15/08/2004 às 20:00.

CHULANI, Sunita Devnani; **Bayesian Analysis of Software Cost and Quality Models**.. Dissertação (Doutorando em Filosofia das Ciências da Computação.) University of Southern California (USC) 1999

CMU/SEI. CARNEGIE MELLON UNIVERSITY, SOFTWARE ENGINEERING INSTITUTE. **The Capability Maturity Model: Guidelines for Improving the Software Process.** Addison Wesley Longman, 1997.

DIGICON. **Sistemas de Bilhetagem.** Disponível em <http://www.digicon.com.br,> Acesso em 30/01/2005 às 16:25.

DINSMORE, Paul Campbell; **Gerência de Projetos e Programas**. São Paulo. Pini, 1992.

PONTOS de FUNÇÃO. **Estimando o tamanho com FPA.** Disponível em <http://www.celepar.br/batebyte/bb70/fpa.htm>. Acesso em 01/10/2004 às 9:50.

GIL, Antonio Carlos; **Métodos e Técnicas de Pesquisa Social**. São Paulo. Atlas 1994.

GONÇALVES, Laís H.V.B. **Métricas com Pontos de Função.** Disponível em http://www.enupf.com.br>. Acesso em 10/10/2004 às 20:45

HUMPHREY, Watts. S.; **A discipline for Software Engineering**. Addison, Wesley, 1995.

ISO. **Sistema da Qualidade.** Disponível em <http://www.iso.org/iso/en/ISOOnline.frontpage>. Acesso em 05/10/2004 às 20:00.

_____.**O que é a ISO.** Disponível em <http://www.iso.org/iso/en/ISOOnline.frontpage>. Acesso em 06/10/2004 às 7:40.

JONES, Capers; **A Short History of Functions Points and Feature Points Software Productivity Research**. Inc. Beurlington MA, 1986.

KIVAL, Chaves Weber; ROCHA, Ana Regina Cavalcanti; **Qualidade e Produtividade em Software**. 3ª Edição, Makron Books, 1999.

KRUTCHEN. Philippe, **Introdução ao RUP: Rational Unified Process.** São Paulo. Ciência Moderna 2003

LEITE, J.C.S.P. et al, **Enhancing a Requirements Baseline with Scenarios**. **Requirements Engineering**, Springer-Verlag London, 1997.

PMBOK. **Tradução da 5ª Edição.** Disponível em <http://www.pmirs.org/PMI20.htm>. Acesso em 06/10/2004 às 15:00.

PMI. **Guia de Referência.** Disponível em <http://www.pmirs.org>. Acesso em 10/10/2004 às 21:50.

PRADO, Darcy**, Gerenciando Projetos nas Organizações**, Belo Horizonte MG, EDG, 2000.

PRESSMAN, Roger S.; **Engenharia de Software.** São Paulo. Atlas 1995.

PUTNAM, Lawrence H.. A General Empirical Solution to the Macro Software Sizing and Estimating Problem. In _____. **Putnam Estimates** IEEE Transactions Software Engineering, V.4 N.4, 1978. p 345 – 361.

RATIONAL UNIFIED PROCESS. **Using the Rational Unified Process.** Disponível em <http://www.rational.com./rup>. Acesso em 15/10/2004 às 23:00.

SEI. **CMM Definitions and KPA's.** Disponível em <http://www.sei.cmu.edu>. Acesso em 08/11/2004 às 15:30.

THOMAS, Marcelo. **Padrão para Estimativas** [mensagem pessoal]. Mensagem recebida por <plopez@digicon.com.br> em 24 maio 2005

TRINDADE, André Luiz P.; PESSÔA, Marcelo Scheneck P.; SPÍNOLA, Mauro; **COCOMO II uma compilação de informações sobre a nova métrica.** In: Congreso Internacional de Ingeniería Informática, 5., 1999. Buenos Aires. **Anais...**Buenos Aires: Universidad de Buenos Aires, 1999.pp. 24-38.

UFRGS: **TQC – Controle de Qualidade Total.** Disponível em <http://www.rcgg.ufrgs.br/cap14.htm>. Acesso em 11/02/2005 às 14:15

YIN, Robert K.; **Case Study Research, Design and Models.** California. Sage. 1994

ANEXOS

Anexo 1 – Tabelas com os valores dos fatores de escala, multiplicadores de esforço e peso das linguagens utilizados nos cálculos

- **Fatores de Escala**

Tabela 14 – Valores dos Fatores de Escala

Fatores de Escala	Muito Baixo	Baixo	Normal	Alto	Muito Alto	Extra Alto
PREC	6.20	4.96	3.72	2.48	1.26	0
FLEX	5.07	4.05	3.04	2.03	1.01	0
RESL	7.07	5.65	4.24	2.83	1.41	0
TEAM	5.48	4.38	3.29	2.19	1.10	0
PMAT	7.80	6.24	4.68	3.12	1.56	0

Fonte: COCOMO MANUAL (2004)

- **Multiplicadores de Esforço**
  - **Early Design**

Tabela 15 – Valores dos Multiplicadores de Esforço para o modelo Early Design

Multiplicador	Extra Baixo	Muito Baixo	Baixo	Normal	Alto	Muito Alto	Extra Alto
RCPX	0.73	0.81	0.98	1	1.30	1.74	2.38
RUSE	-	-	0.95	1	1.07	1.15	1.24
PDIF	-	-	0.87	1	1.29	1.81	2.61
PERS	2.12	1.62	1.26	1	0.83	0.63	0.5
PREX	1.59	1.33	1.12	1	0.87	0.71	0.62
FCIL	1.43	1.30	1.10	1	0.87	0.73	0.62
SCED	-	1.43	1.14	1	1	1	-

Fonte: COCOMO MANUAL (2004)

  - **Post- Architecture**

Tabela 16 – Valores dos Multiplicadores de Esforço para o modelo Post-Architecture

Multiplicador	Muito Baixo	Baixo	Normal	Alto	Muito Alto	Extra Alto
RELY	0.82	0.92	1	1.10	1.26	-
DATA	-	0.90	1	1.14	1.28	-
CPLX	0.73	0.87	1	1.17	1.34	1.74
RUSE	-	0.95	1	1.07	1.15	1.24
DOCU	0.81	0.91	1	1.11	1.23	-
TIME	-	-	1	1.11	1.29	1.63
STOR	-	-	1	1.05	1.17	1.46

PVOL	-	0.87	1	1.15	1.30	-
ACAP	1.42	1.19	1	0.85	0.71	-
PCAP	1.34	1.15	1	0.88	0.76	-
PCON	1.29	1.12	1	0.90	0.81	-
AEXP	1.22	1.10	1	0.88	0.81	-
PEXP	1.19	1.09	1	0.91	0.85	-
LTEX	1.20	1.09	1	0.91	0.84	-
TOOL	1.17	1.09	1	0.90	0.78	-
SITE	1.22	1.09	1	0.93	0.86	0.80
SCED	1.43	1.14	1	1	1	-

Fonte: COCOMO MANUAL (2004)

- **Peso das Linguagens**

Tabela 17 – Peso das linguagens para conversão de PFA a linhas de código

Linguagem	SLOC/PFNA
Ada	71
AI Shell	49
APL	32
Assembly	320
Assembly (Macro)	213
ANSI / Quick / Basic	64
Basic – Compiled	91
Basic Interpreted	128
C	128
C++	29
Visual Basic	32
ANSI Cobol 85	91
Fortran 77	105
Forth	64
Jovial	105
Lisp	64
Modula 2	80
Pascal	91
Prolog	64
Report Generator	80
Spreadsheet	6

Fonte: COCOMO MANUAL (2004)