UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MARISTELA CORRÊA MELLER

MODELOS PARA ESTIMAR CUSTOS DE SOFTWARE: ESTUDO COMPARATIVO COM SOFTWARES DE PEQUENO PORTE

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

PROF. ANTONIO CEZAR BORNIA, Dr
Orientador

Florianópolis, junho de 2002

MODELOS PARA ESTIMAR CUSTOS DE SOFTWARE: ESTUDO COMPARATIVO COM SOFTWARES DE PEQUENO PORTE

MARISTELA CORRÊA MELLER

Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Ciência da Computação, Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Fernando Álvaro Ostuni Gauthier, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Antonio Cezar Bornia, Dr.
Presidente da Banca (Orientador)

Prof. João Bosco da Mota Alves, Dr.

Prof. Vitório Bruno Mazzola, Dr.

AGRADECIMENTOS

A Deus, por ter sido o refúgio quando os problemas pareciam insolúveis.

Ao orientador, Prof. Antonio Cezar Bornia, pela disposição e atenção dedicadas na elaboração deste trabalho.

Aos dirigentes, colegas professores, funcionários e alunos da UNOESC, Campus de São Miguel do Oeste que, direta ou indiretamente, contribuíram para a realização deste trabalho.

Aos professores e funcionários da UFSC, pela atenção que sempre dispensaram.

À UNOESC, Campus de São Miguel do Oeste, pela ajuda financeira concedida através do PICDTU-Programa Institucional de Capacitação Docente e Técnica da UNOESC.

Em especial, a Itamar Leite de Oliveira, antes de tudo, o companheiro de todas as horas, o incentivador incondicional e o apoio nos momentos certos, pelo carinho, pela paciência, pelas idéias e, por que não dizer, pelas críticas que contribuíram para a melhoria deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho faz uma análise comparativa entre as principais técnicas (Julgamento Especialista, Analogia e Modelos Algorítmicos) e os principais modelos (Linhas de Código, Modelo de Estimativa de Putnam, COCOMO, Pontos de Função, Pontos de Particularidade, Ciência do Software de Halstead e Número Ciclomático de McCabe) usados para estimativa de custo de software. Realiza estudos multicaso para comparar valores reais e estimados considerando alguns Modelos Algorítmicos. Os softwares avaliados são divididos em três tipos: Comércio Eletrônico (dois casos), Inteligência Artificial (dois casos) e Sistemas de Informação (três casos) com o objetivo de verificar, no estudo de caso, possíveis relações entre estes tipos e os modelos estudados. A partir desta análise são indicados os modelos mais adequados para cada tipo de software avaliado.

Palavras-chave

Custo de software, estimativa de custo de software, métricas de software.

ABSTRACT

This work brings a comparative analysis among the main techniques (Expert Judgement, Analogy and Algorithmic Models) and the main models (Lines of Code, Putnam's Estimate Model, COCOMO, Function Points, Feature Points, Halstead's Software Science and McCabe's Cyclomatic Number) used for estimating software costs. It presents a multicase study to compare real and estimated values considering some Algorithmic Models. The evaluated softwares are divided in three types: E-commerce (two cases), Artificial Intelligence (two cases) and Information Systems (three cases) with the objective of verifying, in the case study, possible relationships between these types and the studied models. Starting from this analysis the most suitable models for each type of evaluated software are indicated.

Key-words

Software costs, estimating software costs, software metrics.

1. INTRODUÇÃO

1.1. FORMULAÇÃO DO PROBLEMA

Segundo Cruz, Werner e Soares (1999), as estimativas de custo de projetos de desenvolvimento de software, regularmente, são subestimadas e, por isso, em muitas organizações não possuem credibilidade.

A medição de software em si é problemática, pois não há um consenso sobre a técnica ou o modelo a ser utilizado. Alguns pesquisadores defendem a contagem da quantidade de linhas de código-fonte, outros defendem a contagem de pontos de função, mas há os que afirmam que a quantidade de linhas de código de um software é muito dependente da linguagem programação utilizada e ainda há os que afirmam que os pontos de função apresentam uma unidade de medida de difícil comparação (por exemplo, quando se vê uma placa de trânsito "lombada a 200 metros", o motorista do veículo sabe, com razoável precisão, onde está a lombada, mesmo que não a esteja vendo, mas quando alguém diz que "faltam 50 pontos de função para o término do projeto", é muito dificil dizer se vai demorar ou não). Como diz Pressman (1995), o ponto de função "não tem nenhum significado físico direto - é apenas um número".

Como o desenvolvimento de software pode demorar anos, problemas na coleta de dados podem acontecer, principalmente por baixa motivação. Esses problemas podem ser ainda maiores quando a medição tem como objetivo realizar a estimativa de custos de software (especialmente a estimativa de prazos e de esforço de desenvolvimento), provavelmente porque a equipe de desenvolvimento (responsável pelo fornecimento de parte dos dados) teme estar sendo avaliada para ser punida de alguma forma futuramente.

Para Hazan (2001), o processo de medição deve ser integrado com o processo de desenvolvimento de software e os resultados da medição devem ser compartilhados com as equipes de desenvolvimento.

Calvert (1996) afirma que as métricas são ferramentas de administração usadas para estimar o custo e as exigências de recurso de um projeto de software.

A literatura especializada apresenta várias técnicas para a estimativa de custos de software e não há um consenso sobre a totalidade das técnicas. A maioria dos autores, entretanto, concorda com a utilização destas três: Julgamento Especialista (também conhecida como Parecer Técnico), Analogia e Modelos Algorítmicos. Esta última, na realidade, é um conjunto de modelos, dentre os quais se destacam: Linhas de Código, Modelo de Estimativa de Putnam, Modelo de Custo Construtivo (COCOMO), Pontos de Função, Pontos de Particularidade, Ciência do Software de Halstead e Número Ciclomático de McCabe.

Cada uma dessas técnicas apresenta vantagens e desvantagens e, tanto Boehm (1981), quanto Shepperd, Schofield e Kitchenham (1996) afirmam que se deve usar mais do que uma técnica, por exemplo, parecer técnico e um dos modelos algorítmicos. Não fica claro, porém, qual deva ser utilizado.

Propõe-se, então, o seguinte problema de pesquisa: "**Quais dos modelos algorítmicos são mais acurados em suas estimativas?**"

1.2. OBJETIVOS

1.2.1. Objetivo Geral

Comparar a aplicação dos modelos algorítmicos de Linhas de Código, Modelo de Estimativa de Putnam, Modelo de Custo Construtivo (COCOMO) Básico, Pontos de Função e Pontos de Particularidade na estimação de esforço de desenvolvimento de software.

1.2.2. Objetivos Específicos

- Estudar os principais métodos existentes para medição e estimativa do custo de software.
- Analisar comparativamente estes métodos apontando suas vantagens e desvantagens.
- Aplicar alguns dos modelos algorítmicos estudados em diferentes domínios de aplicação de software de pequeno porte.

- Analisar os resultados obtidos, no que se refere às dificuldades na aplicação dos modelos e diferenças encontradas.

1.3. JUSTIFICATIVA

Para um gerente de projetos de desenvolvimento de software, o desconhecimento de quantitativos como o prazo de duração do projeto, alocação de recursos e o esforço a ser empregado é, no mínimo, preocupante. Um erro nesses quantitativos pode levar ao completo caos (WEBER, ROCHA e NASCIMENTO, 2001).

Este estudo é importante devido à limitação dos recursos financeiros para suprir a demanda de novos produtos de software. A estimativa do custo do software torna-se uma prioridade para a maioria das empresas, tanto produtoras, quanto consumidoras de software que, através de uma análise dos custos de desenvolvimento dos produtos propostos podem selecionar e/ou estabelecer prioridades para a construção destes produtos (CRUZ, WERNER e SOARES, 1999).

A análise dos resultados obtidos com este estudo pode indicar quais os modelos mais adequados para os tipos de software avaliados.

Deve ser considerada, ainda, a importância teórica deste estudo, em virtude da escassez de literatura, fato que pode ser comprovado pela seção “Referências” deste trabalho: os autores se repetem (apenas mudam os grupos) mostrando que são poucos os pesquisadores do tema e, provavelmente, como conseqüência, os livros encontrados sobre o assunto são também reduzidos (boa parte das referências foram localizadas na internet). Além disso, nenhum estudo sobre a possível relação entre os modelos e tipos de software foi localizado. Ressalta-se assim a contribuição teórica deste trabalho como base para futuros estudos.

1.4. METODOLOGIA

A partir de uma revisão bibliográfica sobre o tema central deste trabalho, fez-se um estudo comparativo entre algumas técnicas e alguns Modelos Algorítmicos indicados pela literatura

como sendo adequados para a estimativa de custos de software, com o objetivo de identificar vantagens e desvantagens de cada um.

Neste trabalho foi realizada, ainda, uma pesquisa exploratória com “estudos exploratório-descritivos combinados”, conforme explicado por Marconi e Lakatos (1999), partindo de um estudo de caso. Cabe ressaltar que foram avaliados vários casos, ou seja, vários softwares.

Para Chizzotti (2000), “quando se toma um conjunto de casos, a coleção deles deve cobrir uma escala de variáveis que explicite diferentes aspectos do problema” e Marconi e Lakatos (1999) afirmam que para esse tipo de pesquisa os procedimentos de amostragem são flexíveis, porque se deve dar prioridade ao caráter representativo dos casos. Assim, os softwares foram selecionados a partir dos seguintes critérios:

- existência e adequação de uma documentação mínima necessária sobre os softwares;
- disponibilidade dos desenvolvedores em fornecer a documentação do software, em especial os projetos lógico e físico;
- pré-disposição dos desenvolvedores para eliminação de eventuais dúvidas quanto à documentação fornecida;
- disponibilidade dos desenvolvedores em fornecer uma listagem impressa do código-fonte;
- possibilidade de enquadramento dos softwares em alguns domínios de aplicação, como por exemplo, Comércio Eletrônico, Inteligência Artificial (e Otimização) e Sistemas de Informação (convencionais);
- disponibilidade de, no mínimo, dois softwares para cada domínio de aplicação.

A partir desses critérios foram selecionados dois softwares de Inteligência Artificial, dois softwares de Comércio Eletrônico e três softwares de Sistemas de Informação, adequadamente caracterizados na seção 4.1 deste trabalho.

Para cada software são verificados os valores reais e calculados os valores estimados. Da comparação destes valores (reais e estimados) para cada um dos Modelos Algorítmicos, considerando ainda os tipos de software (domínios de aplicação) selecionados, pode-se fazer algumas inferências, como por exemplo, a adequação (ou não) de determinado modelo para determinadas estimativas de custo em tipos específicos de software.

1.5. LIMITAÇÕES

Das técnicas de estimativa de custos de software citadas na seção 1.1, foi aplicada apenas a técnica dos Modelos Algorítmicos, isto porque não se dispunha de condições para implementar outras técnicas.

Considerando os Modelos Algorítmicos apresentados neste trabalho, optou-se pela utilização de Linhas de Código, Modelo de Estimativa de Putnam, Modelo de Custo Construtivo (COCOMO) Básico, Pontos de Função e Pontos de Particularidade. As variações Intermediário e Detalhado do Modelo COCOMO, a Ciência do Software de Halstead e o Número Ciclomático de McCabe não foram aplicados por, pelo menos um, dos seguintes motivos:

- são problemáticos, pouco utilizados e/ou pouco adequados para estimativas;
- não medem esforço e/ou tempo de desenvolvimento;
- pouca literatura sobre o modelo e sua aplicação;
- são de díficil aplicação.

1.6. ESTRUTURA

Este trabalho está estruturado da seguinte forma: o Capítulo 1 é um capítulo introdutório, onde são apresentados o problema, os objetivos do trabalho, sua justificativa, suas limitações e a metodologia que foi seguida. O Capítulo 2 fala de custos de software, traz alguns conceitos fundamentais de contabilidade de custos, custeio de ciclo de vida, custos de software e custos para controle. Trata ainda das principais técnicas de estimativa de custo de software e principais modelos algorítmicos utilizados na medição de softwares. O Capítulo 3 apresenta uma análise comparativa das técnicas e dos modelos estudados. O Capítulo 4 mostra um estudo de caso (com vários casos) onde são selecionados e aplicados alguns dos modelos estudados, fazendo-se uma análise que procura detectar os modelos mais adequados a cada tipo de software medido. As Conclusões e Recomendações são apresentadas no Capítulo 5.

2. CUSTOS DE SOFTWARE

2.1. CONCEITOS FUNDAMENTAIS DE CONTABILIDADE DE CUSTOS

Conforme Bornia (2002) e Perez Jr, Oliveira e Costa (1999), o **custo de fabricação** é o valor dos insumos efetivamente utilizados na fabricação de um determinado produto, ou seja, o valor dos materiais, máquinas e equipamentos, custos gerais de fabricação (por exemplo, energia elétrica, água e depreciação), mão-de-obra produtiva e mão-de-obra de gerência e supervisão, serviços de apoio à produção (por exemplo, manutenção, almoxarifado, refeitório), entre outros, realmente envolvidos no processo de fabricação do produto em questão. Geralmente este custo é subdividido em:

- **Custos de Matéria-Prima (MP)**: na prática, apenas o custo dos materiais mais relevantes é considerado (em termos de custo). Os materiais de custo reduzido tendem a ser considerados como materiais de consumo e são analisados simplificadamente.
- **Custos de Mão-de-Obra Direta (MOD)**: são os custos relacionados aos trabalhadores (por exemplo, salários, encargos sociais, provisões de férias e décimo-terceiro salário) envolvidos diretamente com o processo produtivo.
- **Custos Indiretos de Fabricação (CIF)**: são todos os custos de fabricação que não puderam ser classificados como MP, nem como MOD (por exemplo, materiais de consumo, mão-de-obra indireta, depreciação, água, energia elétrica, telefone, entre outros) e, normalmente, são alocados por intermédio da utilização de algum critério de rateio.

2.2. CUSTEIO DE CICLO DE VIDA

Para Sakurai (1997), o custeio de ciclo de vida considera os custos durante toda a vida útil do produto ou equipamento em questão. Já para o CRC-SP (1995) o custo do ciclo de vida do produto tem duas fases:

- **custo do ciclo de vida para o produtor**: são todos os custos suportados pelo fabricante durante o ciclo de vida do produto, desde o surgimento da idéia (por exemplo, pesquisa, concepção, projeto, desenvolvimento, protótipo, produção, lançamento, distribuição,

atendimento ao cliente, garantia, propaganda, retirada de mercado e ainda fabricação das peças de reposição).

- **custo do ciclo de vida para o consumidor**: são todos os custos suportados pelo cliente para obter, usar e dispor do produto (por exemplo, o preço de compra mais o custo de transporte, instalação, operação, manutenção, serviços de apoio, modificações e descarte). O cliente normalmente escolhe o produto que minimiza a combinação destes itens.

Segundo Sakurai (1997), normalmente as empresas se preocupam com os custos de produção (as empresas produtoras), ou com os custos apenas de aquisição (a empresas usuárias dos produtos), despreocupando-se, às vezes, com os custos de pós-aquisição (custos de operação, manutenção e descarte). Possivelmente isso aconteça em virtude desses custos serem relativamente pequenos. Para produtos de tecnologia como aeronaves, computadores e software, porém, freqüentemente, os custos incorridos após a aquisição são maiores que os custos de aquisição.

Conforme o CRC-SP (1995) a empresa produtora deve basear suas decisões de investimentos tanto no custo do ciclo de vida para o produtor, quanto no custo do ciclo de vida para o consumidor.

Ao se comparar o ciclo de vida do produto apresentado por CRC-SP (1995) e o paradigma de desenvolvimento de software conhecido como ciclo de vida clássico apresentado por Pressman (1995), é possível uma equivalência de etapas, como mostrado na Figura 1.

2.3. CUSTOS DE SOFTWARE

Sakurai (1997) afirma que a indústria japonesa conseguiu atingir um alto grau de automação industrial, provavelmente em decorrência do fato de que utiliza aproximadamente 60% dos robôs existentes no mundo. Tanto os robôs, quanto alguns tipos de máquina, no entanto, não podem ser bem utilizados se não possuírem software de alta qualidade. O custo de software pode representar 40% do custo de automatização de uma fábrica. Em alguns casos específicos este valor pode chegar a 70% do custo total dos equipamentos. Em razão disso, a ênfase do gerenciamento de custos está mudando de foco (do hardware para o software). Conforme Sakurai (1997), o desenvolvimento de software envolve determinação das necessidades do

cliente, consideração de necessidades da engenharia de produção, do desenvolvimento de novos produtos, da produção e de marketing. Além disso, os prazos devem ser cumpridos e a qualidade garantida.

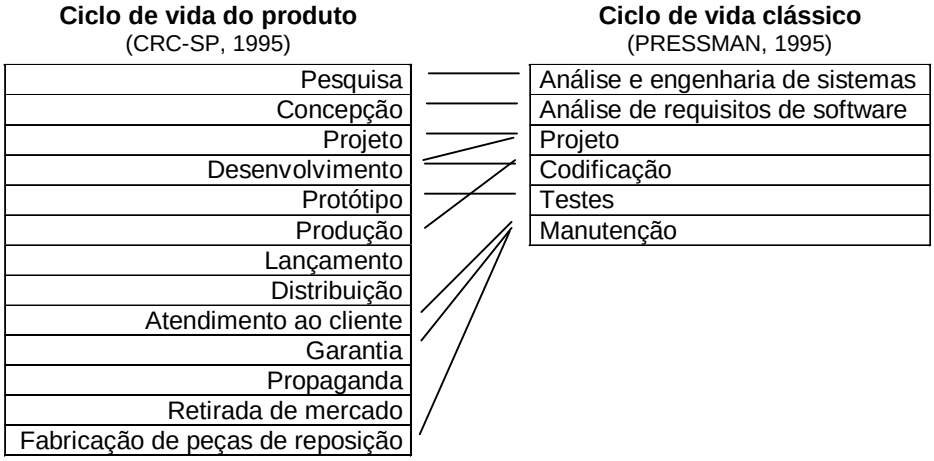

Figura 1. Equivalência de etapas do ciclo de vida do produto e do ciclo de vida clássico (do software)
Fonte: CRC-SP (1995) e Pressman (1995)

O software personalizado é desenvolvido para uma finalidade específica e tem um custo alto a curto prazo, mas a longo prazo este tipo de software pode levar a um aumento de produtividade, pois atende melhor as necessidades específicas da empresa usuária.

Sakurai (1997) lança um questionamento: na contabilidade de custos, como o software deve ser considerado: **bem** ou **serviço**? O MITI-Ministério do Comércio e da Indústria Internacional define software como **serviço**, mas para alguns autores esta definição é questionável. Muitos especialistas japoneses em desenvolvimento de software dizem que a indústria de software é **manufatureira**. Presmann (1995) afirma que tanto a atividade de desenvolvimento de hardware quanto a de software, exigem a construção de um **bem**, mas que as abordagens são diferentes.

Conforme Sakurai (1997), para efeito de custos, o software pode ser assim categorizado:

- Tangível (os pacotes de software);
- Intangível (os serviços de manutenção de software);
- Tangível **e** Intangível (os softwares personalizados).

Caso um serviço de software seja classificado como **intangível**, isto requererá, provavelmente, uma contabilidade de custos diferente dos sistemas em uso. Para Pressman (1995) e Sakurai (1997), como o software é considerado mais como um trabalho de produção intelectual, os sistemas de custos usados para hardware podem ser virtualmente aplicados também a software. Há que se considerar, entretanto, algumas particularidades do software:

- o software é um elemento de sistema lógico, e não físico, ou seja, o software é um produto que não apresenta substância física, o que torna o gerenciamento de custos mais difícil, sem contar que é muito importante medir adequadamente o esforço de desenvolvimento;
- a estrutura dos custos é diferente porque os custos de mão-de-obra são muito altos e os custos de material são muito baixos, sem contar que "o software não se desgasta";
- para o desenvolvimento de software a relação entre o volume de entrada (de insumos) e de saída (do produto) não é muito clara e ainda depende muito do talento dos profissionais desenvolvedores;
- a maioria dos softwares é feita sob medida (software personalizado) em vez de ser montada a partir de componentes existentes.

Ao se considerar o ciclo de vida de um software, as etapas de desenvolvimento são o que se tem de mais importante e onde se emprega o maior esforço. A MOD - Mão-de-Obra Direta, portanto, é o item principal quando se fala de custos de software, ou seja, a MOD é responsável pela maior parte dos custos.

2.4. CUSTOS PARA CONTROLE

Conforme Perez Jr, Oliveira e Costa (1999), **custo-padrão** é "aquele determinado *a priori*, como sendo o custo normal de um produto".

Para Bornia (2002), o objetivo dos **custos para controle** é fornecer um custo-padrão para compará-lo com os custos realmente apurados. Tal método apresenta as seguintes etapas:

- fixar um custo-padrão;
- determinar o custo real;
- verificar a variação entre o padrão e o real;
- analisar a variação.

Perez Jr, Oliveira e Costa (1999) dizem que o cálculo do custo-padrão é uma tarefa que deve ser realizada com base em estudos e análises prévias e é caracterizado pela fixação antecipada dos custos de toda uma linha de produção ou de cada produto em separado. Segundo Bornia (2002), o método do custo-padrão pode ser utilizado, dependendo da conveniência, em todos os custos da empresa, ou somente aos custos de MP, ou de MOD, ou dos insumos mais relevantes.

Conforme Sakurai (1997), o desenvolvimento de software é um trabalho de mão-de-obra intensiva e que o custo-padrão pode ser eficaz para controlar estes custos. Entretanto, o estudo de tempos e movimentos (eficaz na indústria convencional) não pode ser feito para software em virtude de o desenvolvimento de software ser de natureza intelectual e mudar a cada projeto. Os japoneses, por exemplo, constroem modelos de custos usando a experiência no desenvolvimento de software e a grande quantidade de dados que possuem, porque é mais fácil do que usar métodos de medição do trabalho.

Sakurai (1997) ainda afirma que para software não é necessário especificar um custo-padrão para materiais diretos, porque a parcela mais relevante do custo é a MOD. Vários fatores atrapalham o uso do custo-padrão em software:

- definir um custo-padrão para cada software é muito difícil e consome muito tempo, porque cada software é produzido individualmente;
- padronizar a mão-de-obra intelectual é muito delicado;
- acompanhar a evolução das técnicas de desenvolvimento de software não é compatível com a fixação do padrão.

Para Boehm (1981), quando um projeto de software é iniciado, é essencial juntar dados sobre seus custos atuais e seu progresso, como também comparar estas informações com as estimativas. Algumas razões para esta necessidade seriam:

- os dados de entrada para a estimativa são imperfeitos;
- as técnicas de estimativa são imperfeitas;
- alguns projetos não se ajustam exatamente ao modelo de estimativa;
- os projetos de software tendem a ser voláteis: componentes são adicionados e/ou imprevistos acontecem e estes fatores interferem no progresso dos projetos;

- o desenvolvimento de software é um campo em evolução: as técnicas de estimativa são calibradas previamente para projetos, que podem não ser caracterizados por programação estruturada, ajudas automatizadas, especificação de linguagens, microprocessadores, ou processamento de dados distribuído.

Hazan (2001) afirma ainda que o processo de medição deve ser um processo contínuo e sujeito a melhorias.

Ao considerar essas questões pode-se ter uma idéia da complexidade do problema que enfrentam os engenheiros de software e gerentes de projeto no momento de realizar a estimativa dos custos de cada um dos softwares que se encontram sob sua responsabilidade.

2.5. ESTIMATIVA DE CUSTOS DE SOFTWARE

Segundo Fenton e Pfleeger (1997), uma estimativa é uma avaliação de probabilidade. Uma estimativa só é útil se for razoavelmente precisa. Não se espera que uma estimativa seja exata, mas que seja precisa o suficiente para que se tenha segurança de fazer julgamentos e tomar decisões, considerando os limites do intervalo de confiança.

Quando engenheiros de software e gerentes de projeto falam de “estimativa de custo”, normalmente eles estão falando em estimar a provável quantidade necessária de esforço e tempo para o desenvolvimento de um projeto de software. Este uso é justificado pelo fato de que as despesas com pessoal, freqüentemente, dominam o custo global do projeto (FENTON e PFLEEGER, 1997).

O termo **estimativa de custo** pode ser utilizado como um termo “guarda-chuva” para vários tipos de estimativa, mas normalmente é sinônimo de **estimativa de esforço** (FENTON e PFLEEGER, 1997) e assim será considerado neste trabalho.

Com certeza, o tamanho do esforço de desenvolvimento afetará significativamente o custo do projeto. Além disso, provavelmente a intuição conte como um fator adicional, além da experiência e da capacidade da equipe de desenvolvimento, que também tem um grande impacto (ROETZHEIM, 2000b).

Shepperd, Schofield e Kitchenham (1996) destacam que a indústria de engenharia de software possui um registro histórico de dados muito pobre para estimar esforço e prazo de entrega. Os gerentes de projeto e as organizações de desenvolvimento de software, no entanto, têm uma grande necessidade de estimativas nas fases iniciais do ciclo de vida de cada projeto. Alguns exemplos desta necessidade seriam: apresentar propostas apropriadas de negócio e administrar corretamente os recursos.

Conforme Roetzheim (2000b), as pesquisas de Capers Jones indicam que o uso de técnicas formais de estimativa pode dobrar a probabilidade do projeto de software ser concluído com sucesso. Muitas pessoas e empresas percebem a importância da estimativa para os projetos de software, mas poucos entendem que estimativa de software pode ser aprendida. Isto é possível para estimativas consistentes e precisas de custos de desenvolvimento e prazos para uma ampla variedade de projetos.

De acordo com Hazan (2001), Calvert (1996) e Rezende (1999), as principais razões para se medir software são:

- formar uma *baseline* para estimativas;
- verificar se as metas de produtividade e qualidade estão sendo atingidas;
- avaliar as vantagens do uso de novos métodos e ferramentas de engenharia de software;
- melhorar o relacionamento com o cliente;
- ajudar na justificativa de pedidos de treinamento e aquisição de novas ferramentas;
- melhorar a gerência de contratos de software;
- reduzir o risco do estabelecimento de um cronograma inviável;
- melhorar a gerência de projetos de desenvolvimento de software.

Jones (1998) afirma que, além da necessidade de estimativas de custo de software precisas para operações empresariais cotidianas, estas estimativas estão se tornando um aspecto significativo em questões judiciais. Durante alguns anos, Capers Jones e seus colegas observaram vários processos judiciais onde as estimativas de custo de software foram produzidas pelos demandantes, pelos acusados, ou por ambos, como parte fundamental em processos que envolvem:

- rompimento de contrato entre contratados e contratantes;
- valor tributável de ativos de software;
- recuperação do excesso de custos devido a expansão do escopo;
- favoritismo na emissão de contratos de software;
- demissão injusta de pessoal de software.

2.6. PRINCIPAIS ATIVIDADES NA ESTIMATIVA DE CUSTO DE SOFTWARE

Boehm (1981) esclarece que a estimativa de custo de software deve ser planejada, revista e seguida. Com o intuito de detalhar esta tarefa são apresentadas algumas atividades consideradas básicas para o sucesso da estimativa.

2.6.1. Estabelecimento dos Objetivos

É importante estabelecer os objetivos da estimativa de custo como primeiro passo e usar estes objetivos para direcionar o nível de detalhamento e o esforço requerido para desenvolver os passos subseqüentes (BOEHM, 1981).

Segundo Hazan (2001), esses objetivos devem ser claros e bem definidos.

As três maiores diretrizes, conforme Boehm (1981), para o estabelecimento dos objetivos de uma estimativa de custo são:

- usar as necessidades de informação para a tomada de decisão na organização para fundamentar os objetivos da estimativa;
- equilibrar os objetivos de estimativas para os vários componentes do sistema;
- reexaminar os objetivos da estimativa no decorrer do processo e modificá-los onde for necessário.

2.6.2. Planejamento para Recursos e Dados Requeridos

Para Hazan (2001), o processo de medição deve ser fortemente acoplado ao processo de gerência da qualidade e integrado com planos e orçamentos.

Se a atividade de estimativa de custo de software é considerada como um miniprojeto, então este problema automaticamente será resolvido pela geração de um plano de projeto nas primeiras fases. A Tabela 1 mostra uma forma simples e geral de um plano de projeto que se aplica muito naturalmente ao miniprojeto de estimativa de custo (BOEHM, 1981).

Tabela 1. Planejamento do miniprojeto de estimativa de custo de software

Num.	Atividade	Descrição
01	Propósito	Por que a estimativa está sendo feita?
02	Produtos e prazos	O que vai ser fornecido e quando?
03	Responsabilidades	Quem é responsável por cada produto? Onde eles vão fazer o trabalho organizacionalmente? Geograficamente?
04	Procedimentos	Como o trabalho vai ser feito? Quais ferramentas e técnicas de estimativa de custo serão usadas?
05	Recursos requeridos	Quantos dados, tempo, dinheiro e/ou esforço são necessários para fazer o trabalho?
06	Suposições	Sob que condições se está prometendo entregar as estimativas, considerando os recursos anteriores (disponibilidade de pessoal fundamental, tempo de computador, dados de usuário)?

Fonte: Boehm (1981)

2.6.3. Cumprimento dos Requisitos de Software

Se não se conhece quais produtos estão sendo construídos, certamente não se pode estimar muito bem o custo desta construção. Isto significa que é importante ter um conjunto de especificações de software que devem ser não ambíguas e possíveis (BOEHM, 1981).

2.6.4. Consideração de Tantos Detalhes Quanto Possível

Segundo Boehm (1981), em geral, quanto mais detalhes considera-se nas atividades de estimativa, mais precisas essas estimativas serão, por três razões principais:

- quanto mais detalhes são explorados, melhor se entende os aspectos técnicos do software a ser desenvolvido;
- quanto mais partes do software são estimadas, mais se adquire as regras de trabalho para se reduzir a discrepância da estimativa. Se uma parte grande do software tem seu custo superestimado em 20%, tem-se um erro de 20%. Se esta parte grande é dividida em 10 partes menores, pode-se subestimar a maioria das partes, mas superestima-se algumas, e

finalmente acontece um equilíbrio para cima, com um erro de estimativa consideravelmente menor;

- quanto mais se pensa em todas as funções que o software tem que executar, o menos provável é que se perca os custos de alguns dos componentes do software.

2.6.5. Uso de Várias Fontes e Técnicas Independentes

De acordo com Boehm (1981), as principais técnicas disponíveis para estimativa de custo de software são:

- **Julgamento Especialista (ou Parecer Técnico)**: esta técnica envolve a consulta a um ou mais especialistas, possivelmente com a ajuda de um mecanismo de consenso entre eles;
- **Analogia**: esta técnica envolve a utilização dos custos reais de vários projetos completos para realizar a estimativa de custos de um novo projeto similar, ou seja, os estimadores comparam o projeto novo com um ou mais projetos passados;
- **Modelos Algorítmicos**: esta técnica é, na verdade, um conjunto de modelos que apresentam um ou mais algoritmos que produzem uma estimativa de custo de software como uma função de um número de variáveis que são consideradas ser de maior importância para o custo. Fenton e Pfleeger (1997) afirmam que estes modelos apresentam fórmulas matemáticas que relacionam, basicamente, itens como o tamanho, o tempo de desenvolvimento e o esforço. Alguns modelos incluem outras variáveis como a experiência dos desenvolvedores, a linguagem de implementação, o grau de reutilização, e assim por diante.
- **Parkinson**: um princípio de Parkinson (“O trabalho se expande para preencher o volume disponível”) é usado para comparar a estimativa de custo aos recursos disponíveis;
- ***Price-to-win***: esta técnica considera o preço que se acredita ser necessário para ganhar o trabalho (ou o prazo necessário para chegar primeiro no mercado com um produto novo, por exemplo);
- ***Top-down***: uma estimativa de custo global para o projeto é derivada de propriedades globais do produto de software. O custo total é então dividido entre os vários componentes;
- ***Bottom-up***: cada componente do software tem seu custo estimado separadamente e o resultado é agregado para produzir uma estimativa para o trabalho global.

Segundo Boehm (1981), é difícil determinar qual é a melhor técnica, porque suas vantagens e desvantagens são complementares. O ideal é usar uma combinação de técnicas aproveitando as vantagens em comum. Shepperd, Schofield e Kitchenham (1996), afirmam que é amplamente aceita a idéia de que uma estimativa de esforço de software exija o emprego de mais de uma destas técnicas.

O importante não é só executar estimativas independentes, mas também investigar por que elas produzem resultados diferentes. A necessidade de se investigar as diferenças entre estimativas é a principal razão pela qual é tão importante para um modelo de custo algorítmico ser "construtivo". Uma estimativa construtiva facilita as revisões porque o modelo mostra de onde tirou os resultados. Caso contrário, não há nenhum modo de comparar modelos algorítmicos de estimativa de custo com outros tipos de estimativas (BOEHM, 1981).

Neste trabalho, serão detalhadas apenas as técnicas de Julgamento Especialista (seção 2.7), Analogia (seção 2.8) e Modelos Algorítmicos (seção 2.9), por se entender que as técnicas de Parkinson, *Price-to-win*, *Top-down* e *Bottom-up* são estratégias administrativas e não técnicas efetivas de estimativa de custo de software. Boehm (1981), por exemplo, afirma que as técnicas Parkinson e *Price-to-win* não produzem estimativas de custo aceitáveis e, conforme Fenton e Pfleeger (1997), a maioria dos engenheiros de software usa apenas as técnicas de Julgamento Especialista, Analogia e Modelos Algorítmicos.

2.7. JULGAMENTO ESPECIALISTA OU PARECER TÉCNICO

Shepperd, Schofield e Kitchenham (1996) mostram que a maioria das pesquisas na área de estimativa de custo de software está concentrada nos Modelos Algorítmicos, porém o Julgamento Especialista é a técnica mais utilizada na prática. Bournemouth (1997) cita estudos desenvolvidos em 1992[1] e 1994[2] que confirmam a popularidade dessa técnica.

---

[1] HEMSTRA, F. J. Software cost estimation. **Information & Softw. Technol.**, 34 (10), p.627-639, 1992.

[2] VIGDER, M.R.; KARK, A.W. **Software cost estimation and control**. Report available from de link. February/1994.

Em sua forma mais simples, o julgamento especialista consiste na consulta a um ou mais especialistas locais informados sobre o ambiente de desenvolvimento e o domínio da aplicação com o objetivo de estimar o esforço requerido para concluir determinado projeto de software (BOURNEMOUTH, 1997).

Essa técnica pode considerar as diferenças entre experiências de projeto passadas e as novas técnicas, arquiteturas, ou aplicações envolvidas no projeto proposto. O especialista também pode considerar características excepcionais de pessoal e de interações, ou outras características que sejam únicas para o projeto (BOEHM, 1981).

Fenton e Pfleeger (1997) dizem que o parecer técnico tira proveito da experiência pessoal de um desenvolvedor maduro. Para usar esta técnica, o desenvolvedor ou o gerente descreve os parâmetros do projeto a ser empreendido, e os peritos fazem predições baseados em sua experiência passada. Na realidade, os peritos podem usar ferramentas, modelos, ou outros métodos para gerar as estimativas, mas estas técnicas de apoio não são visíveis ao requisitante. O requisitante confia na qualidade dos peritos e na amplitude da experiência deles.

As técnicas mais utilizadas para fazer o julgamento especialista são: *Delphi* e *Wideband-Delphi*.

2.7.1. Delphi

Tanto Bournemouth (1997), quanto Trindade (2000) apresentam a técnica *Delphi* como uma boa técnica utilizada pelo julgamento especialista.

Para Trindade (2000) esta técnica orçamentária se resume à consulta a especialistas de determinada área, em determinada linguagem e/ou em determinado assunto para que façam suas estimativas, utilizando-se de suas experiências e entendimento do projeto proposto.

Wiegers (2000) afirma que esta técnica foi desenvolvida pela *Rand Corporation* em 1948. Trindade (2000), porém, afirma que essa técnica foi criada na década de 60 e é composta das seguintes atividades:

- o coordenador apresenta as especificações do projeto aos especialistas participantes;

- os especialistas preenchem, de forma anônima, um formulário previamente distribuído. Nesta fase, os especialistas podem fazer perguntas ao coordenador e sua equipe, mas não podem trocar opiniões, nem discutir a situação entre si;
- o coordenador e sua equipe compilam os dados e preparam um resumo das respostas e solicitam nova interação, apresentando as razões das estimativas, por intermédio do formulário representado pela Figura 2;
- os especialistas preenchem de forma anônima e sem discussões o formulário para interações. Esta etapa pode ser repetida até que um determinado critério de parada seja atingido, por exemplo, um determinado desvio-padrão seja obtido. Durante todo o processo não há discussões em grupo em momento algum.

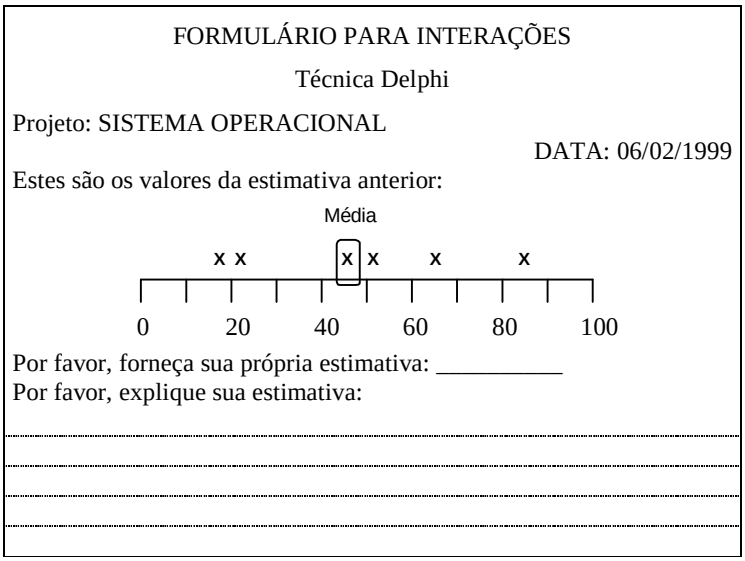

Figura 2. Formulário para interações na técnica Delphi
Fonte: Boehm (1981) e Trindade (2000)

2.7.2. Wideband-delphi

Segundo Trindade (2000), a *Wideband-Delphi* é uma variação da técnica *Delphi*. Wiegers (2000) afirma que no início da década de 70 Barry Boehm e um grupo da *Rand Corporation* modificaram a técnica *Delphi* chamando-a de *Wideband Delphi*. O objetivo desta técnica é produzir uma estimativa precisa e imparcial. Trata-se de uma técnica estruturada de Julgamento Especialista e é essencialmente baseada em procedimentos com múltiplos passos (BOURNEMOUTH, 1997).

Nessa versão, acontecem discussões no decorrer do processo que é composto, conforme Trindade (2000), Bournemouth (1997), Boehm (1981) e Fenton e Pfleeger (1997) das seguintes atividades:

- o coordenador apresenta aos especialistas participantes as especificações do projeto proposto, como também qualquer outra informação pertinente (como dados em projetos semelhantes que foram completados no passado);
- o coordenador convoca os especialistas para uma reunião de discussão do projeto;
- cada especialista anota sua estimativa inicial num formulário previamente distribuído, de forma individual e anônima;
- o coordenador e sua equipe compilam os dados e preparam um resumo das respostas e solicitam nova interação, sem solicitar as razões das estimativas;
- o coordenador convoca os especialistas para uma reunião de discussão sobre os pontos onde as estimativas variam de forma considerável;
- cada especialista, de forma individual e anônima, anota sua nova estimativa num formulário previamente distribuído para interações. Esta etapa, assim como as duas anteriores, acontece quantas vezes for necessário, por exemplo, até que um determinado critério de parada seja atingido.

Putnam e Fitzsimmons apud Fenton e Pfleeger (1997)[3] sugerem que a estimativa do grupo não deve ser a média simples das estimativas individuais, porque os valores muito altos ou os valores muito baixos podem prejudicar a estimativa final. Assim, recomenda-se a utilização da média ponderada das estimativas individuais que pode ser obtida por meio da equação (1):

$$\text{Estimativa} = \frac{\text{MenorEstimativa} + 4\,(\text{EstimativaProvável}) + \text{MaiorEstimativa}}{6} \qquad (1)$$

2.8. ANALOGIA

Segundo Kadoda et al. (2000), a idéia de usar a analogia como uma base para estimar o esforço de um determinado projeto de software não é nova. Boehm (1981) já dizia que esta

[3] PUTNAM, L. H.; FITZSIMMONS, A. Estimating software costs. **Datamation**, p. 312-315, September, October, November, 1979,

técnica envolve raciocínio por analogia utilizando-se um ou mais projetos completos. O objetivo é relacionar os custos dos projetos completos com uma estimativa do custo de um projeto novo que seja semelhante.

Conforme Schofield e Shepperd (1996), algumas pesquisas mostram que a estimação pela Analogia é uma técnica que pode render resultados úteis e pode ser, em muitos casos, mais segura e robusta que a aproximação tradicional por meio de Modelos Algorítmicos, mesmo usando-se somente os dados disponíveis na fase de especificação de um projeto.

Fenton e Pfleeger (1997) afirmam que a analogia é uma aproximação formal, mais visível que o parecer técnico (julgamento especialista), porque os estimadores comparam o projeto proposto com um ou mais projetos passados e tentam identificar semelhanças e diferenças. As diferenças entre os projetos passados e o projeto atual serão usadas como base, por exemplo, para ajustar a estimativa do valor do esforço.

Segundo Kadoda et al. (2000), deve-se ter **n** projetos ou casos, cada um dos quais precisam ser caracterizados em termos de um conjunto de características **p**. Ter-se-á que saber qual a característica que será predita. As características podem ser:

- contínuas (por exemplo, a experiência do gerente de projeto);
- discretas (por exemplo, o número de interfaces);
- categóricas (por exemplo, o ambiente de desenvolvimento).

A estimativa por analogia envolve a caracterização de projetos de software com informações disponíveis no momento em que a estimação é requerida. Por exemplo, um projeto poderia ser caracterizado pelo número de entradas e saídas, o número de telas e a linguagem de programação usada na codificação. A escolha de variáveis dependerá do tipo de aplicação, do ambiente de desenvolvimento e da base pragmática do que está disponível. Uma vez que um banco de dados de projetos completos foi construído, é possível estimar um novo projeto procurando projetos semelhantes ou análogos (SCHOFIELD e SHEPPERD, 1996). Conforme Kadoda et al. (2000), depois que o novo projeto (caso) foi completado ele pode ser acrescentado à base de casos.

Os valores devem ser unificados de forma que cada variável contribua com um peso igual para o processo de encontrar as analogias. Os projetos devem ser plotados em um espaço **n**-

dimensional (onde há uma dimensão para cada variável) e a distância Euclidiana deve ser medida. Os projetos muito semelhantes estarão mais próximos no espaço **n**-dimensional (SCHOFIELD e SHEPPERD, 1996).

Se muitos projetos completos forem disponibilizados, a estimativa preliminar pode ser melhorada selecionando-se dois projetos semelhantes: um maior e um menor que o projeto novo. Pode-se, então, interpolar entre os dois para obter uma estimativa inicial de esforço e prazo de duração para o projeto novo (FENTON e PFLEEGER, 1997).

Kadoda et al. (2000) observam que este raciocínio baseado em casos apresenta alguns aspectos distintos:

- caracterização dos casos;
- armazenamento de casos passados;
- recuperação de casos semelhantes para usá-los como analogias;
- utilização de casos semelhantes para resolver o problema do novo projeto (adaptação do caso).

Conforme Shepperd, Schofield e Kitchenham (1996), em alguns aspectos esta técnica é, freqüentemente, uma forma sistemática de julgamento especialista (desde que os peritos procurem por situações análogas para formar sua opinião). A técnica envolve a caracterização do projeto para o qual uma estimativa seja requerida, formando uma base para se procurar projetos semelhantes ou análogos que tenham sido completados e para os quais o esforço seja conhecido. Estes valores de esforço são então usados, possivelmente com ajuste, para gerar o valor estimado.

Shepperd, Schofield e Kitchenham (1996) afirmam que, em geral, o ideal é ter mais variáveis, porém, na prática o estimador é induzido a usar os dados que estão disponíveis.

Conforme Schofield e Shepperd (1996), originalmente, as estimativas por Analogia foram criadas por meio de um trabalho intensivo de busca manual. Ficou óbvio, porém, que o processo precisaria ser automatizado se fosse usado seriamente. Esses autores apresentaram as principais fases para montagem de uma estimativa por intermédio de um programa de Analogia:

- **identificação dos dados que serão colecionados**: colecionar dados que serão muito dependentes da natureza dos projetos para os quais são requeridas estimativas. Identificar variáveis que podem ser facilmente e confiavelmente coletadas na prática;
- **concordância entre as definições de dados e os mecanismos de coleção**: até mesmo dentro de uma única organização pode não haver nenhum entendimento compartilhado do que é significativo para o esforço. Qualquer programa de estimativa será fatalmente falho se projetos diferentes estiverem medindo os mesmos atributos de modos diferentes. É importante identificar quem é o responsável pela coleta de dados e quando eles devem ser coletados. É conveniente que a mesma pessoa colete os dados para todos os projetos a fim de aumentar o nível de consistência;
- **alimentação do banco de dados**: como todas as técnicas de estimativa, a analogia requer alguma coleta de dados. Na maioria dos casos, a coleta de dados será um processo em andamento, porque à medida que os projetos são completados seus dados de esforço ficam disponíveis;
- **sintonização do método de estimação**: o usuário precisará experimentar, a fim de identificar qual o número ótimo de analogias a serem pesquisadas e se deve usar um subconjunto de variáveis. Isto pode fazer diferença na qualidade das estimativas;
- **estimativa de novos projetos**: deve ser possível caracterizar o projeto em termos das variáveis que foram identificadas na primeira fase do processo de estimativa. Para estas variáveis, um programa pode ser usado para encontrar projetos semelhantes e o usuário pode fazer um julgamento subjetivo sobre o valor das analogias. Onde acredita-se que esses projetos sejam confiáveis, a predição pode ser usada para uma extensão maior do que onde esses projetos pareçam duvidosos.

2.9. MODELOS ALGORÍTMICOS

Shepperd, Schofield e Kitchenham (1996) dizem que, até o momento, a maioria das pesquisas foi focalizada em modelos de custo algorítmicos como COCOMO-*COnstructive COst MOdel* (Modelo de Custo Construtivo) e Pontos de Função. A seguir, serão apresentados os principais Modelos Algorítmicos.

2.9.1. Métricas Orientadas ao Tamanho

2.9.1.1. LINHAS DE CÓDIGO (*LINES OF CODE* - LOC)

Segundo Trindade (2000), a utilização de LOC-Linhas de Código é a primeira e mais natural das formas de medir o tamanho de um sistema computacional.

Park (1992) afirma que essa medida tem aplicação direta no planejamento e estimativa de projetos de software, pois pode ser usada no cálculo da produtividade, na normalização de indicadores de qualidade e ainda na derivação de medidas de utilização de memória.

Em seu relatório técnico sobre o assunto, Park (1992) propõe o uso de *checklists* que apresentam métodos operacionais para construção e comunicação de definições claramente compreendidas dessa medida de software, pois permite satisfazer as necessidades de dados de usuários diferentes mantendo definições consistentes de tamanho.

Segundo Roetzheim (2000a), as linhas em branco e as linhas de comentários não devem ser contadas. A quantidade de linhas de código também é dependente da linguagem e do ambiente de programação. Apesar disso, as LOC foram amplamente usadas no passado, mas tendências atuais indicam que esta métrica está ficando menos estável rapidamente. Isso, provavelmente, deve-se às mudanças no processo de desenvolvimento de software, como por exemplo, a tendência em usar a prototipação e as ferramentas CASE.

Um modelo para a predição do esforço, considerando as LOC, pode levar à equação básica (2):

$$E = S \times fpl \qquad (2)$$

Onde:

E: é o esforço aplicado (em pessoas-mês).

S: é o tamanho do sistema em KLOC (milhares de linhas de código).

fpl: é o Fator de Produtividade Linear (1 / (KLOC/pessoas-mês)).

A equação (2) assume que a relação entre esforço e tamanho é linear. Roetzheim (2000a) apresenta alguns valores comuns encontrados para este Fator de Produtividade Linear. Observe-se que, embora a linguagem afete a produtividade em termos de funcionalidade por

hora, o esforço medido em linhas de código fonte é independente da linguagem. Os valores apresentados na Tabela 2 são provenientes do trabalho de Barry Boehm, Raymond Kyle e U.S. Air Force Cost Analysis Agency's, além de organizações que trabalham diretamente com o Cost Xpert Group (empresa de treinamento e consultoria na área de estimativa de custos de software fundada em 1992 por William Roetzheim que presta serviços para organizações como a Boeing, Sybase, Lotus, Unisys, Dell, NASA, entre outras (COST XPERT, 2002)).

Tabela 2. Valores comuns para o fator de produtividade linear

Tipo de Projeto	Fator de Produtividade Linear
Desenvolvimento padrão	2.94
Desenvolvimento *embedded*	2.58
Desenvolvimento de *E-commerce*	3.60
Desenvolvimento *web*	3.30
Desenvolvimento militar	2.77

Fonte: Roetzheim (2000a)

Roetzheim (2000a) afirma que a produtividade varia de acordo com o tamanho do projeto. Projetos grandes são significativamente menos produtivos que projetos pequenos, provavelmente porque requerem mais coordenação e tempo de comunicação, além de retrabalho para corrigir erros. A diminuição de produtividade com o aumento do tamanho do projeto é visível. A Tabela 3 mostra alguns fatores exponenciais de penalidade típicos para vários tipos de projeto:

Tabela 3. Fatores de penalidade para vários tipos de projetos

Tipo de Projeto	Fator de Penalidade Exponencial do Tamanho
Desenvolvimento padrão	1.052
Desenvolvimento *embedded*	1.110
Desenvolvimento de *E-commerce*	1.030
Desenvolvimento *web*	1.030
Desenvolvimento militar	1.072

Fonte: Roetzheim (2000a)

O custo de um projeto é ajustado com base nas ineficiências associadas ao tamanho crescente do projeto e o Fator de Penalidade Exponencial do tamanho age no valor total de LOC. A equação (2) é então modificada para comportar este fator, como mostrado na equação (3):

$$E = S^{fpe} \times fpl \qquad (3)$$

Onde:

E: é o esforço aplicado (em pessoas-mês).
S: é o tamanho do sistema (em KLOC).
fpl: é o Fator de Produtividade Linear (1 / (KLOC/pessoas-mês)).
fpe: é o Fator de Penalidade Exponencial do tamanho.

Roetzheim (2000b) afirma que os fatores ambientais de projeto podem ter uma influência maior sobre a eficiência da equipe de desenvolvimento (fatores ambientais não-lineares) e que estes fatores são escalonados em muito pequeno, pequeno, normal, grande, muito grande ou extra grande, porém apresenta apenas os valores mostrados na Tabela 4.

Tabela 4. Fatores ambientais não-lineares

Fatores Ambientais	Muito Pequeno	Normal	Extra Grande
Arquitetura / Resolução de risco	0.0423	0.0140	- 0.0284
Flexibilidade de desenvolvimento	0.0223	0.0020	- 0.0284
Precedência	0.0336	0.0088	- 0.0284
Maturidade do processo	0.0496	0.0814	- 0.0284
Entrosamento da equipe	0.0264	0.0045	- 0.0284

Fonte: Roetzheim (2000b)

Cada um desses fatores ambientais não-lineares (mostrados na Tabela 4) devem ser somados ao Fator de Penalidade Exponencial visto na Tabela 3 como apresentado na equação (4):

$$E = p \times S^{(fpe + \Sigma\ fanl)} \times fpl \qquad (4)$$

Onde:
E: é o esforço aplicado (em pessoas-mês).
S: é o tamanho do sistema (em KLOC).
fpl: é o Fator de Produtividade Linear (1 / (KLOC/pessoas-mês)).
fpe: é o Fator de Penalidade Exponencial do tamanho.
Σ fanl: é o somatório dos Fatores Ambientais Não-Lineares.

Segundo Roetzheim (2000b), há fatores lineares de ajuste ambiental, conforme pode ser visto na Tabela 5.

Tabela 5. Impacto dos fatores ambientais lineares

Fatores Ambientais	Muito Pequeno	Normal	Extra Grande
Capacidade do analista	1.42	1.00	0.71
Experiência nas aplicações	1.22	1.00	0.81
Experiência na linguagem de programação e ferramentas	1.20	1.00	0.84
Continuidade de pessoal	1.29	1.00	0.81
Capacidade do gerente	1.18	1.00	0.87
Experiência do gerente	1.11	1.00	0.90
Experiência na plataforma	1.19	1.00	0.85
Capacidade do programador	1.34	1.00	0.76
Restrições de tempo de execução	1.00	1.00	1.63
Restrições de armazenamento principal	1.00	1.00	1.46
Volatilidade da plataforma	0.87	1.00	1.30
Ferramentas efetivas de gerência	1.22	1.00	0.84
Desenvolvimento de *multisite*	1.22	1.00	0.80
Ergonomia de escritório	1.19	1.00	0.82
Uso de ferramentas de s/w	1.17	1.00	0.78
Tamanho do banco de dados	0.90	1.00	1.28
Correspondência da documentação com o ciclo de vida	0.81	1.00	1.23
Internacionalização	0.97	1.00	1.35
Complexidade do produto	0.75	1.00	1.66
Exigência da capacidade de reutilização	0.95	1.00	1.24
Exigência de confiança do software	0.82	1.00	1.26
Gráficos e multimídia	0.95	1.00	1.35
Integração com sistemas legados	1.00	1.00	1.18
Segurança do site	0.92	1.00	1.40
Conteúdo do texto	0.94	1.00	1.16
Seleção de ferramentas	0.95	1.00	1.14
Carga de transações	0.96	1.00	1.59
Estratégia de web	0.88	1.00	1.45

Fonte: Roetzheim (2000b)

Esses números foram determinados empiricamente pelos estudos de Barry Boehm, Capers Jones e William H. Roetzheim e foram validados por mais de 20.000 projetos (ROETZHEIM, 2000b). Esses valores devem ser multiplicados, inclusive com o resultado de (4), formando a equação (5):

$$E = S^{(fpe + \Sigma\ fanl)} \times fpl \times \Pi\ fal \qquad (5)$$

Onde:

E: é o esforço aplicado (em pessoas-mês).

S: é o tamanho do sistema (em KLOC).

fpl: é o Fator de Produtividade Linear (1 / (KLOC/pessoas-mês)).

fpe: é o Fator de Penalidade Exponencial do tamanho.

Σ fanl: é o somatório dos Fatores Ambientais Não-Lineares.

Π fal: é o produtório dos Fatores Ambientais Lineares.

Conforme Roetzheim (2000b), faz-se necessário converter o esforço calculado na equação (5) em um prazo ótimo. Embora pareça relativamente difícil selecionar e calcular os valores corretos para os parâmetros linear e não-linear, uma aproximação razoável é a raiz cúbica do esforço (esforço elevado a 0.33), e multiplicando-se o resultado por um valor da Tabela 6, conforme mostrado na equação (6).

$$T = E^{0.33} \times fce \quad (6)$$

Onde:

T: é o tempo de desenvolvimento (em meses).

E: é o esforço aplicado (em pessoas-mês).

fce: é o Fator de Conversão do Esforço.

Tabela 6. Fatores de conversão do esforço por tipo de projeto

Tipo de Projeto	Fator de Conversão do Esforço
Desenvolvimento padrão	3.67
Desenvolvimento *embedded*	4.00
Desenvolvimento de *E-commerce*	3.20
Desenvolvimento *web*	3.10
Desenvolvimento militar	3.80

Fonte: Roetzheim (2000b)

Conforme Bournemouth (1997), a quantidade de linhas de código fonte podem ser usadas como estimativa do tamanho do software, mas obviamente, este valor não é conhecido no início do projeto.

Roetzheim (2000a) aconselha que as variáveis de entrada sejam determinadas por Parecer Técnico dos próprios desenvolvedores ou com a participação dos mesmos (esta técnica foi abordada na seção 2.7 deste trabalho). O primeiro passo seria dividir o projeto em módulos pequenos o suficiente para que os desenvolvedores, baseados em suas experiências em módulos semelhantes, possam estimar o número de linhas de código fonte.

2.9.1.2. MODELO DE ESTIMATIVA DE PUTNAM

Trindade (2000) afirma que, em 1978, W. Putnam criou um modelo de estimativa para atender às forças armadas americanas que precisavam de um método para estimar esforço e tempo de entrega para grandes projetos.

Segundo Pressman (1995) e Trindade (2000), esse modelo busca medir o esforço e o tempo de desenvolvimento por meio da dinâmica de múltiplas variáveis que pressupõe uma distribuição de esforço específica ao longo da existência de um projeto de desenvolvimento de software, utilizando conceitos desenvolvidos anteriormente por Rayleigh e por Norden. Os estudos de Rayleigh e Norden resultaram numa distribuição de esforço conhecida como Curva de Rayleigh-Norden.

Tal curva pode ser usada para derivar a equação (7) que relaciona linhas de código ao tempo e esforço de desenvolvimento (PRESSMAN, 1995; BOEHM, 1981 e TRINDADE, 2000).

$$S_{loc} = C \times E_{pa}^{1/3} \times t_d^{4/3} \quad (7)$$

onde:

S_{loc} é o tamanho do sistema (em LOC).

C é uma constante do estado de tecnologia e reflete as restrições que impedem o progresso do programador.

E_{pa} é o esforço empregado (em pessoas-ano) para desenvolvimento e manutenção do software.

t_d é o tempo de desenvolvimento (em anos).

Pressman (1995) recomenda que a constante **C** seja derivada para condições locais usando-se dados históricos devidamente compilados, ou que sejam usados os seguintes valores:

- 2.000 para um ambiente de desenvolvimento de software simples (sem metodologias, nem documentação, e com revisões precárias);
- 8.000 para um ambiente de desenvolvimento de software bom (com aplicação de metodologia, documentações, revisões adequadas e desenvolvimento interativo);
- 11.000 para um ambiente de desenvolvimento de software excelente (uso de ferramentas e técnicas automatizadas).

Ao se efetuar um novo arranjo da equação (7), pode-se chegar a uma equação (8) para o esforço de desenvolvimento (PRESSMAN, 1995 e TRINDADE, 2000):

$$E_{pa} = \frac{S_{loc}^{3}}{C^3 \times t_d^4} \quad (8)$$

Conforme Roetzheim (2000b), Norden descobriu que um projeto de pessoal ótimo para projetos típicos que requerem comunicação e aprendizado segue uma distribuição de Rayleigh. Esta curva (Figura 3) apresenta um declive relativamente acentuado, depois diminui suavemente, apresentando a forma de uma onda. Putnam, em suas pesquisas, confirmou estes resultados aplicados a desenvolvimento de software.

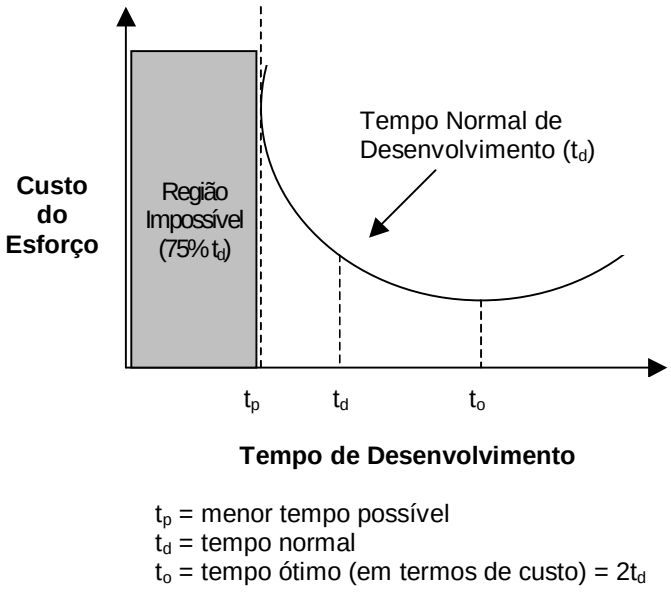

Figura 3. Reduzindo custos com o aumento do tempo de desenvolvimento
Fonte: Roetzheim (2000b) e Peters (2000)

A Figura 3 mostra que é possível reduzir custos aumentando o prazo de desenvolvimento para t_o, definido como o prazo de desenvolvimento ótimo em termos de custo. Este tempo (t_o) foi determinado empiricamente como sendo o dobro de t_d (tempo normal de desenvolvimento) e o resultado é uma redução no custo de 50% (cinqüenta por cento). Como o prazo foi aumentado, os custos começam a subir novamente a uma taxa aproximadamente linear (ROETZHEIM, 2000b).

Embora seja incomum um tempo de projeto ser empurrado para além de t_d, é comum o software ser requerido antes do prazo permitido por t_d. Apressar o tempo de desenvolvimento é possível, mas o custo é alto. Como mostrado na Figura 3, o custo aumenta exponencialmente quando o prazo do projeto é acelerado (ROETZHEIM, 2000b).

Estudos comprovam que nenhum projeto é entregue com sucesso num prazo inferior a 0.75 t_d. Esta barreira é conhecida como região impossível, porque, com abordagens de desenvolvimento tradicionais, é impossível concluir o projeto num prazo menor que 0.75 t_d (ROETZHEIM, 2000b).

Peters (2000) afirma que o menor tempo possível pode não ser realizável por boa parte das organizações. Para alcançar o menor tempo possível, a equipe de projeto precisa ser altamente qualificada e experiente, o processo de desenvolvimento deve ter sido muito bem definido e amadurecido, e o projeto tem que estar perfeito. Não há muitas organizações que podem fazer o menor tempo possível, assim é mais conveniente não esperar por isto. O melhor a fazer é determinar o menor tempo realizável pela organização (geralmente o tempo normal). Neste caso, os dados de projetos passados são a melhor fonte de informação.

Roetzheim (2000b) afirma que, se houver uma exigência de entrega num prazo localizado na região impossível, quatro estratégias básicas podem ser consideradas: reduzir a funcionalidade, dividir as tarefas, usar desenvolvimento paralelo redundante e aumentar a reutilização. Reduzir a funcionalidade é a melhor delas.

Em Trindade (2000), Putnam afirma que tanto o esforço quanto o prazo são variáveis possíveis de serem medidas, mas um desses parâmetros precisa ser estimado para que o outro possa ser calculado (geralmente o prazo, considerando-se pressões ambientais e contratuais).

2.9.1.3. MODELO DE CUSTO CONSTRUTIVO (*CONSTRUCTIVE COST MODEL* - COCOMO)

Conforme Pressman (1995), Roetzheim (2000a) e Azevedo (1999), Barry Boehm apresentou uma hierarquia de modelos de estimativa de software chamada COCOMO (*COnstrutive COst MOdel*), sendo:

- **COCOMO Básico (*Basic*)**: computa o esforço e o custo de desenvolvimento considerando uma estimativa do tamanho do programa (em LOC).
- **COCOMO Intermediário (*Intermediate*)**: computa o esforço e o custo de desenvolvimento considerando uma estimativa do tamanho do programa e um conjunto de direcionadores de custo (avaliações subjetivas do produto, do hardware, do pessoal e dos atributos do projeto).

- **COCOMO Detalhado (*Detailed*)**: além das características do COCOMO intermediário, inclui uma avaliação do impacto dos direcionadores de custo sobre cada etapa do desenvolvimento.

O modelo COCOMO pode ser aplicado em três classes de projetos:

- **Modo Orgânico**: projetos simples, relativamente pequenos, com conjuntos de requisitos não tão rígidos, com equipes pequenas e experientes.
- **Modo Semidestacado**: projetos intermediários (em tamanho e complexidade), com alguns requisitos rígidos e outros não tão rígidos, com níveis mistos de experiência nas equipes.
- **Modo Embutido**: projetos com conjunto rígido de restrições operacionais, tanto de hardware, quanto de software.

a) Modelo COCOMO Básico

Boehm (1981) apresenta as equações (9) e (10) como as duas equações básicas deste modelo:

$$E = a \times S^b \quad (9)$$

$$T = c \times E^d \quad (10)$$

onde:

E: é o esforço aplicado (em pessoas-mês).
T: é o tempo de desenvolvimento (em meses cronológicos).
S: é o número estimado de linhas de código em milhares (KLOC).
a: é um coeficiente fornecido pela Tabela 7.
b: é um expoente fornecido pela Tabela 7.
c: é um coeficiente fornecido pela Tabela 7.
d: é um expoente fornecido pela Tabela 7.

Segundo Boehm (1981), os valores dos coeficientes “a” e “c” e dos expoentes “b” e “d” do modelo COCOMO Básico são apresentados na Tabela 7.

Tabela 7. COCOMO Básico

Projeto de Software	a	b	c	d
Orgânico	2,40	1,05	2,50	0,38
Semidestacado	3,00	1,12	2,50	0,35
Embutido	3,60	1,20	2,50	0,32

Fonte: Boehm (1981)

b) Modelo COCOMO Intermediário

Neste modelo, o COCOMO Básico é ampliado com a finalidade de levar em consideração um conjunto de atributos direcionadores do custo que são agrupados em quatro categorias:

- Atributos do produto: - confiabilidade exigida do software;
  - tamanho do banco de dados;
  - complexidade do produto.
- Atributos do hardware: - restrições ao tempo de execução;
  - restrições de memória;
  - volatilidade do ambiente de máquina virtual;
  - tempo de *turnaround* (tempo para completar o ciclo) exigido.
- Atributos de pessoal: - capacidade do analista;
  - experiência em aplicações;
  - capacidade do programador;
  - experiência em máquina virtual;
  - experiência com a linguagem de programação.
- Atributos de projeto: - uso de práticas modernas de programação;
  - uso de ferramentas de software;
  - cronograma exigido de desenvolvimento.

Cada um desses atributos deve ser classificado de acordo com uma escala que varia de “muito baixo” a “extremamente elevado” (em importância e valor). Esta escala é apresentada na Tabela 8. A partir desta classificação determina-se o Multiplicador de Esforço (considerando a Tabela 8 publicada por Boehm (1981)). O produto de todos os resultados de Multiplicadores de Esforços é chamado de Fator de Ajustamento de Esforço (PRESSMAN, 1995).

Tabela 8. Multiplicadores de Esforço de Desenvolvimento de Software

Direcionadores de Custo	Muito Baixo	Baixo	Normal	Elevado	Muito Elevado	Extrema-mente Elevado
ATRIBUTOS DO PRODUTO						
Confiabilidade exigida do software	0,75	0,88	1,00	1,15	1,40	-
Tamanho do banco de dados	-	0,94	1,00	1,08	1,16	-
Complexidade do produto	0,70	0,85	1,00	1,15	1,30	1,65
ATRIBUTOS DO HARDWARE						
Restrições ao tempo de execução	-	-	1,00	1,11	1,30	1,66
Restrições de memória	-	-	1,00	1,06	1,21	1,56
Volatilidade do ambiente de máquina virtual	-	0,87	1,00	1,15	1,30	-
Tempo de *turnaround* (tempo para completar o ciclo) exigido	-	0,87	1,00	1,07	1,15	-
ATRIBUTOS DE PESSOAL						
Capacidade do analista	1,46	1,19	1,00	0,86	0,71	-
Experiência em aplicações	1,29	1,13	1,00	0,91	0,82	-
Capacidade do programador	1,42	1,17	1,00	0,86	0,70	-
Experiência em máquina virtual	1,21	1,10	1,00	0,90	-	-
Experiência com a linguagem de programação	1,14	1,07	1,00	0,95	-	-
ATRIBUTOS DE PROJETO						
Uso de práticas modernas de programação	1,24	1,10	1,00	0,91	0,82	-
Uso de ferramentas de software	1,24	1,10	1,00	0,91	0,83	-
Cronograma exigido de desenvolvimento	1,23	1,08	1,00	1,04	1,10	-

Fonte: Boehm (1981)

O modelo COCOMO Intermediário usa a equação (11) para a estimativa do esforço e a equação (10) para a estimativa do tempo de desenvolvimento:

$$E = a \times S^b \times fae \qquad (11)$$

onde:

E: é o esforço aplicado (em pessoas-mês).

S: é o número (estimado) de linhas de código para o projeto (em milhares).

a: é um coeficiente fornecido pela Tabela 9.

b: é um expoente fornecido pela Tabela 9.

fae: é o Fator de Ajustamento do Esforço (multiplicação de cada um dos Multiplicadores de Esforço fornecidos pela Tabela 8).

Para Boehm (1981) os valores do coeficiente “a” e do expoente “b” do modelo COCOMO Intermediário são apresentados na Tabela 9.

Tabela 9. COCOMO Intermediário

Projeto de Software	a	b
Orgânico	3,20	1,05
Semidestacado	3,00	1,12
Embutido	2,80	1,20

Fonte: Boehm (1981)

c) Modelo COCOMO Detalhado

Boehm (1981) afirma que as duas principais características deste modelo estão relacionadas com as limitações do COCOMO Intermediário, ou seja, o COCOMO Detalhado soluciona essas limitações. As principais características deste modelo são:

- **Multiplicadores de esforço sensíveis à fase**: este modelo apresenta um conjunto de multiplicadores de esforço sensíveis à fase para cada atributo direcionador de custo, com o objetivo de determinar a quantidade de esforço exigida para completar cada fase.
- **Hierarquia de produto de três níveis**: este modelo evita o problema de ter que definir vários direcionadores de custo quando vários componentes se agrupam em um subsistema com praticamente as mesmas avaliações para esses direcionadores provendo uma hierarquia de produto em três níveis que são:
  - **Nível de Módulo**: são tratados neste nível os efeitos que tendem a variar com cada nível de módulo base;
  - **Nível de Subsistema**: são tratados neste nível os efeitos que freqüentemente pouco variam;
  - **Nível de Sistema**: são tratados neste nível os efeitos como o do tamanho total do produto.

Conforme Boehm (1981), para algumas necessidades como a estimativa do tempo total de desenvolvimento e da distribuição do esforço por atividade a técnica a ser utilizada é semelhante a dos modelos COCOMO Básico e Intermediário[4].

[4] Para um estudo mais aprofundado e detalhado deste modelo recomenda-se a leitura minuciosa dos capítulos 23, 24, 25, 26 e 27 de BOEHM (1981).

2.9.2. Métricas Orientadas à Função

2.9.2.1. PONTOS DE FUNÇÃO *(FUNCTION POINTS)*

Segundo Azevedo (1999) e Braga (1996), o modelo de Pontos de Função foi criado na década de 70 por Allan J. Albrecht.

Em outubro de 1979, Albrecht apresentou os primeiros resultados desta nova técnica de medição de software, chamando-a "*Function Points*", numa conferência em Monterey, na Califórnia, mostrando que a produtividade econômica do software podia ser medidae isso marcou a história da computação (JONES, 1997).

Em 1986 foi formado o *IFPUG-International Function Point Users Group* para ajudar na transmissão de dados e informações sobre essa métrica. Em 1987, o governo britânico adotou uma forma modificada de Pontos de Função como a métrica padrão para a produtividade de software. Em 1990, o IFPUG publicou a versão 3.0 do Manual Prático de Contagem de Pontos de Função que representou uma visão de consenso das regras de contagem dos pontos (JONES, 1997).

Segundo Jones (1997), Allan J. Albrecht defendeu a posição de que a unidade de produção econômica de projetos de software deveria ser válida para todas as linguagens de programação e deveria representar questões que dizem respeito aos usuários do software. Resumindo, seu objetivo era medir a funcionalidade do software.

Para Dekkers (1999), o modelo de Pontos de Função mede o tamanho do **que** o software faz, ao invés de medir **como** ele é (ou será) desenvolvido e implementado, ou seja, o tamanho funcional do software será o mesmo, não importa a linguagem de programação utilizada, nem a técnica de desenvolvimento (métodos estruturados, ou RAD-Desenvolvimento Rápido de Aplicações, por exemplo).

Albrecht considerou que os aspectos externos visíveis de software que poderiam ser enumerados com precisão consistiam de cinco itens: as entradas para a aplicação, as saídas, as consultas dos usuários, os arquivos de dados a serem atualizados pela aplicação e as interfaces para outras aplicações (JONES, 1997).

Para se determinar o número de Pontos de Função inicia-se pela estimativa do número de entradas externas, arquivos de interface externa, saídas externas, consultas externas e arquivos lógicos internos (ROETZHEIM, 2000a).

Hazan (2000) e Braga (1996) asseguram que a contagem de Pontos de Função brutos (não ajustados) leva em consideração dois tipos de funções:

- **Funções do Tipo Dado**: representam a funcionalidade fornecida ao usuário a fim de atender aos requisitos internos e externos à aplicação, referentes a dados. São consideradas funções do tipo dado: arquivo lógico interno e arquivo de interface externa.

- **Funções do Tipo Transação**: representam a funcionalidade fornecida ao usuário de processamento dos dados pela aplicação. São consideradas funções do tipo transação: entrada externa, saída externa e consulta externa.

Cada uma das funções do tipo dado e do tipo transação serão definidas e identificadas a seguir:

a) Arquivo Lógico Interno-ALI (Internal Logical File-ILF)

Um arquivo lógico interno é um grupo de dados logicamente relacionados ou informações de controle. É identificado pelo usuário e recebe manutenção dentro das fronteiras da aplicação que está sendo contada.

A complexidade de um arquivo lógico interno é calculada a partir da quantidade de registros lógicos referenciados (um subgrupo de elementos de dados, reconhecido pelo usuário, dentro de um arquivo lógico interno) e da quantidade de dados elementares referenciados (um campo, reconhecido pelo usuário, que esteja presente em um arquivo lógico interno), usando-se ainda as regras de contagem e a definição de complexidade citadas por Braga (1996), conforme pode ser visto na Tabela 10:

Tabela 10. Identificação da complexidade para arquivos lógicos internos

Registros Lógicos Referenciados	Dados Elementares Referenciados		
	01 até 19	20 até 50	51 ou mais
01	Simples	Simples	Média
02 até 05	Simples	Média	Complexa
06 ou mais	Média	Complexa	Complexa

Fonte: Braga (1996) e Azevedo (1999)

b) Arquivo de Interface Externa-AIE (External Interface File-EIF)

Um arquivo de interface externa é um grupo de dados logicamente relacionados ou informações de controle utilizados no sistema que está sendo analisado, mas que não recebem manutenção dentro das fronteiras da aplicação que está sendo contada. Isso significa que um AIE contado para uma aplicação deve ser um ALI de uma outra aplicação.

A complexidade de um arquivo de interface externa é calculada a partir da quantidade de registros lógicos referenciados (um subgrupo de elementos de dados, reconhecido pelo usuário, dentro de um arquivo de interface externa) e da quantidade de dados elementares referenciados (um campo, reconhecido pelo usuário, que esteja presente em um arquivo de interface externa), usando-se ainda as regras de contagem e a definição de complexidade citadas por Braga (1996), conforme pode ser visto na Tabela 11:

Tabela 11. Identificação da complexidade para arquivos de interface externa

Registros Lógicos Referenciados	Dados Elementares Referenciados		
	01 até 19	20 até 50	51 ou mais
01	Simples	Simples	Média
02 até 05	Simples	Média	Complexa
06 ou mais	Média	Complexa	Complexa

Fonte: Braga (1996) e Azevedo (1999)

c) Entrada Externa-EE (External Input-EI)

Uma entrada externa é um processo elementar que processa dados ou informações de controle que entram pela fronteira da aplicação (geradas por fonte externa à aplicação que está sendo contada). O objetivo principal é manter um ou mais ALI e/ou alterar o comportamento do sistema.

A complexidade de uma entrada externa é calculada a partir da quantidade de arquivos lógicos referenciados (um arquivo lógico interno ou um arquivo de interface externa lidos ou mantidos por um tipo de função) e da quantidade de dados elementares referenciados (um único campo não recursivo, identificado pelo usuário, mantido em um arquivo lógico interno pela entrada externa), usando-se ainda as regras de contagem e a definição de complexidade citadas por Braga (1996), conforme pode ser visto na Tabela 12:

Tabela 12. Identificação da complexidade para entrada externa

Arquivos Lógicos Referenciados	Dados Elementares Referenciados		
	01 até 04	05 até 15	16 ou mais
00 ou 01	Simples	Simples	Média
02	Simples	Média	Complexa
03 ou mais	Média	Complexa	Complexa

Fonte: Braga (1996) e Azevedo (1999)

d) Saída Externa-SE (External Output-EO)

Uma saída externa é um processo elementar que envia dados ou informações de controle para fora da fronteira da aplicação. O objetivo principal é apresentar informação para um usuário por meio de um processamento lógico e não apenas por uma simples recuperação de dados. O processamento lógico deve conter, no mínimo, uma fórmula matemática, ou cálculo, ou criar dados derivados. Pode também manter um ou mais ALI e/ou alterar o comportamento do sistema.

A complexidade de uma saída externa é calculada a partir da quantidade de arquivos lógicos referenciados (um arquivo lido pela lógica de processamento da saída externa) e da quantidade de dados elementares referenciados (um único campo não recursivo, identificado pelo usuário, que aparece em uma saída externa), usando-se ainda as regras de contagem e a definição de complexidade citadas por Braga (1996), conforme pode ser visto na Tabela 13.

Tabela 13. Identificação da complexidade para saída externa

Arquivos Lógicos Referenciados	Dados Elementares Referenciados		
	01 até 05	06 até 19	20 ou mais
00 ou 01	Simples	Simples	Média
02 ou 03	Simples	Média	Complexa
04 ou mais	Média	Complexa	Complexa

Fonte: Braga (1996) e Azevedo (1999)

e) Consulta Externa-CE (External Inquiry-EQ)

Uma consulta externa é um processo elementar que envia dados ou informações de controle para fora da fronteira da aplicação. O objetivo principal é apresentar informação para o usuário por intermédio da recuperação de dados. O processamento lógico **não** contém fórmulas matemáticas, **nem** cálculos e **não** cria dados derivados. **Nenhum** ALI é mantido durante o processamento e o comportamento do sistema **não** é alterado.

A complexidade de uma consulta externa é calculada a partir da quantidade de arquivos lógicos referenciados (um arquivo lido quando a consulta externa é processada) e da quantidade de dados elementares referenciados (um campo não recursivo, identificado pelo usuário, que aparece em uma consulta externa).

Braga (1996), apresenta ainda regras de contagem e definição de complexidade distintas para a parte de entrada (Tabela 14) e para a parte de saída (Tabela 15) da consulta externa. Deve-se utilizar apenas a maior complexidade encontrada entre as partes de entrada e de saída.

Tabela 14. Identificação da complexidade para consulta externa - parte de entrada

Arquivos Lógicos Referenciados	Dados Elementares Referenciados		
	01 até 04	05 até 15	16 ou mais
00 ou 01	Simples	Simples	Média
02	Simples	Média	Complexa
03 ou mais	Média	Complexa	Complexa

Fonte: Braga (1996) e Azevedo (1999)

Tabela 15. Identificação da complexidade para consulta externa - parte de saída

Arquivos Lógicos Referenciados	Dados Elementares Referenciados		
	01 até 05	06 até 19	20 ou mais
00 ou 01	Simples	Simples	Média
02 ou 03	Simples	Média	Complexa
04 ou mais	Média	Complexa	Complexa

Fonte: Braga (1996) e Azevedo (1999)

f) Cálculo dos Pontos de Função Brutos

Após a definição de cada função e sua complexidade relativa, deve-se calcular os Pontos de Função brutos (não ajustados). Braga (1996), sugere o quadro apresentado na Tabela 16 para o cálculo dos Pontos de Função brutos:

Tabela 16. Dados para cálculo dos pontos de função brutos

Tipo da Função	Complexidade Funcional	Peso da Complexidade	Qtde	Subtotal de Pontos
Arquivo Lógico Interno	Simples	7 x		
	Média	10 x		
	Complexa	15 x		
Arquivo de Interface Externa	Simples	5 x		
	Média	7 x		
	Complexa	10 x		
Entrada Externa	Simples	3 x		
	Média	4 x		
	Complexa	6 x		
Saída Externa	Simples	4 x		
	Média	5 x		
	Complexa	7 x		
Consulta Externa	Simples	3 x		
	Média	4 x		
	Complexa	6 x		
Total de Pontos de Função Brutos				

Fonte: Braga (1996) e Azevedo (1999)

g) Cálculo do Fator de Ajuste

Os seguintes passos devem ser executados para o cálculo do fator de ajuste:

- Atribuir um nível de influência (Tabela 17) que pode variar de 0 (nenhuma influência) até 5 (influência forte) para cada uma das catorze características gerais do sistema (Tabela 18) na aplicação que está sendo contada.

Tabela 17. Níveis de influência

Grau	Descrição
0	Nenhuma influência
1	Influência mínima
2	Influência moderada
3	Influência média
4	Influência significante
5	Influência forte

Fonte: Braga (1996) e Azevedo (1999)

Tabela 18. Características gerais do sistema

Características Gerais do Sistema		Nível de Influência
1.	Comunicação de dados	
2.	Funções distribuídas	
3.	Performance	
4.	Configuração do equipamento	
5.	Volume de transações	
6.	Entrada de dados on-line	
7.	Interface com o usuário	
8.	Atualização on-line	
9.	Processamento complexo	
10.	Reusabilidade	
11.	Facilidade de implantação	
12.	Facilidade operacional	
13.	Múltiplos locais	
14.	Facilidade de mudanças (flexibilidade)	
Total do Nível de Influência (NI)		

Fonte: Braga (1996) e Azevedo (1999)

- Calcular NI (total do nível de influência) através da soma da influência de cada uma das 14 características.
- Calcular o FA (Fator de Ajuste) por meio da equação (12):

$$FA = (NI \times 0{,}01) + 0{,}65 \quad (12)$$

- Calcular os PFA (Pontos de Função Ajustados), usando o FA (Fator de Ajuste) e os PFB (Pontos de Função Brutos) pela equação (13):

$$PFA = PFB \times FA \quad (13)$$

O fator de ajuste é aplicado sobre os Pontos de Função Brutos para permitir o cálculo dos Pontos de Função Ajustados. Esse valor pode variar de 0,65 até 1,35.

Roetzheim (2000a), Calvert (1996) e Jones (apud TRINDADE, 2000) sugerem uma tabela de linhas de código fonte por ponto de função para as linguagens de programação mais comuns atualmente, com o objetivo de fazer estimativas de quantidade de LOC para um determinado projeto a partir de sua quantidade de pontos de função. Parte dessa tabela foi reproduzida na Tabela 19:

Tabela 19. Linhas de código fonte por ponto de função para algumas linguagens de programação

Linguagem de Programação	LOC por Ponto de Função
C++ padrão	53
Cobol padrão	107
Delphi 5	18
HTML 4	14
Visual Basic 6	24
SQL padrão	13
Java 2 padrão	46
4GLs	20
Geradores de código	16

Fonte: Roetzheim (2000a), Calvert (1996) e Jones (apud TRINDADE, 2000)

Conforme Longstreet (2000), os Pontos de Função podem ser utilizados com várias finalidades:

- estimar o custo, o cronograma e o esforço de projeto;
- calcular o custo real do software;
- desenvolver um conjunto padrão de métricas;
- ajudar a entender faixas de produtividade amplas;
- entender o aumento do escopo;
- ajudar em negociações contratuais;
- estimar casos de teste;
- definir quando e onde fazer reengenharia;
- ajudar a entender os custos de manutenção.

2.9.2.2. PONTOS DE PARTICULARIDADE *(FEATURE POINTS)*

Jones (1997) declara que os Pontos de Função tradicionais não se aplicam bem a alguns tipos de software que possuem alto grau de complexidade algorítmica, pois como possuem poucas entradas e saídas geram uma pequena quantidade de Pontos de Função. Esses tipos de software requerem um modelo de contagem semelhante aos Pontos de Função, mas que seja sensível a essa dificuldade.

Segundo SPR (2002a) e Jones (1997), em 1986, o *Software Productivity Research, Inc*. desenvolveu um método experimental para aplicação da lógica de Pontos de Função em sistemas de software como os sistemas operacionais, sistemas comutadores de telefone, e semelhantes. Para evitar confusão com o modelo de ponto de função, esta alternativa experimental foi chamada *Feature Points*, ou Pontos de Particularidade. Considerando que os resultados iniciais foram favoráveis, o modelo foi aplicado experimentalmente a alguns tipos de software:

- Software de tempo real, como sistemas de defesa com mísseis;
- Software de sistemas, como sistemas operacionais;
- Software embutido, como pacotes de navegação com radar;
- Software de comunicação, como sistemas de comutação por telefone;
- Software de controle de processos, como controladores de refinaria;
- Software de engenharia, como CAD, CIM, simulação discreta, software matemático, entre outros.

Como o modelo de Pontos de Particularidade considera a complexidade algoritmica do software, julga-se conveniente apresentar uma definição de **algoritmo**. Para SPR (2002a), um algoritmo pode ser definido como um conjunto de regras que devem ser completamente expressas para resolver um problema computacional significativo.

Segundo Pressman (1995) e Azevedo (1999), para computar o Ponto de Particularidade, procede-se como no Ponto de Função, acrescentando-se uma nova característica, os algoritmos, com peso fixo 3.

Há uma divergência entre Pressman (1995) que afirma que este modelo deve reduzir (e fixar) os pesos empíricos das outras características como mostrado na Tabela 20 e SPR (2002a) que afirma que quando Pontos de Particularidade e Pontos de Função são usados em projetos comuns, os resultados costumam ser **idênticos**. Calvert (1996) concorda com SPR (2002a) afirmando que Pontos de Função e Pontos de Particularidade representam a funcionalidade ou utilidade de um programa e que em projetos de software convencional e de sistemas de informação as duas medidas produzem **os mesmos resultados**.

Tabela 20. Modelo *Feature Points*

Características	Qtde	Pesos	Subtotal de Pontos
Número de algoritmos		3	
Número de entradas		4	
Número de saídas		5	
Número de consultas		4	
Número de arquivos de dados		7	
Número de arquivos de interface		7	
Total de pontos não ajustados			
Ajuste de complexidade			
Total de Feature Points			

Fonte: Pressman (1995) e Azevedo (1999)

Segundo Azevedo (1999), como os algoritmos variam em dificuldade e complexidade, torna-se difícil estabelecer sua magnitude. Isso faz com que o peso (inicialmente sugerido como sendo 3) possa variar entre 1 e 10, ou seja, algoritmos que usam operações matemáticas muito simples podem usar peso 1, enquanto que, algoritmos que exigem equações complexas e/ou processamento lógico e matemático muito difícil podem ter peso 10.

Neste trabalho, calcular-se-á os Pontos de Particularidade exatamente da mesma forma que os Pontos de Função, acrescentando-se apenas os algoritmos para softwares mais complexos.

Para Calvert (1996) em sistemas complexos, os Pontos de Particularidade produzem, freqüentemente, números que são 20 a 35 por cento mais altos que os Pontos de Função.

2.9.3. Métricas Orientadas à Complexidade

2.9.3.1. CIÊNCIA DO SOFTWARE DE HALSTEAD *(HALSTEAD´S SOFTWARE SCIENCE)*

De acordo com Kan (1995), a premissa da ciência do software de Halstead é que qualquer tarefa de programação consiste em selecionar e organizar num programa, um número finito de símbolos que são unidades sintáticas básicas distinguíveis por um compilador. Um programa de computação, de acordo com este modelo, é considerado uma coleção de símbolos que podem ser classificados como operadores ou operandos.

Conforme Pressman (1995), esse modelo usa um conjunto de medidas primitivas que pode ser calculado depois que o código for gerado, ou estimado assim que o projeto estiver completo. São elas:

n_1: o número de operadores distintos que aparecem num programa;
n_2: o número de operandos distintos que aparecem num programa;
N_1: o número total de ocorrências de operadores;
N_2: o número total de ocorrências de operandos.

Essas medidas são usadas para desenvolver expressões para o comprimento global do programa, o volume mínimo potencial para um algoritmo, o volume real (número de bits exigido para especificar um programa), o nível de programa (uma medida de complexidade de software), o nível de linguagem (uma constante para uma determinada linguagem) e outras características como: o esforço de desenvolvimento, o tempo de desenvolvimento e até mesmo uma projeção do número de falhas no software (PRESSMAN, 1995).

Nesse modelo, o comprimento (N) pode ser calculado pela equação (14):

$$N = n_1 \times \log_2 \times n_1 + n_2 \times \log_2 \times n_2 \qquad (14)$$

O volume (V) do programa pode ser calculado pela equação (15):

$$V = N \times \log_2 \times (n_1 + n_2) \qquad (15)$$

Esse volume pode variar conforme a linguagem de programação, e representa o volume de informação (em bits) exigido para especificar um programa (PRESSMAN, 1995).

Boehm (1981) apresenta a equação (16) para o cálculo do esforço (E):

$$E = \frac{n_1 \times N_2 \times V}{2 \times S \times n_2} \qquad (16)$$

Pressman (1995) afirma que a relação entre o volume da forma mais compacta de um programa e o volume do programa real é chamado de índice volumétrico (L), calculado pela equação (17):

$$L = \left(\frac{2}{N_1} \right) \times \left(\frac{n_2}{N_2} \right) \qquad (17)$$

O autor desse modelo propõe que cada linguagem possa ser categorizada por um nível de linguagem (variável conforme a linguagem) e que esse nível seja constante para determinada linguagem, mas há quem diga que o nível de linguagem depende tanto da linguagem, quanto do programador (PRESSMAN, 1995).

2.9.3.2. NÚMERO CICLOMÁTICO DE MCCABE *(MCCABE´S CYCLOMATIC NUMBER)*

Conforme Pressman (1995), a medida de complexidade de software proposta por Thomas McCabe baseia-se numa representação do fluxo de controle de um programa. A Figura 4 mostra um grafo exemplificando a descrição do fluxo de controle. Cada letra envolta em um círculo representa uma ou mais instruções do código fonte e as setas de ligação mostram o fluxo de controle, propriamente dito.

Uma técnica que pode ser usada para computar a métrica da complexidade ciclomática V(G) é determinar o número de regiões num grafo planar, contando todas as áreas delimitadas (na Figura 4, R_2 .. R_5), mais a área não limitada (R_1). O grafo representado na Figura 4 possui complexidade ciclomática V(G) = 5 (PRESSMAN, 1995). Segundo Agena (2000), na prática essa métrica é normalmente equivalente a um, mais o número de decisões no programa.

Kan (1995) diz que a medida de complexidade ciclomática de McCabe foi projetada para indicar a testabilidade e a manutenibilidade de um programa. É o número de caminhos

independentes linearmente encontrados num programa e pode ser usado para indicar o esforço exigido para testar um programa. Para determinar os caminhos, os procedimentos do programa são representados como um grafo fortemente conectado com uma única entrada e um único ponto de saída, como pode ser visto na Figura 4.

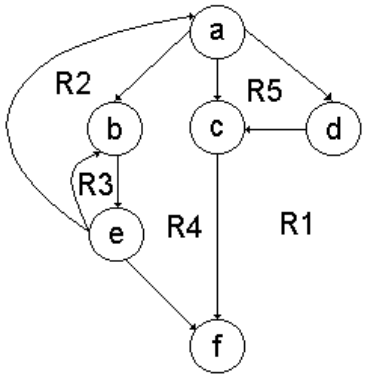

Figura 4. Complexidade do grafo de fluxo de controle
Fonte: Pressman (1995)

Essa métrica pode ainda ser utilizada como indicação quantitativa do tamanho modular máximo. Quando a complexidade ciclomática de um módulo for superior a dez, esse módulo será extremamente difícil de ser testado adequadamente (PRESSMAN, 1995).

2.10. COMENTÁRIOS FINAIS

Neste capítulo foi realizada uma ampla revisão bibliográfica sobre Custos de Software apresentando desde os conceitos fundamentais de contabilidade de custos até as principais técnicas utilizadas em estimativas de custo de software.

Foram elencadas as principais atividades na estimativa de custo de software e destacadas as principais técnicas utilizadas, ou seja, Julgamento Especialista (também conhecida como Parecer Técnico), Analogia e Modelos Algorítmicos que foram divididos, conforme sua orientação: tamanho, função e complexidade.

No próximo capítulo será realizada uma análise dessas técnicas individualmente e, em seguida, comparativamente.

3. ANÁLISE DAS TÉCNICAS

Este capítulo trata, inicialmente, da análise individual das técnicas estudadas, procurando destacar as vantagens e desvantagens de cada uma. Posteriormente, é realizada uma análise comparativa buscando analisar estas técnicas em conjunto, a partir de alguns critérios.

3.1. ANÁLISE INDIVIDUAL

3.1.1. Julgamento Especialista ou Parecer Técnico

A técnica de Julgamento Especialista parece ser bastante subjetiva, especialmente quando se considera que é realizada por especialistas, ou melhor, por pessoas que, por mais que sejam consideradas “especialistas” não perdem sua condição humana. Assim, estão expostos (e podem reagir de maneiras diferentes) a pressões de seus superiores ou de clientes, possuem problemas pessoais/particulares, podem apresentar desvios de personalidade e estão sujeitos ao estresse da vida moderna, só para citar alguns exemplos. Resumindo, estes especialistas podem chegar a conclusões diferentes quando apresentados duas vezes ao mesmo problema, dependendo de fatores externos e não intrísecos ao dito problema.

Bournemouth (1997), porém, afirma que a técnica *Wideband-Delphi* (uma forma de se aplicar o Julgamento Especialista) é muito boa para remover as políticas de uma estimativa, já que os especialistas não se comunicam sobre sua estimativa particular. Isto sem contar que esta técnica filtra opiniões extremas realizando uma estimativa imparcial e a discussão (realizada apenas em grupo) torna-se vantajosa, porque nenhum ponto da estimativa é negligenciado. Esse consenso ajuda a eliminar tendências em estimativas produzidas por pessoas que se dizem especialistas (e, provavelmente não o sejam), estimadores sem experiência ou pessoas influentes que tenham objetivos divergentes ou obscuros (WIEGERS, 2000).

Por outro lado, o especialista pode (e deve) fazer inferências baseado em sua experiência passada, o que não é possível quando se aplica, por exemplo, os Modelos Algorítmicos que são baseados em equações matemáticas.

Além disso, conforme Wiegers (2000), a técnica *Wideband-Delphi* pode ser usada para qualquer estimativa, como por exemplo, o número de meses necessários para implementar um sistema específico, o número de linhas ou de classes num programa completo.

De qualquer forma, para Boehm (1981), as vantagens e desvantagens dessa técnica são complementares às vantagens e desvantagens da técnica dos Modelos Algorítmicos, ou seja, uma desvantagem da técnica de Julgamento Especialista pode ser compensada por uma vantagem de outra técnica.

Provavelmente, a maior dificuldade na aplicação desta técnica seja encontrar os especialistas em estimativa de custos de software.

3.1.2. Analogia

Para Boehm (1981), as estimativas por Analogia podem ser feitas em nível de um projeto total ou em nível de subsistema. Em nível de projeto total, tem-se a vantagem de que serão considerados todos os componentes de custo do sistema (inclusive os custos de integrar os subsistemas), enquanto em nível de subsistema, tem-se a vantagem de realizar uma avaliação mais detalhada das semelhanças e diferenças entre o projeto novo e os projetos que já foram completados.

Na prática, pode-se verificar que alguns módulos de um projeto novo são muito semelhantes a alguns módulos de um determinado projeto já completado e que outros módulos deste projeto novo são idênticos a outros módulos de outros projetos (também já completados), conforme mostrado na Figura 5.

Fenton e Pfleeger (1997) afirmam que esta análise normalmente é documentada, estando assim disponível para uso em estimativas subseqüentes; e pode ser revisada para encontrar razões que justifiquem o porquê de uma estimativa não ter sido tão precisa quanto foi esperada.

Esta técnica força o estimador a identificar as características fundamentais de cada projeto, assim o modelo de esforço começa a aparecer (FENTON e PFLEEGER, 1997). Identificar as

características relevantes de cada software, contudo, pode ser um problema, porque o que é relevante num projeto pode não o ser em outro.

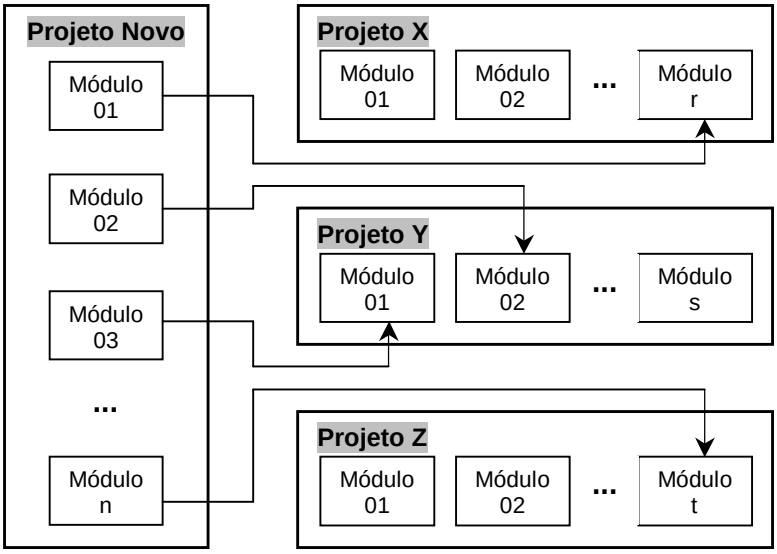

Figura 5. Exemplo de semelhança entre alguns módulos de projetos

Outro problema na aplicação/utilização dessa técnica para estimar custos de software é que com a adoção de novas técnicas e ferramentas de desenvolvimento a utilização da Analogia tende a se tornar uma tarefa ainda mais complexa do que já é.

Além disso, manter um banco de dados histórico e atualizado com as características dos projetos já desenvolvidos pela organização pode revelar-se uma tarefa trabalhosa, porque, freqüentemente, não se cultiva o hábito de manter informações sobre projetos já concluídos e isso pode implicar numa mudança de cultura na empresa.

Segundo Schofield e Shepperd (1995), embora o conceito de estimar por analogia seja relativamente direto, há vários problemas a serem considerados:

- tem-se que determinar como os projetos serão melhor caracterizados. Algumas possibilidades de caracterização incluem o domínio de aplicação, o número de entradas, o número de telas e assim sucessivamente. Note-se que variáveis diferentes devem ser medidas em escalas diferentes (o tipo de domínio de aplicação é uma variável categórica e

assim será medido em uma escala nominal, o número de entradas será medido em uma escala absoluta);

- uma vez que os projetos foram caracterizados, como se determinará a semelhança e a relação entre os mesmos? Quanta confiança se pode colocar nas analogias? Tem-se ainda o problema de saber quantas analogias devem ser procuradas. Pouquíssimas podem conduzir a projetos que diferem muito do que está sendo estimado; muitas podem conduzir à diluição do efeito das analogias mais próximas;
- como devem ser usados os valores de esforço conhecidos dos projetos análogos para derivar uma estimativa para o projeto novo? Algumas possibilidades incluem médias ponderadas (dando mais influência às analogias mais próximas).

A principal vantagem da estimação por Analogia é que a estimativa está baseada na experiência real de um ou de vários projetos. Essa experiência pode ser estudada para determinar diferenças específicas do projeto novo, e o provável impacto de custo dessas diferenças. A principal desvantagem dessa técnica é que não está claro em que grau o projeto prévio é realmente o representante das exigências, técnicas, pessoal, e funções a serem executados pelo software no novo projeto (BOEHM, 1981).

3.1.3. Modelos Algorítmicos

3.1.3.1. LINHAS DE CÓDIGO (*LINES OF CODE*)

Cockburn (1999) e Roetzheim (2000a) afirmam que o uso da quantidade de LOC como métrica de software foi popularizado pelo modelo COCOMO. É importante salientar que o Modelo de Estimativa de Putnam também utiliza linhas de código.

O uso de LOC para estimar custos de software, ou simplesmente, para medi-lo é bastante controverso. Os argumentos dessa polêmica são os mais variados possíveis e boa parte mostra-se coerente.

De um lado, a contagem de linhas de código fonte parece ser uma forma muito natural de se medir software, principalmente porque, independentemente da linguagem de programação empregada, ou do domínio da aplicação, qualquer software apresenta LOC.

Por outro lado, a quantidade de LOC pode ser altamente dependente da linguagem de programação utilizada (mais, ou menos, poderosa) e da competência e/ou experiência do programador e das técnicas e ferramentas de desenvolvimento utilizadas. Por exemplo, Fenton e Pfleeger (1997) alertam para o problema que, numa determinada linguagem de programação, uma linha de código pode ser contada várias vezes, dependendo da técnica de contagem utilizada.

Pode-se ainda ter um software com poucas linhas de código fonte, mas de custo muito alto, porque sua complexidade resultou num esforço muito grande e/ou na necessidade de contratação de pessoal altamente especializado (e caro) para a solução do problema.

Boa parte dos custos de desenvolvimento de software são fixos, ou seja, não dependem da quantidade de linhas de código fonte gerada. Herron e Garmus (1999), por exemplo, afirmam que o resultado da utilização de LOC é impreciso para a maioria dos grandes projetos, e que o código é um aspecto limitado (talvez 5 a 15% do esforço de projeto total) para projetos de desenvolvimento de software.

Em razão disso, Trindade (2000) recomenda que haja uma cultura de documentação histórica de dados, constantemente atualizados e avaliados, padronização das rotinas e utilização de linguagens específicas e não mutáveis. Com o crescente aparecimento de novas e melhores linguagens de programação, no entanto, torna-se, no mínimo, incoerente utilizar sempre a mesma linguagem. E, com a rápida evolução e modernização observadas nas linguagens existentes, não é possível considerá-las “imutáveis” (tome-se como exemplo as versões 1.0 e 6.0 do Delphi).

Jones (1997) e SPR (2002b) afirmam que o desenvolvimento de software envolve uma percentagem significativa de custos fixos que não são associados à atividade de programar. Quando se usa uma linguagem de programação mais poderosa, o resultado é uma redução no número de linhas de código. Porém, os requisitos, as especificações, a documentação do usuário, e muitos outros elementos de custo tendem a se comportar como custos fixos. A Tabela 21, por exemplo, mostra duas versões de um mesmo projeto: o caso A foi desenvolvido em ASSEMBLER, já o caso B foi desenvolvido em FORTRAN.

Tabela 21. O paradoxo da métrica de linhas de código fonte (LOC).

Atividade	Caso A Assembler (10.000 LOC)	Caso B Fortran (3.000 LOC)
Requisitos	02 semanas	02 semanas
Projeto	03 semanas	03 semanas
Codificação	10 semanas	03 semanas
Integração/Testes	05 semanas	03 semanas
Documentação do Usuário	02 semanas	02 semanas
Gerência/Suporte	03 semanas	02 semanas
Tempo Total	**25 semanas**	**15 semanas**
Custo Total	**125,000.00 $**	**75,000.00 $**
Custo por LOC	**12.50 $**	**25.00 $**
LOC/pessoas-mês	**400**	**200**

Fonte: Jones (1997) e SPR (2002b)

No caso A, uns 40% dos custos concentravam-se na atividade de codificação, enquanto no caso B este percentual foi de apenas 20%. Os custos da não codificação tendem a se comportar como custos fixos, invalidando esta métrica como indicador econômico (JONES, 1997) e (SPR, 2002b).

3.1.3.2. MODELO DE ESTIMATIVA DE PUTNAM

Possivelmente uma das maiores desvantagens deste modelo seja a sua não adequação para pequenos projetos de software, o que restringe bastante sua popularização.

Outro fator limitador é a dificuldade na definição da constante C (constante do estado de tecnologia): ou se usa um dos três valores sugeridos por Pressman (1995), ou então calcula-se o valor desta constante para a organização, o que exige um levantamento de dados históricos na empresa sobre desenvolvimento de software. Dependendo do grau de organização da empresa, esta tarefa pode revelar-se muito simples, extremamente complexa, ou até mesmo virtualmente impossível.

Uma contribuição importante desse modelo é a definição da “região impossível” como sendo o prazo impossível de ser cumprido. Roetzheim (2000b) afirma que é impossível concluir um projeto num prazo inferior a 75% do prazo que seria considerado normal. Outro ponto interessante é que, aumentar o tempo de desenvolvimento para além do normal (até o dobro deste) pode reduzir os custos em até 50%.

Bournemouth (1997) aponta como vantagem desse modelo o uso de programação linear para considerar restrições de desenvolvimento, tanto no custo, quanto no esforço e apresenta como desvantagens o fato de as estimativas serem muito sensíveis ao fator tecnologia e a não adequação desse modelo para pequenos projetos.

3.1.3.3. MODELO DE CUSTO CONSTRUTIVO (*CONSTRUCTIVE COST MODEL* - COCOMO)

Uma das desvantagens do modelo COCOMO, segundo Peters e Pedrycz (2001), é que ele está apoiado em duas suposições:

- o desenvolvimento do software estaria baseado no paradigma do Ciclo de Vida Clássico (também conhecido como Modelo Cascata);
- o gerenciamento da equipe de desenvolvimento seria de alto padrão (leia-se “sem tempo ocioso”).

O modelo COCOMO pode ser subdividido em três: básico, intermediário e detalhado.

Para utilização do Modelo COCOMO **Básico** é necessário que se tenha apenas uma estimativa da quantidade de linhas de código fonte para que se possa estimar o esforço e o tempo necessário para o desenvolvimento do projeto de software em questão, ou seja, é muito fácil usar este modelo, porém, segundo Bournemouth (1997), é difícil estimar com precisão a quantidade de linhas de código no início do projeto (quando mais se precisa das estimativas de esforço).

O Modelo COCOMO **Básico** não considera nenhuma particularidade do projeto de software, nem do pessoal de desenvolvimento, nem do hardware onde o software será instalado. Para as situações onde estas particularidades devem ser consideradas, o mais recomendável é que se utilize o Modelo COCOMO **Intermediário** que apresenta um conjunto de 15 (quinze) direcionadores de custo agrupados em 04 (quatro) categorias: atributos do produto, atributos do hardware, atributos de pessoal e atributos de projeto.

Uma dificuldade na utilização do Modelo COCOMO **Intermediário** pode ser a classificação de cada um dos direcionadores de custo em uma escala que varia de “muito baixo” até “extremamente elevado”. Conforme Bournemouth (1997), o modelo é vulnerável à má classificação do modo de desenvolvimento e o sucesso das estimativas depende em grande

parte de afinar o modelo às necessidades da organização, usando dados históricos que nem sempre estão disponíveis.

Uma limitação deste modelo (**intermediário**) é que os valores dos multiplicadores de esforço de cada direcionador de custo são fixos para todo o projeto, ou seja, não consideram cada fase do desenvolvimento do software em separado.

Segundo Peters e Pedrycz (2001), o Modelo COCOMO **Intermediário** é a variação mais utilizada do Modelo COCOMO e trata-se, na realidade, de um "refinamento" do Modelo COCOMO **Básico**.

O Modelo COCOMO **Detalhado**, além de apresentar multiplicadores de esforço para cada fase do desenvolvimento do software, traz ainda uma hierarquia de níveis que facilita a definição do valor dos direcionadores quando os efeitos podem ser agrupados (num dos três níveis: módulo, subsistema e sistema).

Conforme Boehm (1981), o Modelo COCOMO **Detalhado** ainda apresenta outras vantagens, como por exemplo, um procedimento para ajustar a fase de distribuição do tempo de desenvolvimento.

Uma das maiores dificuldades na aplicação e, provavelmente para a não popularização desta variação do Modelo COCOMO (**Detalhado**), seja a falta de bibliografia sobre o assunto. A única obra encontrada que detalha a utilização deste modelo foi Bohem (1981) – o próprio criador do modelo.

Conforme Bournemouth (1997), de uma forma geral, esse modelo é transparente, ou seja, o estimador pode ver como ele trabalha e, além disso, os direcionadores de custo são particularmente úteis para o estimador entender o impacto de diferentes fatores que afetam os custos do projeto.

3.1.3.4. PONTOS DE FUNÇÃO (*FUNCTION POINTS*)

A contagem dos Pontos de Função usa, como principal fonte de informação, os documentos de especificação do software. Tais documentos estão disponíveis numa fase inicial do projeto,

ou seja, no início do processo de desenvolvimento de software. Conforme Paula Filho (2001), a especificação dos requisitos de um software apresenta dados suficientes para que se calcule a quantidade de Pontos de Função. Em decorrência disso, estimativas de custo de software, como por exemplo, estimativas de esforço e tempo de desenvolvimento podem ser feitas logo no início do projeto, ou seja, quando a necessidade de se fazer tais estimativas é maior.

Braga (1996) diz que a contagem de Pontos de Função nas fases iniciais do desenvolvimento é uma estimativa da funcionalidade desejada. Segundo esse autor, o objetivo da estimativa não é a precisão, porém estimar é de fundamental importância para o dimensionamento de recursos, para um controle efetivo e para a antecipação de problemas, ou seja, há necessidade de estimar prazos, esforços e custos.

Os Pontos de Função são uma medida da funcionalidade do software, ou seja, independentemente da tecnologia utilizada pode-se calcular estes pontos. Tanto Peters e Pedrycz (2001), quanto Santos et al (2001) concordam com esta afirmação.

De todos os Modelos Algorítmicos estudados neste trabalho, com certeza o Modelo de Pontos de Função é o que conta com bibliografia mais farta, possivelmente devido a sua grande popularidade. Para se ter uma idéia dessa popularidade, o governo brasileiro e alguns governos estaduais e municipais, quando precisam terceirizar o desenvolvimento de software, lançam editais abrindo concorrência pública para contratação por Pontos de Função. Além disso, a utilização comercial deste modelo é tão grande que existe no Brasil uma entidade conhecida como BFPUG-Brazilian Function Point Users Group (Grupo de Usuários de Pontos de Função Brasileiro)[5] que está ligada a uma organização internacional chamada de IFPUG-International Function Point Users Group (Grupo de Usuários de Pontos de Função Internacional)[6]. Estes grupos funcionam como fóruns de discussão e troca de informações e experiências sobre a utilização de Pontos de Função entre seus associados.

Para Fenton e Pfleeger (1997), se a técnica de Pontos de Função realmente captura a funcionalidade, então ela é vantajosa por várias razões:

---

[5] O *site* do BFPUG na *Word Wide Web* é <http://www.bfpug.com.br>.

[6] O *site* do IFPUG na *Word Wide Web* é <http://www.ifpug.org>.

- esta medida reflete a produtividade com mais precisão;
- a produtividade da equipe de desenvolvimento pode ser avaliada em qualquer fase do ciclo de vida do software;
- o progresso pode ser medido comparando-se os Pontos de Função completos e incompletos.

Por outro lado, o Modelo de Pontos de Função também apresenta algumas desvantagens, por exemplo, o que significa fisicamente um Ponto de Função? Na verdade, o Ponto de Função não tem nenhum significado físico ou direto, segundo Pressman (1995) “é só um número”.

Este modelo não considera nenhuma particularidade do pessoal de desenvolvimento, nem de complexidade algorítmica do projeto. Parece poder ser aplicado apenas a Sistemas de Informação convencionais, sem nenhum tipo de detalhe em especial.

Além disso, a contagem de Pontos de Função envolve uma certa dose de subjetividade, porque as funções são contadas e consideradas a partir do ponto de vista do usuário, ou seja, as soluções técnicas encontradas (por exemplo, a solução de um relacionamento de “muitos-para-muitos”) devem ser analisadas com muita cautela.

Com relação à reutilização de código e/ou de componentes, o Modelo de Pontos de Função é falho, porque, simplesmente não considera esta possibilidade. Com relação à manutenção, o problema é o “trabalho dobrado”, porque se deve calcular os Pontos de Função do sistema atual e depois de realizada a manutenção. Deve-se ainda fazer alguns ajustes para só então calcular os Pontos de Função específicos da manutenção.

Outro problema deste modelo é que ele não considera (pelo menos, não no modelo oficial) a utilização de ferramentas automatizadas, geradores de código, ou ferrramentas CASE. Peters e Pedrycz (2001) ainda afirmam que o Modelo de Pontos de Função ignora as tecnologias de redução do esforço como ambientes integrados de desenvolvimento de software, descrições de projetos executáveis, bibliotecas de reuso e orientação a objetos.

3.1.3.5. PONTOS DE PARTICULARIDADE *(FEATURE POINTS)*

Segundo Pressman (1995), o modelo de Pontos de Particularidade foi proposto por Jones, como uma extensão do modelo de Pontos de Função, possibilitando que essa medida seja utilizada em aplicações de engenharia de software e de sistemas. A medida do Ponto de Particularidade acomoda aplicações em que a complexidade algorítmica é elevada, como por exemplo, em aplicações de tempo real, de controle de processos e de software embutido.

Mesmo sendo uma extensão do Modelo de Pontos de Função, este modelo não apresenta a mesma popularidade (que Pontos de Função), tanto que a bibliografia e as pesquisas/publicações sobre a utilização deste modelo são poucas. Além disso, sua utilização comercial é praticamente inexistente.

A justificativa da proposta do Modelo de Pontos de Particularidade é cobrir uma deficiência do Modelo de Pontos de Função, que não leva em consideração o fator complexidade de algoritmos (AZEVEDO, 1999). Ainda assim, não existem indicativos claros e eficientes para se considerar a complexidade dos algoritmos, tanto da sua identificação/contagem, quanto dos pesos de cada um.

3.1.3.6. CIÊNCIA DO SOFTWARE DE HALSTEAD *(HALSTEAD´S SOFTWARE SCIENCE)*

Tanto Pressman (1995), quanto Agena (2000) afirmam que esta métrica pode ser derivada antes da codificação, durante a fase de projeto. Nenhum dos dois autores explica claramente como isso pode ser feito, entretanto.

Não são consideradas a experiência (ou falta dela) do desenvolvedor, nem as diferenças de linguagens de programação. Na prática, um mesmo programa contruído em linguagens de programação diferentes e/ou usando ferramentas diferentes podem apresentar valores (para este modelo) completamente divergentes. Além disso, dependendo do estilo de programação e da experiência do programador, o software pode apresentar mais (ou menos) operadores e operandos e pode, conseqüentemente, calcular valores diferentes para o comprimento, o volume e o índice volumétrico do programa.

Não se pode ainda deixar de considerar que além dos operandos e operadores (que fazem parte do código), muitos outros fatores influenciam no esforço e no tempo de desenvolvimento de um software. Estes fatores são conhecidos como custos fixos, exatamente porque não dependem, por exemplo, da quantidade de linhas de código, ou da quantidade de operadores e operandos.

Outro ponto que merece atenção é com relação ao reuso de código já desenvolvido. A literatura consultada nada diz sobre contar (ou não contar) este código que já está pronto, mas que, na realidade, provavelmente necessitará de adaptações para que seja utilizado.

A falta de popularidade desse modelo deve ser atribuída à sua complexidade de aplicação, além de ser bastante trabalhoso contar os operandos e operadores. Outro ponto negativo é a falta de significado (pelo menos para o usuário) de termos como “comprimento de programa” e “índice volumétrico”, por exemplo.

Conforme Kan (1995), o Modelo da Ciência do Software de Halstead é criticado por muitos pesquisadores por causa da metodologia utilizada e pela forma como suas equações foram derivadas.

3.1.3.7. NÚMERO CICLOMÁTICO DE MCCABE *(MCCABE´S CYCLOMATIC NUMBER)*

Como o número de regiões aumenta com o número de caminhos de decisão e laços, esta métrica pode oferecer uma medida quantitativa da dificuldade de fazer testes e uma indicação da confiabilidade final. Estudos experimentais indicaram relação entre esta métrica e o número de erros existentes no código fonte, bem como o tempo necessário para descobrir e corrigir esses erros (PRESSMAN, 1995). Nada se diz sobre o tempo necessário para desenvolver o software, tampouco o esforço necessário para esta tarefa, o que leva a se pensar na não adequação deste modelo para tal fim.

Outro ponto a ser considerado é a dificuldade de se conseguir contar o número de regiões formadas pelo fluxo de controle do programa num dos estágios iniciais do desenvolvimento do projeto (a menos que o projeto seja muito bem documentado e detalhado).

Essa métrica é crítica por não considerar a complexidade de fluxo de dados, nem a complexidade de programas não estruturados (AGENA, 2000), ou seja, aqueles escritos em outros paradigmas de linguagens de programação, como por exemplo, LISP e PROLOG.

A quase inexistência de publicações sobre esse modelo indica sua falta de popularidade, tanto entre os pesquisadores, quanto entre possíveis usuários.

A aplicação desse modelo pode ser bastante trabalhosa e demorada. Por outro lado, deve-se admitir que, se o programa encontra-se bem estruturado, trata-se de um modelo de fácil aplicação.

3.2. ANÁLISE COMPARATIVA

Ao se analisar comparativamente cada uma das técnicas e cada um dos modelos estudados e apresentados nas seções 2.7, 2.8 e 2.9 e analisados individualmente na seção 3.1, observa-se que cada um apresenta pontos positivos e negativos e que, em geral, os aspectos negativos de uma técnica (ou de um modelo) podem ser "compensados" pelas vantagens de outra técnica (ou de outro modelo).

3.2.1. Utilização Comercial, Pesquisas e Publicações

Quanto à utilização, pode-se dizer que a técnica de Julgamento Especialista é muito empregada, muitas vezes inadequadamente, sem seguir regras muito bem definidas, ou seja, sem usar *Delphi*, ou *Wideband-Delphi*.

Os Modelos Algorítmicos concentram a maior parte das pesquisas em estimativas de esforço e tempo de desenvolvimento de software. Entre esses modelos destaca-se a contagem de Linhas de Código, usada com freqüência sem muitos critérios como, por exemplo, não contar as linhas em branco, ou de comentários, mas é muito comum as pessoas se interessarem pela quantidade de linhas de código deste, ou daquele software.

Destaca-se ainda a popularidade do Modelo de Pontos de Função (obedecendo aos critérios internacionais do modelo) que possui um grupo internacional de usuários (IFPUG) que chega

a fornecer certificação para os especialistas na área que se submeterem e forem aprovados em exames de certificação. Além disso, no Brasil, por exemplo, o governo está contratando o desenvolvimento terceirizado de software em Pontos de Função. É importante ainda destacar que são muitas as pesquisas sobre a aplicação de Pontos de Função, tanto no Brasil, quanto no exterior, fato comprovado pela grande quantidade de publicações de artigos e livros sobre o assunto.

Os modelos conhecidos como Ciência do Software de Halstead e Número Ciclomático de McCabe são os menos populares, pois são de aplicação trabalhosa e contam com poucas publicações, tanto de artigos, quanto em livros.

3.2.2. Independência da Linguagem de Programação

Pode-se considerar que o Parecer Técnico é independente da linguagem de programação utilizada, pois os especialistas podem considerar esta independência ao fazer suas estimativas.

Ao se julgar que a linguagem de programação utilizada é uma característica importante do software, é difícil aplicar a técnica de Analogia para estimar-se o esforço necessário e o tempo de desenvolvimento, independentemente da linguagem utilizada.

Os Modelos de Linhas de Código, Estimativa de Putnam, COCOMO, Ciência do Software de Halstead e Número Ciclomático de McCabe, são totalmente dependentes da linguagem de programação utilizada, porque todos eles se servem da quantidade de LOC do software, ou de valores dependentes de LOC, para calcular suas estimativas.

Por outro lado, os Modelos de Pontos de Função e de Pontos de Particularidade são totalmente independentes da linguagem de programação utilizada.

3.2.3. Consideração do Uso de Técnicas e Ferramentas Automatizadas

O especialista, na técnica de Julgamento Especialista, pode (e deve) considerar o uso de novas (ou diferentes) técnicas para o desenvolvimento do projeto em questão. Com a técnica de Analogia, entretanto, a consideração do uso de novas ferramentas pode inviabilizar as

estimativas, porque se as ferramentas são novas não há nenhum projeto já desenvolvido na empresa que possa ser usado como base para se encontrar as Analogias.

O Modelo de Estimativa de Putnam apresenta a constante C, que é uma constante do "estado de tecnologia" e pode ser ajustada para a realidade da empresa. Quanto maior for o valor desta constante, melhor é o ambiente de desenvolvimento, no sentido de se utilizar ferramentas e técnicas automatizadas.

Quanto ao Modelo COCOMO, apenas a versão "básica" não considera a utilização dessas ferramentas. Tanto a versão COCOMO Intermediário, quanto a versão COCOMO Detalhado consideram essas questões em seus direcionadores e multiplicadores de esforço de desenvolvimento.

Os demais Modelos Algorítmicos não consideram o uso de técnicas e ferramentas automatizadas de desenvolvimento. Apenas os Modelos de Pontos de Função e Pontos de Particularidade não terão seus valores modificados por este fator.

3.2.4. Consideração da Experiência da Equipe de Desenvolvimento

Na técnica de Julgamento Especialista, os estimadores podem considerar a experiência da equipe de desenvolvimento, mas nada garante que isso realmente acontecerá.

Quando se usa a técnica de Analogia é possível incluir a experiência da equipe de desenvolvimento como uma característica do projeto, mas isto não é uma exigência da técnica, ou seja, os estimadores precisam decidir que esta característica é relevante.

Entre os Modelos Algorítmicos, o único que considera, explicitamente, a experiência da equipe de desenvolvimento é o Modelo COCOMO (Intermediário e Detalhado). Apenas os Modelos de Pontos de Função e Pontos de Particularidade, porém, não sofrerão alteração em seus valores com a experiência (ou não) da equipe de desenvolvimento. Os demais modelos, apesar de não considerarem este fator, poderão ter seus valores alterados por ele.

3.2.5. Complexidade dos Algoritmos

O especialista pode considerar a complexidade do software, entretanto, esta consideração não é uma regra explícita da técnica de Julgamento Especialista. Quanto à técnica de Analogia, a complexidade pode ser uma característica do software, mas isso depende de o estimador considerar esta característica como relevante.

Dos Modelos Algorítmicos, os Modelos de Linhas de Código, Estimativa de Putnam, COCOMO Básico e Pontos de Função não consideram, explicitamente, a complexidade dos algoritmos do software.

3.2.6. Independência do Tamanho do Projeto

É provável que, quanto maior o tamanho do projeto, mais trabalhoso se torna aplicar as técnicas de Julgamento Especialista e Analogia, mas isso não chega a comprometer os resultados das estimativas.

O mesmo acontece com os Modelos Algorítmicos que, em geral, são bem tolerantes ao tamanho do projeto. Exceção deve ser feita ao Modelo de Estimativa de Putnam que não se adapta bem a projetos de software de pequeno porte.

3.2.7. Aplicabilidade em Projetos de Manutenção

No uso de Parecer Técnico, o especialista sempre vai poder considerar/adaptar sua técnica para fazer estimativas para projetos de manutenção. Da mesma forma, na aplicação da técnica de Analogia, o estimador pode construir um banco de dados histórico sobre projetos de manutenção e estabelecer características relevantes para estes projetos, como por exemplo, a quantidade de código aproveitado, adaptado e descartado.

Os Modelos Algorítmicos são para aplicação em projetos novos, exceto o Modelo de Pontos de Função que apresenta regras claras (mas não discutidas neste trabalho) para aplicação em projetos de manutenção de software e o Modelo de Estimativa de Putnam que parte do esforço para o desenvolvimento e manutenção do software, conforme apresentado na seção 2.9.1.2.

3.2.8. Etapa do Desenvolvimento onde pode ser Usado

As técnicas de Julgamento Especialista e Analogia podem ser utilizadas a partir da etapa de Análise de Requisitos. Todos os Modelos Algorítmicos podem ser utilizados a partir da etapa de Projeto (quanto mais completa, mais precisas serão as estimativas).

No caso dos modelos de Linhas de Código, Estimativa de Putnam e COCOMO o Projeto é necessário para que se possa melhor estimar a quantidade de LOC. Já para os Modelos de Pontos de Função e Pontos de Particularidade para que as função sejam melhor identificadas. Quanto à Ciência do Software de Halstead e ao Número Ciclomático de McCabe, tanto para se estimar os operandos e operadores, quanto para se estimar o número de decisões. Quanto mais completo o projeto, mais precisa será esta estimativa.

3.2.9. Resumo da Análise Comparativa

A Tabela 22 apresenta um resumo das análises realizadas entre as técnicas de Julgamento Especialista, Analogia e Modelos Algorítmicos.

Observadas as vantagens e desvantagens de cada um dos Modelos Algorítmicos, foram selecionados os seguintes modelos para aplicação no estudo de caso (que será apresentado no próximo capítulo): Linhas de Código (*Lines of Code*), Modelo de Estimativa de Putnam, Modelo de Custo Construtivo (*COnstructive COst MOdel* ou COCOMO) Básico, Pontos de Função (*Function Points*) e Pontos de Particularidade (*Feature Points*).

Tabela 22. Resumo da análise comparativa entre as principais técnicas de estimativa de custo de software

CRITÉRIOS / TÉCNICAS		Utilização comercial, pesquisas e publicações	Independência da Linguagem de Programação	Consideração do Uso de Técnicas e Ferramentas Automatizadas	Consideração da Experiência da Equipe de Desenvolvimento	Complexidade dos Algoritmos	Independência do Tamanho do Projeto	Aplicabilidade em Projetos de Manutenção	Etapa do desenvolvimento onde podem ser usados
JULGAMENTO ESPECIALISTA		Muito utilizada e poucas publicações	Sim, mas depende do estimador	Sim, mas depende do estimador	Sim, mas depende do estimador	Sim, mas depende do estimador	Sim	Sim, mas depende do estimador	A partir da Análise de Requisitos
ANALOGIA		Pouco utilizada e poucas publicações	Não, mas depende do estimador	Não	Sim, mas depende do estimador	Sim, mas depende do estimador	Sim	Sim, mas depende do estimador	A partir da Análise de Requisitos
MODELOS ALGORÍTMICOS	Linhas de Código	Muito utilizada e poucas publicações	Não	Não, mas o uso pode interferir nos resultados	Não, mas este fator pode interferir nos resultados	Não, mas este fator pode interferir nos resultados	Sim	Não	A partir da etapa de Projeto
	Modelo de Estimativa de Putnam	Pouco utilizada e poucas publicações	Não	Sim	Não, mas este fator pode interferir nos resultados	Não, mas este fator pode interferir nos resultados	Não se adapta bem a projetos de pequeno porte	Sim	A partir da etapa de Projeto
	COCOMO	Pouco utilizada e poucas publicações	Não	Sim, exceto no COCOMO Básico	Sim, exceto no COCOMO Básico	Sim, exceto no COCOMO Básico	Sim	Não	A partir da etapa de Projeto
	Pontos de Função	Muito utilizada e muitas publicações	Sim	Não	Não	Não	Sim	Sim	A partir da etapa de Projeto
	Pontos de Particularidade	Poucas publicações	Sim	Não	Não	Sim	Sim	Não	A partir da etapa de Projeto
	Ciência do Software de Halstead	Poucas publicações	Não, mas depende indiretamente	Não, mas o uso pode interferir nos resultados	Não, mas este fator pode interferir nos resultados	Sim, mas implicitamente	Sim	Não	A partir da etapa de Projeto
	Número Ciclomático de McCabe	Poucas publicações	Não, mas depende indiretamente	Não, mas o uso pode interferir nos resultados	Não, mas este fator pode interferir nos resultados	Sim, mas implicitamente	Sim	Não	A partir da etapa de Projeto

4. ESTUDO DE CASO

4.1. CÁLCULO DAS ESTIMATIVAS

Segundo Jones (1998), há muitos tipos de ferramentas automatizadas que podem ser usadas para criar estimativas de prazo e esforço para projetos de software, tais como:

- planilhas eletrônicas;
- ferramentas de administração de projeto;
- ferramentas de administração de processo e de metodologias;
- ferramentas de estimativa de custo de software.

Jones (1998) ainda afirma que as ferramentas comerciais de estimativa de software são diferenciadas de todos os outros tipos de ferramentas de administração de projeto de software e de ferramentas de propósito geral, como planilhas eletrônicas, nestes atributos fundamentais:

- contêm bases de conhecimento de centenas ou milhares de projetos de software;
- podem executar predições de tamanho, qualidade e confiança que ferramentas de propósito geral não podem;
- podem ajustar estimativas automaticamente baseadas em ferramentas, linguagens e processos;
- podem predizer custos de manutenção e suporte pós-desenvolvimento;
- podem previnir problemas antes que eles aconteçam.

A partir dos modelos utilizados, do tamanho dos softwares estudados e das ponderações de Jones (1998), optou-se por usar apenas ferramentas de propósito geral (planilhas eletrônicas) neste trabalho.

Para cada software avaliado, foram verificados o esforço e o tempo de desenvolvimento reais. A partir da listagem do código fonte, foi realizada uma análise seguida de marcação com caneta tipo “destaca-texto” das linhas de código que deveriam ser contadas. Não foram consideradas linhas de comentário, declarações, marcação de início e fim de blocos, conforme

Park (1992). Na sequência, foram contadas as LOC marcadas em cada página da listagem e, finalmente, foi realizado o somatório das LOC encontradas.

O Diagrama de Entidade e Relacionamento, o Dicionário de Dados, o Projeto de Telas e Relatórios de cada software foram usados como base para calcular a quantidade de Arquivos Lógicos Internos, de Arquivos de Interface Externa, de Entradas Externas, de Saídas Externas e de Consultas Externas (separados por complexidade funcional: simples, média, complexa). Para auxiliar no cálculo dos Pontos de Função Brutos, o quadro apresentado na Tabela 16 foi montado numa planilha eletrônica. Em seguida, foram avaliados os níveis de influência (Tabela 17) para cada uma das Características Gerais do Sistema (Tabela 18). O Fator de ajuste foi calculado pela equação (12) e, finalmente, a quantidade de PFA-Pontos de Função Ajustados foi calculada pela equação (13).

No cálculo dos Pontos de Particularidade foram considerados para os algoritmos os seguintes pesos: 01-Simples, 05-Médio e 10-Complexo.

Calvert (1996) sugere a equação (18) para o cálculo da produtividade da equipe de desenvolvimento:

$$\text{Produtividade} = S / E \qquad (18)$$

onde:

S: é o tamanho do software (medido em KLOC ou em PFA);

E: é o esforço medido em pessoas-mês.

Assim, para cada software foram calculados dois valores de produtividade: em KLOC/pessoas-mês e em PFA/pessoas-mês.

4.1.1. Software 01

Este software é uma aplicação de Comércio Eletrônico para uma loja de materiais de construção. Os desenvolvedores usaram PHP, HTML, Java Script e MySQL. O tempo de desenvolvimento foi de 03 meses e o esforço de 06 pessoas-mês.

4.1.1.1. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO LINHAS DE CÓDIGO

A contagem de Linhas de Código foi realizada conforme detalhado no primeiro parágrafo da seção 4.1 deste trabalho, resultando para este software em 2.966 LOC (ou 2,966 KLOC).

A produtividade real foi calculada usando a equação (18):

$$\text{Produtividade} = S / E$$
$$\text{Produtividade} = 2{,}966 / 6$$
$$\text{Produtividade} = 0{,}494 \text{ KLOC/pessoas-mês}$$

A seguir serão mostradas as estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo:

a) Estimativa do Esforço

$$E = S \times fpl$$
$$E = 2{,}966 \times 3{,}6$$
$$E = 10{,}678 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S = E / fpl$$
$$S = 6 / 3{,}6$$
$$S = 1{,}667 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = S \times fpl / ted \ ^{7}$$
$$T = 2{,}966 \times 3{,}6 / 2$$
$$T = 5{,}339 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$
$$\text{Produtividade} = 1{,}667 / 10{,}678$$
$$\text{Produtividade} = 0{,}156 \text{ KLOC/pessoas-mês}$$

[7] Tamanho da Equipe de Desenvolvimento

4.1.1.2. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO DE ESTIMATIVA DE PUTNAM

Como este modelo utiliza unidades de medida diferentes, após os cálculos os valores foram convertidos para unidades padrões (pessoas-mês, KLOC e meses). As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E_{pa} = \frac{S_{loc}^{\ 3}}{C^3 \times t_d^{\ 4}}$$

$$E_{pa} = \frac{2.966^{\ 3}}{2000^3 \times 0{,}25^4}$$

$$E_{pa} = 834{,}956 \text{ pessoas-ano } [8]$$

$$E = 10.019{,}468 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S_{loc} = C \times E_{pa}^{\ 1/3} \times t_d^{\ 4/3}$$

$$S_{loc} = 2000 \times 0{,}5^{1/3} \times 0{,}25^{4/3}$$

$$S_{loc} = 250{,}00 \text{ LOC } [9]$$

$$S = 0{,}25 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = (S / C \times E^{1/3})^{3/4}$$

$$T = (2.966 / 2000 \times 0{,}5^{1/3})^{3/4}$$

$$T = 1{,}598 \text{ anos } [10]$$

$$T = 19{,}178 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 0{,}25 / 10.019{,}468$$

$$\text{Produtividade} = 0{,}000025 \text{ KLOC/pessoas-mês}$$

---

[8] A conversão de pessoas-ano para pessoas-mês é feita pela multiplicação por 12.

[9] A conversão de LOC para KLOC é feita pela divisão por 1000.

[10] A conversão de ano para meses é feita pela multiplicação por 12.

4.1.1.3. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO COCOMO BÁSICO

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E = a \times S^{b}$$
$$E = 2,4 \times 2,966^{1,05}$$
$$E = 7,516 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S = (E / a)^{1/b}$$
$$S = (6 / 2,4)^{1/1,05}$$
$$S = 2,393 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = c \times E^{d}$$
$$T = 2,5 \times 6^{0,38}$$
$$T = 4,939 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$
$$\text{Produtividade} = 2,393 / 7,516$$
$$\text{Produtividade} = 0,318 \text{ KLOC/pessoas-mês}$$

4.1.1.4. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE FUNÇÃO

Inicialmente foram calculados os PFB-Pontos de Função Brutos e o NI-Nível de Influência:

Tipo da Função	Complexidade Funcional	Peso da Complexidade	Qtde	Subtotal de Pontos
Arquivo Lógico Interno	Simples	7 x	3	21
	Média	10 x	0	0
	Complexa	15 x	0	0
Arquivo de Interface Externa	Simples	5 x	0	0
	Média	7 x	0	0
	Complexa	10 x	0	0
Entrada Externa	Simples	3 x	11	33
	Média	4 x	2	8
	Complexa	6 x	0	0
Saída Externa	Simples	4 x	2	8
	Média	5 x	2	10
	Complexa	7 x	1	7
Consulta Externa	Simples	3 x	2	6
	Média	4 x	1	4
	Complexa	6 x	1	6
Total de Pontos de Função Brutos				103

Características Gerais do Sistema		Nível de Influência
1.	Comunicação de dados	0
2.	Funções distribuídas	0
3.	Performance	3
4.	Configuração do equipamento	2
5.	Volume de transações	1
6.	Entrada de dados on-line	5
7.	Interface com o usuário	1
8.	Atualização on-line	2
9.	Processamento complexo	1
10.	Reusabilidade	0
11.	Facilidade de implantação	0
12.	Facilidade operacional	0
13.	Múltiplos locais	0
14.	Facilidade de mudanças (flexibilidade)	1
Total do Nível de Influência (NI)		16

Posteriormente foi calculado o FA-Fator de Ajuste:

$$FA = (NI \times 0,01) + 0,65$$

$$FA = (16 \times 0,01) + 0,65$$

$$FA = 0,81$$

E, finalmente foram calculados os PFA-Pontos de Função Ajustados:

$$PFA = PFB \times FA$$

PFA = 103 x 0,81

PFA = 83,43

A produtividade real do software, em PFA/pessoas-mês foi então calculada:

Produtividade = Tamanho / Esforço

Produtividade = 83,43 / 6

Produtividade = 13,905 PFA/pessoas-mês

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Tamanho em KLOC

$S = S_{pfa} \times LOC_{pfa}$ [11]

S = 83,43 x 14

S = 1.168,02 LOC [9]

S = 1,168 KLOC

b) Estimativa do Tamanho em PFA

$S = S_{loc} / LOC_{pfa}$ [11]

S = 2.966 / 14

S = 211,857 PFA

c) Estimativa de Produtividade

Produtividade = S / E

Produtividade = 211,857 / 6

Produtividade = 35,31 PFA/pessoas-mês

d) Estimativa do Esforço

$E = S_{pfa}$ / Produtividade

E = 83,43 / 35,31

E = 2,363 pessoas-mês

[11] S_{pfa} é o tamanho em Pontos de Função Ajustados e LOC_{pfa} foi retirado da Tabela 19.

e) Estimativa do Tempo

$$T = E / ted^{7}$$

$$T = 2{,}363 / 2$$

$$T = 1{,}181 \text{ meses}$$

4.1.1.5. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE PARTICULARIDADE

Como este software não apresentou nenhum algoritmo complexo, as estimativas ficam idênticas às estimativas feitas para Pontos de Função (na Seção 4.1.1.4).

4.1.1.6. ANÁLISE DAS MEDIÇÕES/ESTIMATIVAS PARA O SOFTWARE 01

Os gráficos mostrados nas Figuras 6, 7 e 8 apresentam os comparativos entre os valores reais e estimados para o Software 01.

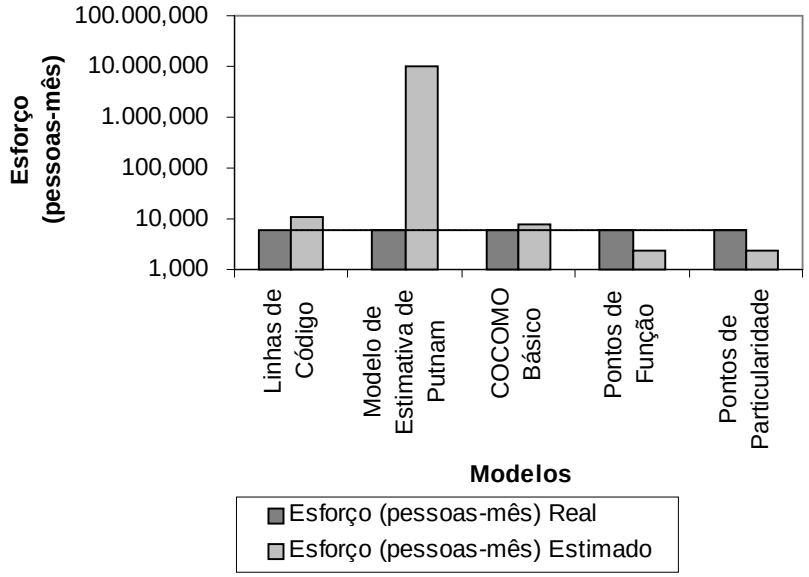

Figura 6. Esforço Real x Estimado para o Software 01

A Figura 6 mostra claramente a não adequação do Modelo de Estimativa de Putnam e a pouca adequação de LOC para estimar o esforço para este software.

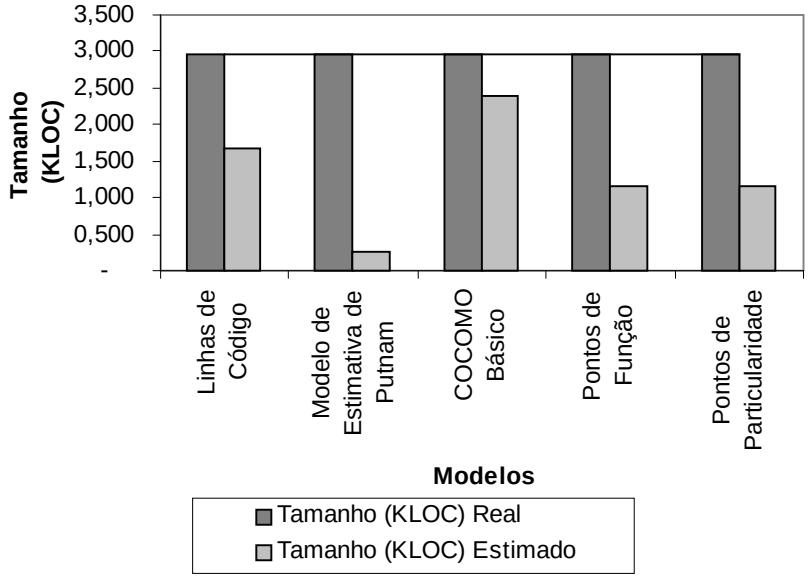

Figura 7. Tamanho Real x Estimado para o Software 01

Conforme pode ser observado na Figura 7 apenas o Modelo COCOMO Básico mostrou-se indicado para estimar o tamanho deste software.

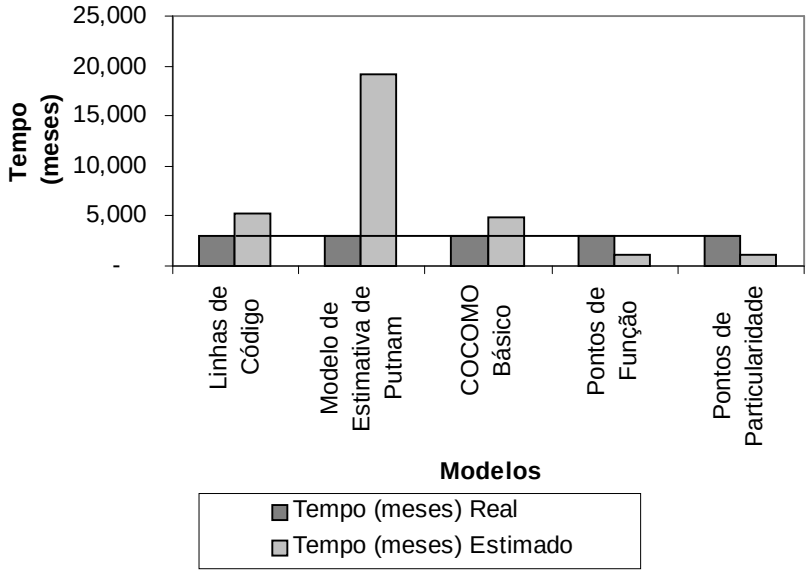

Figura 8. Tempo Real x Estimado para o Software 01

A Figura 8 mostra que, apenas o Modelo de Estimativa de Putnam, não é adequado para estimar o tempo de desenvolvimento deste software.

4.1.2. Software 02

Este software é uma aplicação de Comércio Eletrônico para uma loja de roupas jovens. Os pedidos ficam armazenados no provedor até que seja feito o *download* dos mesmos por meio deste software. O desenvolvedor usou PHP, HTML, Visual Basic e MySQL. O tempo de desenvolvimento foi de 02 meses e o esforço de 02 pessoas-mês.

4.1.2.1. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO LINHAS DE CÓDIGO

A contagem de Linhas de Código foi realizada conforme detalhado no primeiro parágrafo da seção 4.1 deste trabalho, resultando para este software em 1.451 LOC (ou 1,451 KLOC).

A produtividade real foi calculada usando a equação (18):

$$\text{Produtividade} = S / E$$
$$\text{Produtividade} = 1{,}451 / 2$$
$$\text{Produtividade} = 0{,}726 \text{ KLOC/pessoas-mês}$$

A seguir serão mostradas as estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo:

a) Estimativa do Esforço

$$E = S \times fpl$$
$$E = 1{,}451 \times 3{,}6$$
$$E = 5{,}224 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S = E / fpl$$
$$S = 2 / 3{,}6$$
$$S = 0{,}556 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = S \times fpl / ted^{7}$$
$$T = 1{,}451 \times 3{,}6 / 1$$
$$T = 5{,}224 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 0{,}556 / 5{,}224$$

$$\text{Produtividade} = 0{,}106 \text{ KLOC/pessoas-mês}$$

4.1.2.2. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO DE ESTIMATIVA DE PUTNAM

Como este modelo utiliza unidades de medida diferentes, após os cálculos os valores foram convertidos para unidades padrões (pessoas-mês, KLOC e meses). As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E_{pa} = \frac{S_{loc}{}^{3}}{C^{3} \times t_d{}^{4}}$$

$$E_{pa} = \frac{1.451{}^{3}}{2000^{3} \times 0{,}167^{4}}$$

$$E_{pa} = 494{,}90 \text{ pessoas-ano}^{8}$$

$$E = 5.938{,}797 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S_{loc} = C \times E_{pa}{}^{1/3} \times t_d{}^{4/3}$$

$$S_{loc} = 2000 \times 0{,}167^{1/3} \times 0{,}167^{4/3}$$

$$S_{loc} = 101{,}00 \text{ LOC}^{9}$$

$$S = 0{,}101 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = (S / C \times E^{1/3})^{3/4}$$

$$T = (1.451 / 2000 \times 0{,}167^{1/3})^{3/4}$$

$$T = 1{,}23 \text{ anos}^{10}$$

$$T = 14{,}764 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 0{,}101 / 5.938{,}797$$

$$\text{Produtividade} = 0{,}000017 \text{ KLOC/pessoas-mês}$$

4.1.2.3. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO COCOMO BÁSICO

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E = a \times S^b$$

$$E = 2,4 \times 1,451^{1,05}$$

$$E = 3,548 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S = (E / a)^{1/b}$$

$$S = (2 / 2,4)^{1/1,05}$$

$$S = 0,841 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = c \times E^d$$

$$T = 2,5 \times 2^{0,38}$$

$$T = 3,253 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 0,841 / 3,548$$

$$\text{Produtividade} = 0,237 \text{ KLOC/pessoas-mês}$$

4.1.2.4. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE FUNÇÃO

Inicialmente foram calculados os PFB-Pontos de Função Brutos e o NI-Nível de Influência:

Tipo da Função	Complexidade Funcional	Peso da Complexidade	Qtde	Subtotal de Pontos
Arquivo Lógico Interno	Simples	7 x	3	21
	Média	10 x	0	0
	Complexa	15 x	0	0
Arquivo de Interface Externa	Simples	5 x	3	15
	Média	7 x	0	0
	Complexa	10 x	0	0
Entrada Externa	Simples	3 x	11	33
	Média	4 x	2	8
	Complexa	6 x	0	0
Saída Externa	Simples	4 x	2	8
	Média	5 x	2	10
	Complexa	7 x	1	7
Consulta Externa	Simples	3 x	2	6
	Média	4 x	1	4
	Complexa	6 x	1	6
Total de Pontos de Função Brutos				118

Características Gerais do Sistema		Nível de Influência
1.	Comunicação de dados	0
2.	Funções distribuídas	0
3.	Performance	3
4.	Configuração do equipamento	2
5.	Volume de transações	1
6.	Entrada de dados on-line	5
7.	Interface com o usuário	1
8.	Atualização on-line	2
9.	Processamento complexo	1
10.	Reusabilidade	0
11.	Facilidade de implantação	0
12.	Facilidade operacional	0
13.	Múltiplos locais	0
14.	Facilidade de mudanças (flexibilidade)	1
Total do Nível de Influência (NI)		16

Posteriormente foi calculado o FA-Fator de Ajuste:

$$FA = (NI \times 0,01) + 0,65$$

$$FA = (16 \times 0,01) + 0,65$$

$$FA = 0,81$$

E, finalmente foram calculados os PFA-Pontos de Função Ajustados:

$$PFA = PFB \times FA$$

$$PFA = 118 \times 0{,}81$$
$$PFA = 95{,}58$$

A produtividade real do software, em PFA/pessoas-mês foi então calculada:

$$Produtividade = Tamanho / Esforço$$
$$Produtividade = 95{,}58 / 2$$
$$Produtividade = 47{,}79 \text{ PFA/pessoas-mês}$$

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Tamanho em KLOC

$$S_1 = S_{pfa} \times LOC_{pfa}^{11}$$
$$S_1 = 42{,}09 \times 14$$
$$S_1 = 589{,}26 \text{ LOC}$$

$$S_2 = S_{pfa} \times LOC_{pfa}^{11}$$
$$S_2 = 64{,}11 \times 24$$
$$S_2 = 1.538{,}64 \text{ LOC}$$

$$S = S_1 + S_2$$
$$S = 589{,}26 + 1.538{,}64$$
$$S = 2.127{,}90^{9}$$
$$S = 2{,}128 \text{ KLOC}$$

Onde:

S_1: Tamanho estimado do software (em LOC) desenvolvido em HTML;

S_2: Tamanho estimado do software (em LOC) desenvolvido em Visual Basic.

b) Estimativa do Tamanho em PFA

$$S_1 = S_{loc} / LOC_{pfa}^{11}$$
$$S_1 = 575 / 14$$
$$S_1 = 41{,}071 \text{ PFA}$$

$$S_2 = S_{loc} / LOC_{pfa}$$ [11]

$$S_2 = 876 / 24$$

$$S_2 = 36,50 \text{ PFA}$$

$$S = S_1 + S_2$$

$$S = 41,071 + 36,50$$

$$S = 77,571 \text{ PFA}$$

Onde:

S_1: Tamanho estimado do software (em PFA) desenvolvido em HTML;

S_2: Tamanho estimado do software (em PFA) desenvolvido em Visual Basic.

c) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 77,571 / 2$$

$$\text{Produtividade} = 38,786 \text{ PFA/pessoas-mês}$$

d) Estimativa do Esforço

$$E = S_{pfa} / \text{Produtividade}$$

$$E = 95,58 / 38,786$$

$$E = 2,464 \text{ pessoas-mês}$$

e) Estimativa do Tempo

$$T = E / ted$$ [7]

$$T = 2,464 / 1$$

$$T = 2,464 \text{ meses}$$

4.1.2.5. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE PARTICULARIDADE

Como este software não apresentou nenhum algoritmo complexo, as estimativas ficam idênticas às estimativas feitas para Pontos de Função (na Seção 4.1.2.4).

4.1.2.6. ANÁLISE DAS MEDIÇÕES/ESTIMATIVAS PARA O SOFTWARE 02

Os gráficos mostrados nas Figuras 9, 10 e 11 apresentam os comparativos entre os valores reais e estimados para o Software 02.

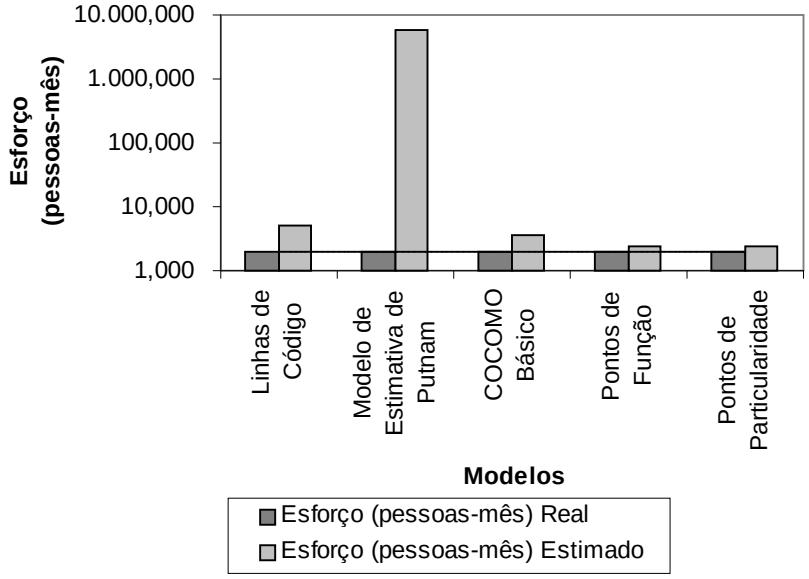

Figura 9. Esforço Real x Estimado para o Software 02

A Figura 9 mostra claramente a não adequação do Modelo de Estimativa de Putnam e a pouca adequação de LOC para estimar o esforço para este software.

Conforme pode ser observado na Figura 10, os Modelos COCOMO Básico, Pontos de Função e Pontos de Particularidade mostraram-se pouco indicados para estimar o tamanho deste software. Os demais modelos mostraram-se totalmente inadequados para esta estimativa.

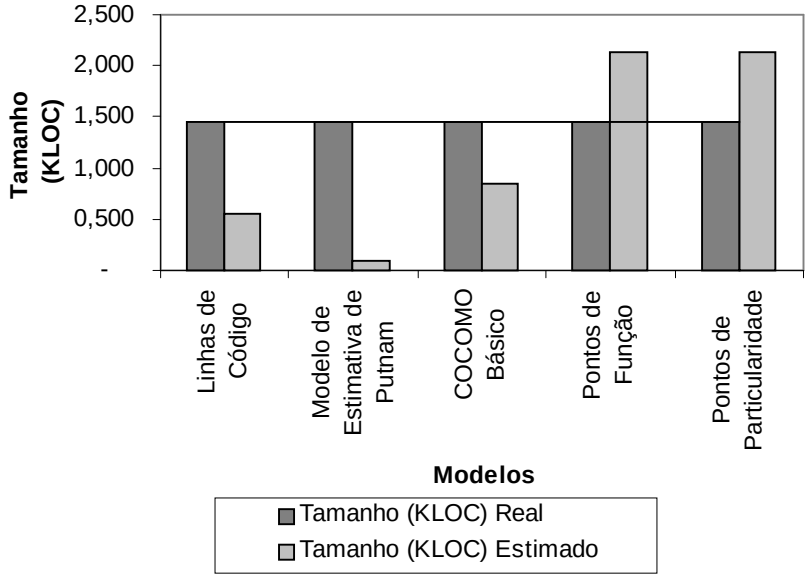

Figura 10. Tamanho Real x Estimado para o Software 02

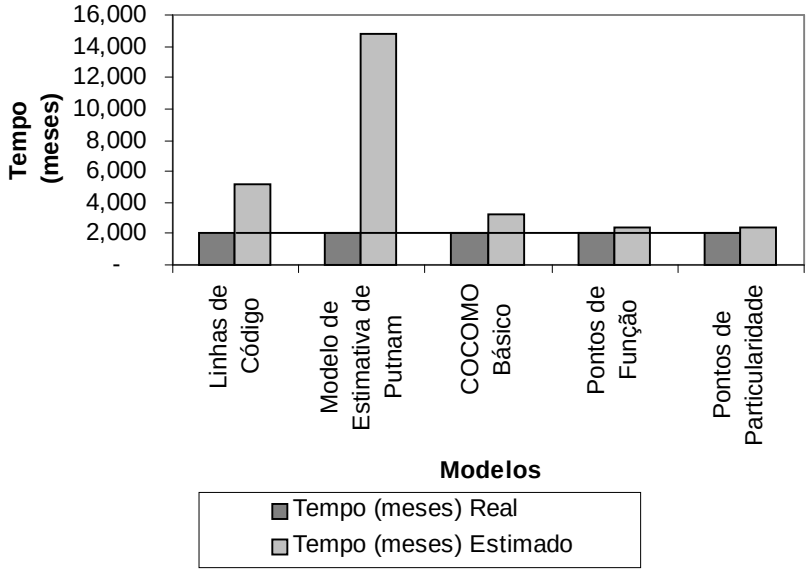

Figura 11. Tempo Real x Estimado para o Software 02

A Figura 11 mostra que apenas o Modelo de Estimativa de Putnam é inadequado para estimar o tempo de desenvolvimento deste software. A contagem de Linhas de Código mostra-se pouco adequada.

4.1.3. Software 03

Este software é uma aplicação de Inteligência Artificial, mais especificamente de Redes Neurais Artificiais, para estimar o valor do CUB-SC (Custo Unitário Básico da Construção Civil em Santa Catarina). O desenvolvedor usou Delphi com Paradox. O tempo de desenvolvimento foi de 06 meses e o esforço de 06 pessoas-mês.

4.1.3.1. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO LINHAS DE CÓDIGO

A contagem de Linhas de Código foi realizada conforme detalhado no primeiro parágrafo da seção 4.1 deste trabalho, resultando para este software em 544 LOC (ou 0,544 KLOC).

A produtividade real foi calculada usando a equação (18):

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 0{,}544 / 6$$

$$\text{Produtividade} = 0{,}091 \text{ KLOC/pessoas-mês}$$

A seguir serão mostradas as estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo:

a) Estimativa do Esforço

$$E = S \times fpl$$

$$E = 0,544 \times 2,94$$

$$E = 1,599 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S = E / fpl$$

$$S = 6 / 2,94$$

$$S = 2,041 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = S \times fpl / ted \quad [7]$$

$$T = 0,544 \times 2,94 / 1$$

$$T = 1,599 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 2,041 / 1,599$$

$$\text{Produtividade} = 1,276 \text{ KLOC/pessoas-mês}$$

4.1.3.2. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO DE ESTIMATIVA DE PUTNAM

Como este modelo utiliza unidades de medida diferentes, após os cálculos os valores foram convertidos para unidades padrões (pessoas-mês, KLOC e meses). As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E_{pa} = \frac{S_{loc}{}^3}{C^3 \times t_d{}^4}$$

$$E_{pa} = \frac{544^3}{2000^3 \times 0,5^4}$$

$$E_{pa} = 0,322 \text{ pessoas-ano} \quad [8]$$

$$E = 3,864 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S_{loc} = C \times E_{pa}^{1/3} \times t_d^{4/3}$$

$$S_{loc} = 2000 \times 0{,}5^{1/3} \times 0{,}5^{4/3}$$

$$S_{loc} = 630{,}00 \text{ LOC}$$ [9]

$$S = 0{,}63 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = (S / C \times E^{1/3})^{3/4}$$

$$T = (544 / 2000 \times 0{,}5^{1/3})^{3/4}$$

$$T = 0{,}448 \text{ anos}$$ [10]

$$T = 5{,}375 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 0{,}63 / 3{,}864$$

$$\text{Produtividade} = 0{,}163044 \text{ KLOC/pessoas-mês}$$

4.1.3.3. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO COCOMO BÁSICO

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E = a \times S^b$$

$$E = 2{,}4 \times 0{,}544^{1{,}05}$$

$$E = 1{,}266 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S = (E / a)^{1/b}$$

$$S = (6 / 2{,}4)^{1/1{,}05}$$

$$S = 2{,}393 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = c \times E^d$$

$$T = 2{,}5 \times 6^{0{,}38}$$

$$T = 4{,}939 \text{ meses}$$

d) Estimativa de Produtividade

Produtividade = S / E

Produtividade = 2,393 / 1,266

Produtividade = 1,89 KLOC/pessoas-mês

4.1.3.4. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE FUNÇÃO

Inicialmente foram calculados os PFB-Pontos de Função Brutos e o NI-Nível de Influência:

Tipo da Função	Complexidade Funcional	Peso da Complexidade	Qtde	Subtotal de Pontos
Arquivo Lógico Interno	Simples	7 x	9	63
	Média	10 x	0	0
	Complexa	15 x	0	0
Arquivo de Interface Externa	Simples	5 x	0	0
	Média	7 x	0	0
	Complexa	10 x	0	0
Entrada Externa	Simples	3 x	3	9
	Média	4 x	0	0
	Complexa	6 x	0	0
Saída Externa	Simples	4 x	2	8
	Média	5 x	0	0
	Complexa	7 x	0	0
Consulta Externa	Simples	3 x	0	0
	Média	4 x	0	0
	Complexa	6 x	0	0
Total de Pontos de Função Brutos				80

	Características Gerais do Sistema	Nível de Influência
1.	Comunicação de dados	0
2.	Funções distribuídas	0
3.	Performance	3
4.	Configuração do equipamento	2
5.	Volume de transações	4
6.	Entrada de dados on-line	2
7.	Interface com o usuário	1
8.	Atualização on-line	3
9.	Processamento complexo	1
10.	Reusabilidade	1
11.	Facilidade de implantação	0
12.	Facilidade operacional	0
13.	Múltiplos locais	0
14.	Facilidade de mudanças (flexibilidade)	0
Total do Nível de Influência (NI)		17

Posteriormente foi calculado o FA-Fator de Ajuste:

$$FA = (NI \times 0,01) + 0,65$$

$$FA = (17 \times 0,01) + 0,65$$

$$FA = 0,82$$

E, finalmente foram calculados os PFA-Pontos de Função Ajustados:

$$PFA = PFB \times FA$$

$$PFA = 80 \times 0,82$$

$$PFA = 65,60$$

A produtividade real do software, em PFA/pessoas-mês foi então calculada:

$$Produtividade = Tamanho / Esforço$$

$$Produtividade = 65,60 / 6$$

$$Produtividade = 10,933 \text{ PFA/pessoas-mês}$$

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Tamanho em KLOC

$$S = S_{pfa} \times LOC_{pfa}$$ [11]

$$S = 65,60 \times 18$$

$$S = 1.180,80 \text{ LOC}$$ [9]

$$S = 1,181 \text{ KLOC}$$

b) Estimativa do Tamanho em PFA

$$S = S_{loc} / LOC_{pfa}$$ [11]

$$S = 544 / 18$$

$$S = 30,222 \text{ PFA}$$

c) Estimativa de Produtividade

$$Produtividade = S / E$$

$$Produtividade = 30,222 / 6$$

$$Produtividade = 5,037 \text{ PFA/pessoas-mês}$$

d) Estimativa do Esforço

$$E = S_{pfa} / \text{Produtividade}$$

$$E = 65,60 / 5,037$$

$$E = 13,024 \text{ pessoas-mês}$$

e) Estimativa do Tempo

$$T = E / \text{ted}^{7}$$

$$T = 13,024 / 1$$

$$T = 13,024 \text{ meses}$$

4.1.3.5. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE PARTICULARIDADE

Inicialmente foram calculados os PPB-Pontos de Particularidade Brutos e o NI-Nível de Influência:

Tipo da Função	Complexidade Funcional	Peso da Complexidade	Qtde	Subtotal de Pontos
Algoritmos Complexos	Simples	1 x	2	2
	Média	5 x	1	5
	Complexa	10 x	0	0
Arquivo Lógico Interno	Simples	7 x	9	63
	Média	10 x	0	0
	Complexa	15 x	0	0
Arquivo de Interface Externa	Simples	5 x	0	0
	Média	7 x	0	0
	Complexa	10 x	0	0
Entrada Externa	Simples	3 x	3	9
	Média	4 x	0	0
	Complexa	6 x	0	0
Saída Externa	Simples	4 x	2	8
	Média	5 x	0	0
	Complexa	7 x	0	0
Consulta Externa	Simples	3 x	0	0
	Média	4 x	0	0
	Complexa	6 x	0	0
Total de Pontos de Particularidade Brutos				87

Características Gerais do Sistema		Nível de Influência
1.	Comunicação de dados	0
2.	Funções distribuídas	0
3.	Performance	3
4.	Configuração do equipamento	2
5.	Volume de transações	4
6.	Entrada de dados on-line	2
7.	Interface com o usuário	1
8.	Atualização on-line	3
9.	Processamento complexo	1
10.	Reusabilidade	1
11.	Facilidade de implantação	0
12.	Facilidade operacional	0
13.	Múltiplos locais	0
14.	Facilidade de mudanças (flexibilidade)	0
Total do Nível de Influência (NI)		17

Posteriormente foi calculado o FA-Fator de Ajuste:

$$FA = (NI \times 0,01) + 0,65$$

$$FA = (17 \times 0,01) + 0,65$$

$$FA = 0,82$$

E, finalmente foram calculados os PPA-Pontos de Particularidade Ajustados:

$$PPA = PPB \times FA$$

$$PPA = 87 \times 0,82$$

$$PPA = 71,34$$

A produtividade real do software, em PPA/pessoas-mês foi então calculada:

$$Produtividade = Tamanho / Esforço$$

$$Produtividade = 71,34 / 6$$

$$Produtividade = 11,89 \text{ PPA/pessoas-mês}$$

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Tamanho em KLOC

$$S = S_{ppa} \times LOC_{ppa} \quad [11]$$

$$S = 71{,}34 \times 18$$

$$S = 1.284{,}12 \text{ LOC} \quad [9]$$

$$S = 1{,}284 \text{ KLOC}$$

b) Estimativa do Tamanho em PPA

$$S = S_{loc} / LOC_{ppa} \quad [11]$$

$$S = 544 / 18$$

$$S = 30{,}222 \text{ PPA}$$

c) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 30{,}222 / 6$$

$$\text{Produtividade} = 5{,}037 \text{ PPA/pessoas-mês}$$

d) Estimativa do Esforço

$$E = S_{ppa} / \text{Produtividade}$$

$$E = 71{,}34 / 5{,}037$$

$$E = 14{,}163 \text{ pessoas-mês}$$

e) Estimativa do Tempo

$$T = E / \text{ted} \quad [7]$$

$$T = 14{,}163 / 1$$

$$T = 14{,}163 \text{ meses}$$

4.1.3.6. ANÁLISE DAS MEDIÇÕES/ESTIMATIVAS PARA O SOFTWARE 03

Os gráficos mostrados nas Figuras 12, 13 e 14 apresentam os comparativos entre os valores reais e estimados para o Software 03.

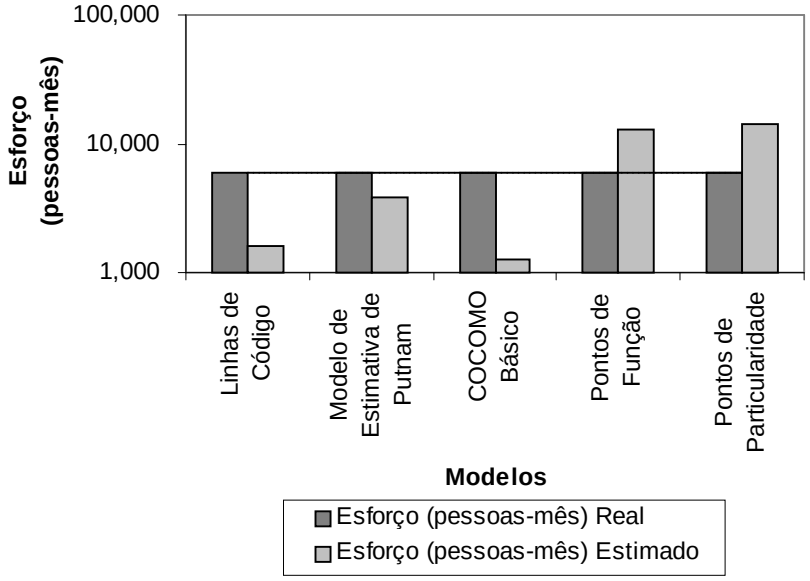

Figura 12. Esforço Real x Estimado para o Software 03

A Figura 12 mostra claramente a adequação do Modelo de Estimativa de Putnam para estimar o esforço para este software.

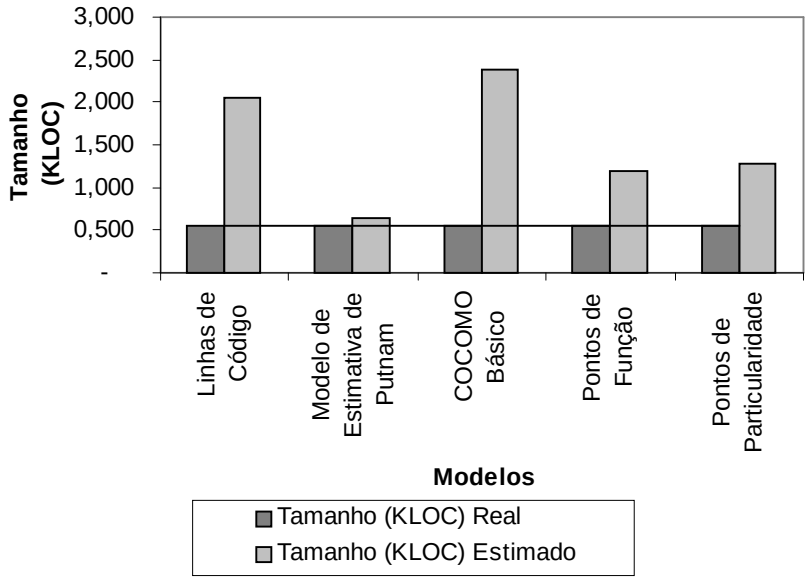

Figura 13. Tamanho Real x Estimado para o Software 03

Conforme pode ser observado na Figura 13, o Modelo de Estimativa de Putnam mostrou-se totalmente adequado para estimar o tamanho deste software.

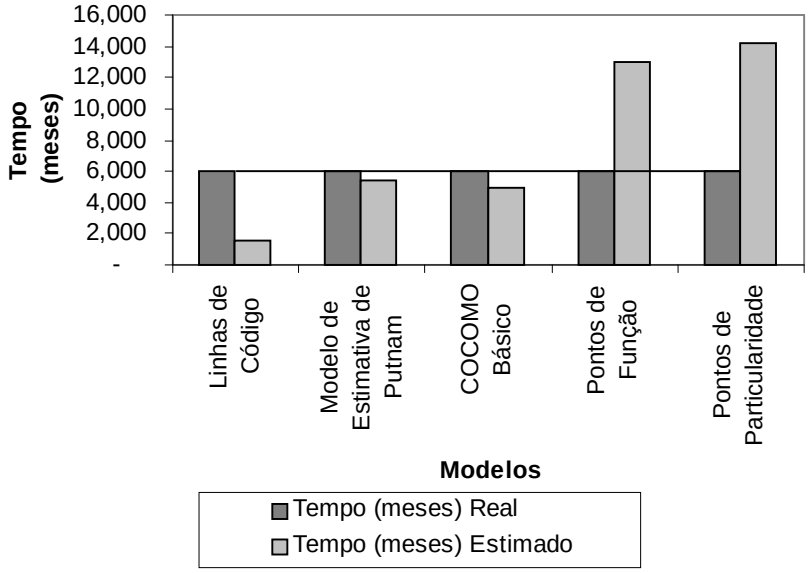

Figura 14. Tempo Real x Estimado para o Software 03

A Figura 14 mostra que os Modelos de Estimativa de Putnam e COCOMO Básico são indicados para estimar o tempo de desenvolvimento deste software.

4.1.4. Software 04

Este software é uma aplicação de Inteligência Artificial para otimizar o padrão de corte dos painéis utilizados na montagem de paredes divisórias, calcular o orçamento de projetos de paredes deste tipo e imprimir os desenhos correspondentes aos projetos de corte dos painéis e de construção das paredes. Os desenvolvedores usaram Delphi com Paradox e arquivos binários. O tempo de desenvolvimento foi de 24 meses e o esforço de 48 pessoas-mês.

4.1.4.1. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO LINHAS DE CÓDIGO

A contagem de Linhas de Código foi realizada conforme detalhado no primeiro parágrafo da seção 4.1 deste trabalho, resultando para este software em 5.816 LOC (ou 5,816 KLOC).

A produtividade real foi calculada usando a equação (18):

Produtividade = S / E

Produtividade = 5,816 / 48

Produtividade = 0,121 KLOC/pessoas-mês

A seguir serão mostradas as estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo:

a) Estimativa do Esforço

E = S x fpl

E = 5,816 x 2,94

E = 17,099 pessoas-mês

b) Estimativa do Tamanho

S = E / fpl

S = 48 / 2,94

S = 16,327 KLOC

c) Estimativa do Tempo

T = S x fpl / ted [7]

T = 5,816 x 2,94 / 2

T = 8,55 meses

d) Estimativa de Produtividade

Produtividade = S / E

Produtividade = 16,327 / 17,099

Produtividade = 0,955 KLOC/pessoas-mês

4.1.4.2. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO DE ESTIMATIVA DE PUTNAM

Como este modelo utiliza unidades de medida diferentes, após os cálculos os valores foram convertidos para unidades padrões (pessoas-mês, KLOC e meses). As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E_{pa} = \frac{S_{loc}{}^{3}}{C^{3} \times t_{d}{}^{4}}$$

$$E_{pa} = \frac{5.816^{3}}{2000^{3} \times 2^{4}}$$

$$E_{pa} = 1{,}537 \text{ pessoas-ano } [8]$$

$$E = 18{,}444 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S_{loc} = C \times E_{pa}{}^{1/3} \times t_{d}{}^{4/3}$$

$$S_{loc} = 2000 \times 4^{1/3} \times 2^{4/3}$$

$$S_{loc} = 8.000 \text{ LOC } [9]$$

$$S = 8{,}00 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = (S / C \times E^{1/3})^{3/4}$$

$$T = (5.816 / 2000 \times 4^{1/3})^{3/4}$$

$$T = 1{,}575 \text{ anos } [10]$$

$$T = 18{,}896 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 8{,}00 / 18{,}444$$

$$\text{Produtividade} = 0{,}433756 \text{ KLOC/pessoas-mês}$$

4.1.4.3. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO COCOMO BÁSICO

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E = a \times S^{b}$$

$$E = 3 \times 5{,}816^{1{,}12}$$

$$E = 21{,}553 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S = (E / a)^{1/b}$$

$$S = (48 / 3)^{1/1,12}$$

$$S = 11,888 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = c \times E^{d}$$

$$T = 2,5 \times 48^{0,35}$$

$$T = 9,691 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 11,888 / 21,553$$

$$\text{Produtividade} = 0,552 \text{ KLOC/pessoas-mês}$$

4.1.4.4. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE FUNÇÃO

Inicialmente foram calculados os PFB-Pontos de Função Brutos e o NI-Nível de Influência:

Tipo da Função	Complexidade Funcional	Peso da Complexidade	Qtde	Subtotal de Pontos
Arquivo Lógico Interno	Simples	7 x	1	7
	Média	10 x	2	20
	Complexa	15 x	1	15
Arquivo de Interface Externa	Simples	5 x	0	0
	Média	7 x	0	0
	Complexa	10 x	0	0
Entrada Externa	Simples	3 x	24	72
	Média	4 x	5	20
	Complexa	6 x	0	0
Saída Externa	Simples	4 x	4	16
	Média	5 x	0	0
	Complexa	7 x	0	0
Consulta Externa	Simples	3 x	0	0
	Média	4 x	0	0
	Complexa	6 x	0	0
Total de Pontos de Função Brutos				150

Características Gerais do Sistema		Nível de Influência
1.	Comunicação de dados	0
2.	Funções distribuídas	0
3.	Performance	4
4.	Configuração do equipamento	2
5.	Volume de transações	4
6.	Entrada de dados on-line	2
7.	Interface com o usuário	2
8.	Atualização on-line	2
9.	Processamento complexo	2
10.	Reusabilidade	2
11.	Facilidade de implantação	1
12.	Facilidade operacional	1
13.	Múltiplos locais	0
14.	Facilidade de mudanças (flexibilidade)	1
Total do Nível de Influência (NI)		23

Posteriormente foi calculado o FA-Fator de Ajuste:

$$FA = (NI \times 0,01) + 0,65$$

$$FA = (23 \times 0,01) + 0,65$$

$$FA = 0,88$$

E, finalmente foram calculados os PFA-Pontos de Função Ajustados:

$$PFA = PFB \times FA$$

$$PFA = 150 \times 0,88$$

$$PFA = 132,00$$

A produtividade real do software, em PFA/pessoas-mês foi então calculada:

$$Produtividade = Tamanho / Esforço$$

$$Produtividade = 132,00 / 48$$

$$Produtividade = 2,75 \text{ PFA/pessoas-mês}$$

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Tamanho em KLOC

$$S = S_{pfa} \times LOC_{pfa}$$ [11]

$$S = 132,00 \times 18$$

$$S = 2.376{,}00 \text{ LOC} \quad [9]$$

$$S = 2{,}376 \text{ KLOC}$$

b) Estimativa do Tamanho em PFA

$$S = S_{loc} / \text{ LOC}_{pfa} \quad [11]$$

$$S = 5.816 / 18$$

$$S = 323{,}111 \text{ PFA}$$

c) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 323{,}111 / 48$$

$$\text{Produtividade} = 6{,}731 \text{ PFA/pessoas-mês}$$

d) Estimativa do Esforço

$$E = S_{pfa} / \text{Produtividade}$$

$$E = 132{,}00 / 6{,}731$$

$$E = 19{,}609 \text{ pessoas-mês}$$

e) Estimativa do Tempo

$$T = E / \text{ted} \quad [7]$$

$$T = 19{,}609 / 2$$

$$T = 9{,}805 \text{ meses}$$

4.1.4.5. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE PARTICULARIDADE

Inicialmente foram calculados os PPB-Pontos de Particularidade Brutos e o NI-Nível de Influência:

Tipo da Função	Complexidade Funcional	Peso da Complexidade	Qtde	Subtotal de Pontos
Algoritmos Complexos	Simples	1 x	0	0
	Média	5 x	2	10
	Complexa	10 x	2	20
Arquivo Lógico Interno	Simples	7 x	1	7
	Média	10 x	2	20
	Complexa	15 x	1	15
Arquivo de Interface Externa	Simples	5 x	0	0
	Média	7 x	0	0
	Complexa	10 x	0	0
Entrada Externa	Simples	3 x	24	72
	Média	4 x	5	20
	Complexa	6 x	0	0
Saída Externa	Simples	4 x	4	16
	Média	5 x	0	0
	Complexa	7 x	0	0
Consulta Externa	Simples	3 x	0	0
	Média	4 x	0	0
	Complexa	6 x	0	0
Total de Pontos de Particularidade Brutos				180

	Características Gerais do Sistema	Nível de Influência
1.	Comunicação de dados	0
2.	Funções distribuídas	0
3.	Performance	4
4.	Configuração do equipamento	2
5.	Volume de transações	4
6.	Entrada de dados on-line	2
7.	Interface com o usuário	2
8.	Atualização on-line	2
9.	Processamento complexo	2
10.	Reusabilidade	2
11.	Facilidade de implantação	1
12.	Facilidade operacional	1
13.	Múltiplos locais	0
14.	Facilidade de mudanças (flexibilidade)	1
Total do Nível de Influência (NI)		23

Posteriormente foi calculado o FA-Fator de Ajuste:

$$FA = (NI \times 0,01) + 0,65$$

$$FA = (23 \times 0,01) + 0,65$$

$$FA = 0,88$$

E, finalmente foram calculados os PPA-Pontos de Particularidade Ajustados:

$$PPA = PPB \times FA$$

$$PPA = 180 \times 0,88$$

$$PPA = 158,40$$

A produtividade real do software, em PPA/pessoas-mês foi então calculada:

$$Produtividade = Tamanho / Esforço$$

$$Produtividade = 158,40 / 48$$

$$Produtividade = 3,30 \text{ PPA/pessoas-mês}$$

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Tamanho em KLOC

$$S = S_{ppa} \times LOC_{ppa}^{11}$$

$$S = 158,40 \times 18$$

$$S = 2.851,20 \text{ LOC}^{9}$$

$$S = 2,851 \text{ KLOC}$$

b) Estimativa do Tamanho em PPA

$$S = S_{loc} / LOC_{ppa}^{11}$$

$$S = 5.816 / 18$$

$$S = 323,111 \text{ PPA}$$

c) Estimativa de Produtividade

$$Produtividade = S / E$$

$$Produtividade = 323,111 / 48$$

$$Produtividade = 6,731 \text{ PPA/pessoas-mês}$$

d) Estimativa do Esforço

$$E = S_{ppa} / Produtividade$$

$$E = 158,40 / 6,731$$

$$E = 23,531 \text{ pessoas-mês}$$

e) Estimativa do Tempo

$$T = E / ted \quad [7]$$

$$T = 23{,}531 / 2$$

$$T = 11{,}766 \text{ meses}$$

4.1.4.6. ANÁLISE DAS MEDIÇÕES/ESTIMATIVAS PARA O SOFTWARE 04

Os gráficos mostrados nas Figuras 15, 16 e 17 apresentam os comparativos entre os valores reais e estimados para o Software 04.

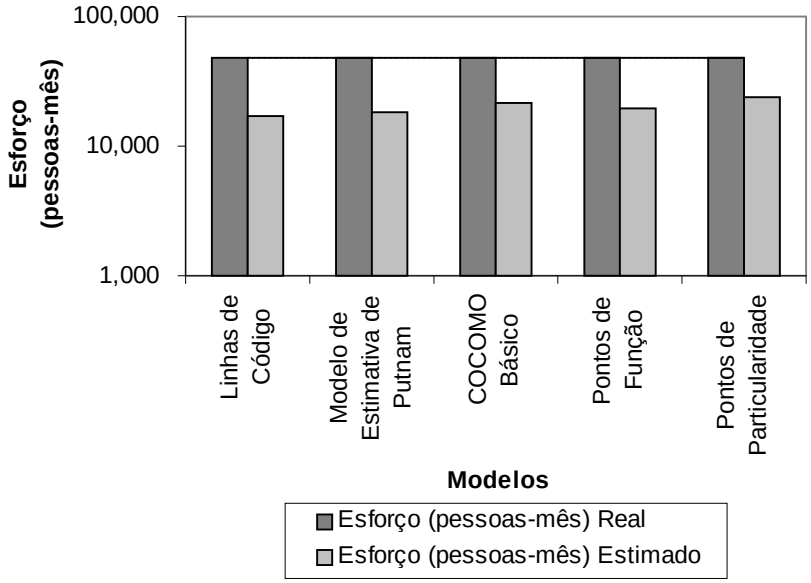

Figura 15. Esforço Real x Estimado para o Software 04

A Figura 15 mostra claramente a não adequação dos modelos para estimar o esforço para este software.

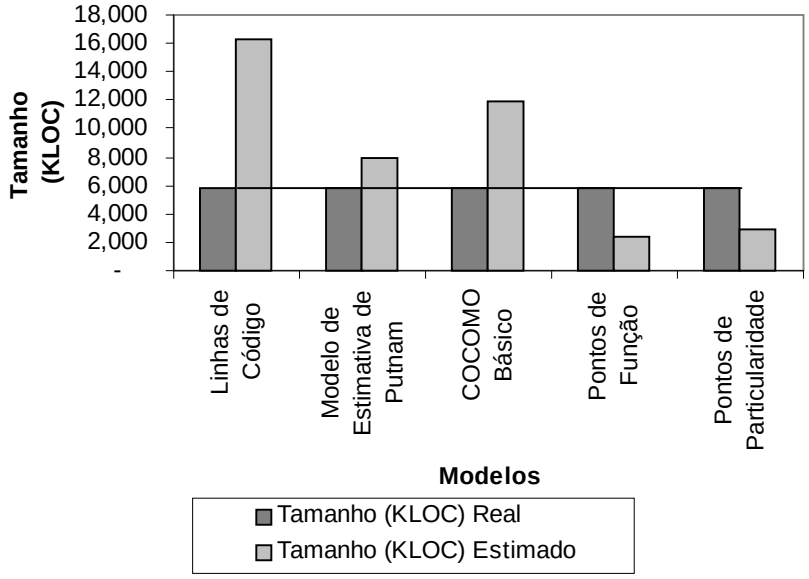

Figura 16. Tamanho Real x Estimado para o Software 04

Conforme pode ser observado na Figura 16, o Modelo de Estimativa de Putnam mostrou-se pouco adequado para estimar o tamanho deste software.

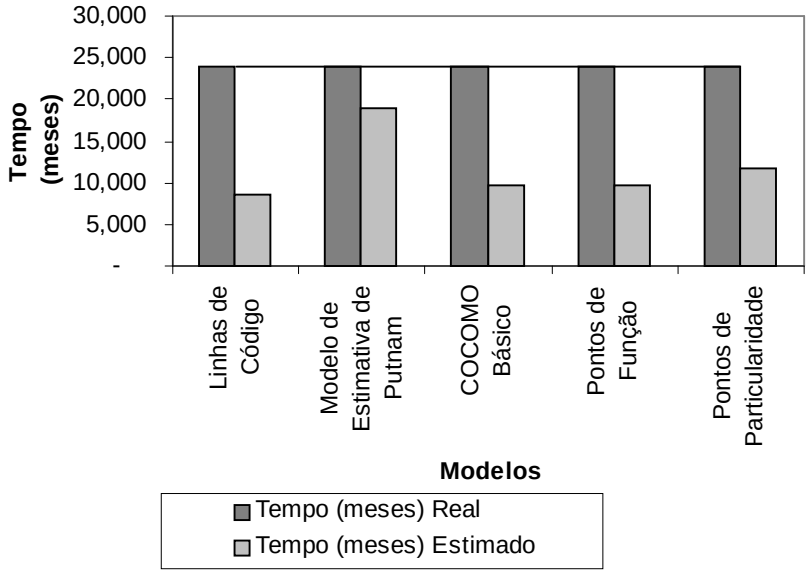

Figura 17. Tempo Real x Estimado para o Software 04

A Figura 17 mostra que o Modelo de Estimativa de Putnam é pouco indicado para estimar o tempo de desenvolvimento deste software.

4.1.5. Software 05

Este software é uma aplicação de Sistemas de Informação para automatizar as rotinas de controle financeiro, estoque e vendas de uma lancheria. O desenvolvedor usou um gerador de código conhecido como Clarion. O tempo de desenvolvimento foi de 03 meses e o esforço de 03 pessoas-mês.

4.1.5.1. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO LINHAS DE CÓDIGO

A contagem de Linhas de Código foi realizada conforme detalhado no primeiro parágrafo da seção 4.1 deste trabalho, resultando para este software em 1.203 LOC (ou 1,203 KLOC).

A produtividade real foi calculada usando a equação (18):

$$\text{Produtividade} = S / E$$
$$\text{Produtividade} = 1{,}203 / 3$$
$$\text{Produtividade} = 0{,}401 \text{ KLOC/pessoas-mês}$$

A seguir serão mostradas as estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo:

a) Estimativa do Esforço

$$E = S \times fpl$$
$$E = 1{,}203 \times 2{,}94$$
$$E = 3{,}537 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S = E / fpl$$
$$S = 3 / 2{,}94$$
$$S = 1{,}020 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = S \times fpl / ted^{7}$$
$$T = 1{,}203 \times 2{,}94 / 1$$
$$T = 3{,}537 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 1{,}02 / 3{,}537$$

$$\text{Produtividade} = 0{,}289 \text{ KLOC/pessoas-mês}$$

4.1.5.2. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO DE ESTIMATIVA DE PUTNAM

Como este modelo utiliza unidades de medida diferentes, após os cálculos os valores foram convertidos para unidades padrões (pessoas-mês, KLOC e meses). As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E_{pa} = \frac{S_{loc}{}^3}{C^3 \times t_d{}^4}$$

$$E_{pa} = \frac{1.203^3}{2000^3 \times 0{,}25^4}$$

$$E_{pa} = 55{,}712 \text{ pessoas-ano} \quad [8]$$

$$E = 668{,}541 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S_{loc} = C \times E_{pa}{}^{1/3} \times t_d{}^{4/3}$$

$$S_{loc} = 2000 \times 0{,}25^{1/3} \times 0{,}25^{4/3}$$

$$S_{loc} = 198{,}00 \text{ LOC} \quad [9]$$

$$S = 0{,}198 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = (S / C \times E^{1/3})^{3/4}$$

$$T = (1.203 / 2000 \times 0{,}25^{1/3})^{3/4}$$

$$T = 0{,}966 \text{ anos} \quad [10]$$

$$T = 11{,}591 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 0{,}198 / 668{,}541$$

$$\text{Produtividade} = 0{,}000297 \text{ KLOC/pessoas-mês}$$

4.1.5.3. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO COCOMO BÁSICO

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E = a \times S^b$$

$$E = 2,4 \times 1,203^{1,05}$$

$$E = 2,914 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S = (E / a)^{1/b}$$

$$S = (3 / 2,4)^{1/1,05}$$

$$S = 1,237 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = c \times E^d$$

$$T = 2,5 \times 3^{0,38}$$

$$T = 3,795 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 1,237 / 2,914$$

$$\text{Produtividade} = 0,424 \text{ KLOC/pessoas-mês}$$

4.1.5.4. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE FUNÇÃO

Inicialmente foram calculados os PFB-Pontos de Função Brutos e o NI-Nível de Influência:

Tipo da Função	Complexidade Funcional	Peso da Complexidade	Qtde	Subtotal de Pontos
Arquivo Lógico Interno	Simples	7 x	5	35
	Média	10 x	0	0
	Complexa	15 x	0	0
Arquivo de Interface Externa	Simples	5 x	0	0
	Média	7 x	0	0
	Complexa	10 x	0	0
Entrada Externa	Simples	3 x	23	69
	Média	4 x	0	0
	Complexa	6 x	3	18
Saída Externa	Simples	4 x	5	20
	Média	5 x	1	5
	Complexa	7 x	0	0
Consulta Externa	Simples	3 x	7	21
	Média	4 x	1	4
	Complexa	6 x	1	6
Total de Pontos de Função Brutos				178

Características Gerais do Sistema		Nível de Influência
1.	Comunicação de dados	0
2.	Funções distribuídas	0
3.	Performance	1
4.	Configuração do equipamento	1
5.	Volume de transações	0
6.	Entrada de dados on-line	1
7.	Interface com o usuário	1
8.	Atualização on-line	2
9.	Processamento complexo	0
10.	Reusabilidade	1
11.	Facilidade de implantação	0
12.	Facilidade operacional	0
13.	Múltiplos locais	0
14.	Facilidade de mudanças (flexibilidade)	1
Total do Nível de Influência (NI)		8

Posteriormente foi calculado o FA-Fator de Ajuste:

$$FA = (NI \times 0,01) + 0,65$$

$$FA = (8 \times 0,01) + 0,65$$

$$FA = 0,73$$

E, finalmente foram calculados os PFA-Pontos de Função Ajustados:

$$PFA = PFB \times FA$$

PFA = 178 x 0,73

PFA = 129,94

A produtividade real do software, em PFA/pessoas-mês foi então calculada:

Produtividade = Tamanho / Esforço

Produtividade = 129,94 / 3

Produtividade = 43,313 PFA/pessoas-mês

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Tamanho em KLOC

$S = S_{pfa} \times LOC_{pfa}$ [11]

S = 129,94 x 16

S = 2.079,04 LOC [9]

S = 2,079 KLOC

b) Estimativa do Tamanho em PFA

$S = S_{loc} / LOC_{pfa}$ [11]

S = 1.203 / 16

S = 75,188 PFA

c) Estimativa de Produtividade

Produtividade = S / E

Produtividade = 75,188 / 3

Produtividade = 25,063 PFA/pessoas-mês

d) Estimativa do Esforço

$E = S_{pfa}$ / Produtividade

E = 129,94 / 25,063

E = 5,185 pessoas-mês

e) Estimativa do Tempo

T = E / ted [7]

T = 5,185 / 1

T = 5,185 meses

4.1.5.5. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE PARTICULARIDADE

Como este software não apresentou nenhum algoritmo complexo, as estimativas ficam idênticas às estimativas feitas para Pontos de Função (na Seção 4.1.5.4).

4.1.5.6. ANÁLISE DAS MEDIÇÕES/ESTIMATIVAS PARA O SOFTWARE 05

Os gráficos mostrados nas Figuras 18, 19 e 20 apresentam os comparativos entre os valores reais e estimados para o Software 05.

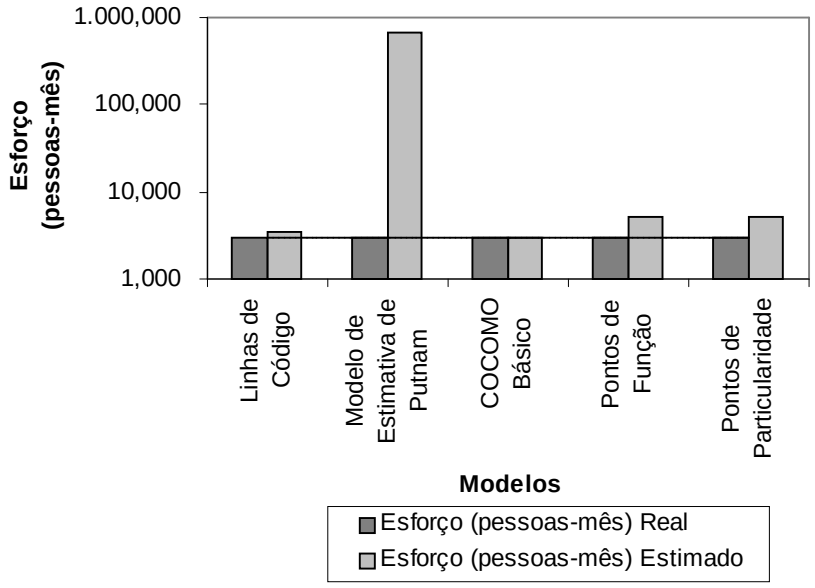

Figura 18. Esforço Real x Estimado para o Software 05

A Figura 18 mostra claramente a indicação de todos modelos para estimar o esforço para este software, exceto o Modelo de Estimativa de Putnam.

Conforme pode ser observado na Figura 19, os Modelos de Linhas de Código e COCOMO Básico mostram-se totalmente adequados para estimar o tamanho (em KLOC) deste software.

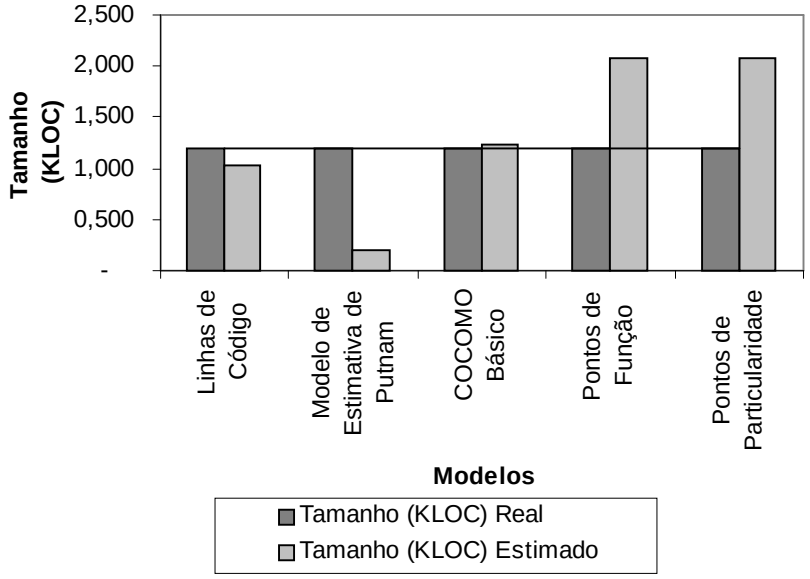

Figura 19. Tamanho Real x Estimado para o Software 05

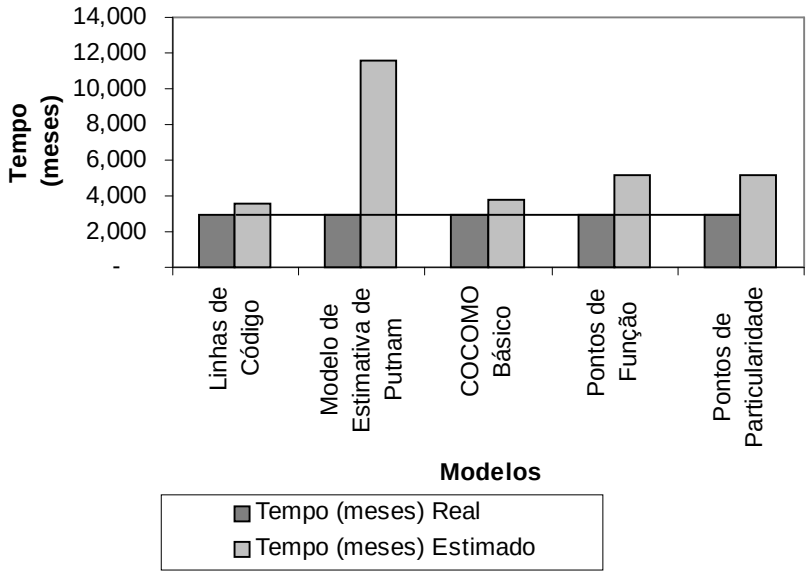

Figura 20. Tempo Real x Estimado para o Software 05

A Figura 20 mostra que o Modelo de Estimativa de Putnam é totalmente inadequado para estimar o tempo de desenvolvimento deste software.

4.1.6. Software 06

Este software é uma aplicação de Sistemas de Informação para controlar a realização de eventos (por exemplo, palestras, seminários, congressos). Os desenvolvedores usaram Delphi, SQL e o banco de dados Oracle. O tempo de desenvolvimento foi de 03 meses e o esforço de 06 pessoas-mês.

4.1.6.1. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO LINHAS DE CÓDIGO

A contagem de Linhas de Código foi realizada conforme detalhado no primeiro parágrafo da seção 4.1 deste trabalho, resultando para este software em 2.486 LOC (ou 2,486 KLOC).

A produtividade real foi calculada usando a equação (18):

$$\text{Produtividade} = S / E$$
$$\text{Produtividade} = 2{,}486 / 6$$
$$\text{Produtividade} = 0{,}414 \text{ KLOC/pessoas-mês}$$

A seguir serão mostradas as estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo:

a) Estimativa do Esforço

$$E = S \times fpl$$
$$E = 2{,}486 \times 2{,}94$$
$$E = 7{,}309 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S = E / fpl$$
$$S = 6 / 2{,}94$$
$$S = 2{,}041 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = S \times fpl / ted^{7}$$
$$T = 2{,}486 \times 2{,}94 / 2$$
$$T = 3{,}654 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 2{,}041 / 7{,}309$$

$$\text{Produtividade} = 0{,}279 \text{ KLOC/pessoas-mês}$$

4.1.6.2. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO DE ESTIMATIVA DE PUTNAM

Como este modelo utiliza unidades de medida diferentes, após os cálculos os valores foram convertidos para unidades padrões (pessoas-mês, KLOC e meses). As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E_{pa} = \frac{S_{loc}{}^{3}}{C^{3} \times t_d{}^{4}}$$

$$E_{pa} = \frac{2.486^{3}}{2000^{3} \times 0{,}25^{4}}$$

$$E_{pa} = 491{,}647 \text{ pessoas-ano}^{8}$$

$$E = 5.899{,}763 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S_{loc} = C \times E_{pa}{}^{1/3} \times t_d{}^{4/3}$$

$$S_{loc} = 2000 \times 0{,}5^{1/3} \times 0{,}25^{4/3}$$

$$S_{loc} = 250{,}00 \text{ LOC}^{9}$$

$$S = 0{,}25 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = (S / C \times E^{1/3})^{3/4}$$

$$T = (2.486 / 2000 \times 0{,}5^{1/3})^{3/4}$$

$$T = 1{,}4 \text{ anos}^{10}$$

$$T = 16{,}799 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 0{,}25 / 5.899{,}763$$

$$\text{Produtividade} = 0{,}000042 \text{ KLOC/pessoas-mês}$$

4.1.6.3. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO COCOMO BÁSICO

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E = a \times S^b$$

$$E = 2,4 \times 2,486^{1,05}$$

$$E = 6,244 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S = (E / a)^{1/b}$$

$$S = (6 / 2,4)^{1/1,05}$$

$$S = 2,393 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = c \times E^d$$

$$T = 2,5 \times 6^{0,38}$$

$$T = 4,939 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 2,393 / 6,244$$

$$\text{Produtividade} = 0,383 \text{ KLOC/pessoas-mês}$$

4.1.6.4. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE FUNÇÃO

Inicialmente foram calculados os PFB-Pontos de Função Brutos e o NI-Nível de Influência:

Tipo da Função	Complexidade Funcional	Peso da Complexidade	Qtde	Subtotal de Pontos
Arquivo Lógico Interno	Simples	7 x	5	35
	Média	10 x	0	0
	Complexa	15 x	0	0
Arquivo de Interface Externa	Simples	5 x	0	0
	Média	7 x	0	0
	Complexa	10 x	0	0
Entrada Externa	Simples	3 x	3	9
	Média	4 x	6	24
	Complexa	6 x	6	36
Saída Externa	Simples	4 x	2	8
	Média	5 x	0	0
	Complexa	7 x	4	28
Consulta Externa	Simples	3 x	3	9
	Média	4 x	2	8
	Complexa	6 x	2	12
Total de Pontos de Função Brutos				169

Características Gerais do Sistema		Nível de Influência
1.	Comunicação de dados	3
2.	Funções distribuídas	0
3.	Performance	1
4.	Configuração do equipamento	1
5.	Volume de transações	0
6.	Entrada de dados on-line	1
7.	Interface com o usuário	1
8.	Atualização on-line	2
9.	Processamento complexo	0
10.	Reusabilidade	1
11.	Facilidade de implantação	0
12.	Facilidade operacional	0
13.	Múltiplos locais	1
14.	Facilidade de mudanças (flexibilidade)	2
Total do Nível de Influência (NI)		13

Posteriormente foi calculado o FA-Fator de Ajuste:

$$FA = (NI \times 0,01) + 0,65$$

$$FA = (13 \times 0,01) + 0,65$$

$$FA = 0,78$$

E, finalmente foram calculados os PFA-Pontos de Função Ajustados:

$$PFA = PFB \times FA$$

PFA = 169 x 0,78

PFA = 131,82

A produtividade real do software, em PFA/pessoas-mês foi então calculada:

Produtividade = Tamanho / Esforço

Produtividade = 131,82 / 6

Produtividade = 21,970 PFA/pessoas-mês

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Tamanho em KLOC

$S = S_{pfa} \times LOC_{pfa}$ [11]

S = 131,82 x 18

S = 2.372,76 LOC [9]

S = 2,373 KLOC

b) Estimativa do Tamanho em PFA

$S = S_{loc} / LOC_{pfa}$ [11]

S = 2.486 / 18

S = 138,111 PFA

c) Estimativa de Produtividade

Produtividade = S / E

Produtividade = 138,111 / 6

Produtividade = 23,019 PFA/pessoas-mês

d) Estimativa do Esforço

$E = S_{pfa}$ / Produtividade

E = 131,82 / 23,019

E = 5,727 pessoas-mês

e) Estimativa do Tempo

T = E / ted [7]

T = 5,727 / 2

T = 2,863 meses

4.1.6.5. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE PARTICULARIDADE

Como este software não apresentou nenhum algoritmo complexo, as estimativas ficam idênticas às estimativas feitas para Pontos de Função (na Seção 4.1.6.4).

4.1.6.6. ANÁLISE DAS MEDIÇÕES/ESTIMATIVAS PARA O SOFTWARE 06

Os gráficos mostrados nas Figuras 21, 22 e 23 apresentam os comparativos entre os valores reais e estimados para o Software 06.

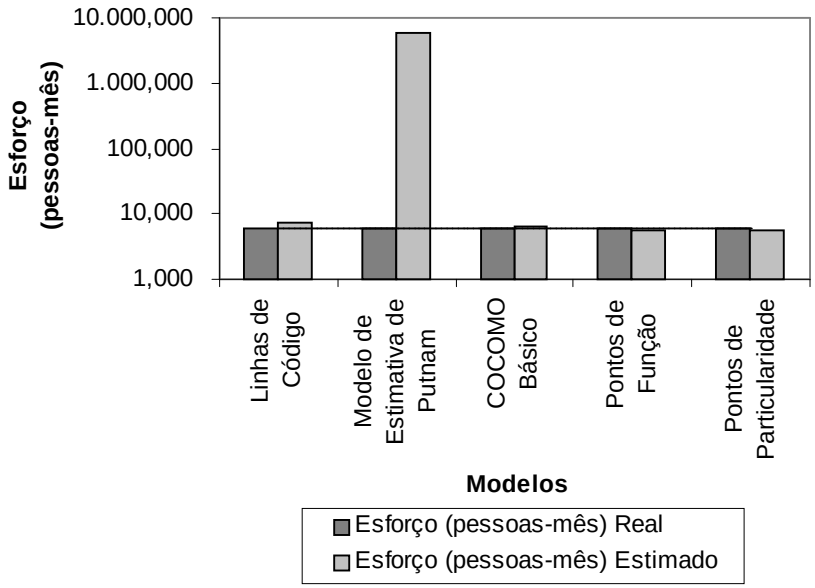

Figura 21. Esforço Real x Estimado para o Software 06

A Figura 21 mostra claramente a indicação de todos modelos para estimar o esforço para este software, exceto o Modelo de Estimativa de Putnam.

Conforme pode ser observado na Figura 22, todos os modelos mostram-se totalmente adequados para estimar o tamanho (em KLOC) deste software, exceto o Modelo de Estimativa de Putnam.

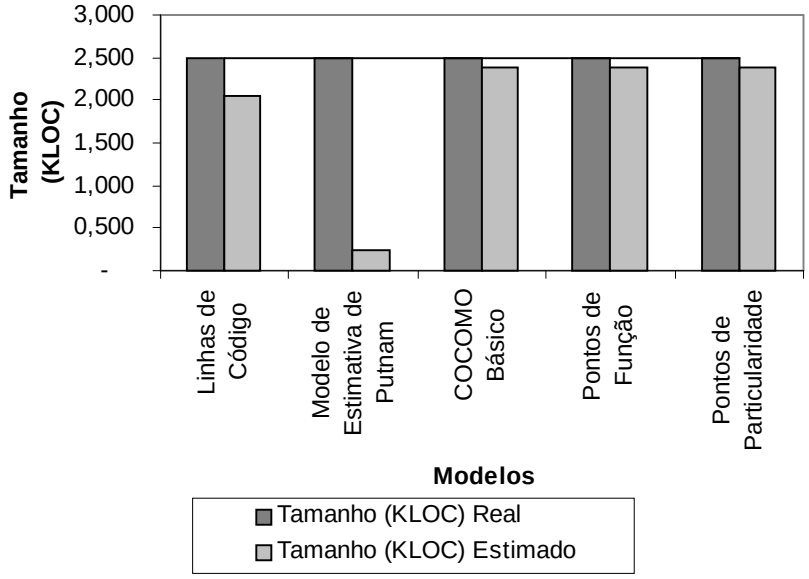

Figura 22. Tamanho Real x Estimado para o Software 06

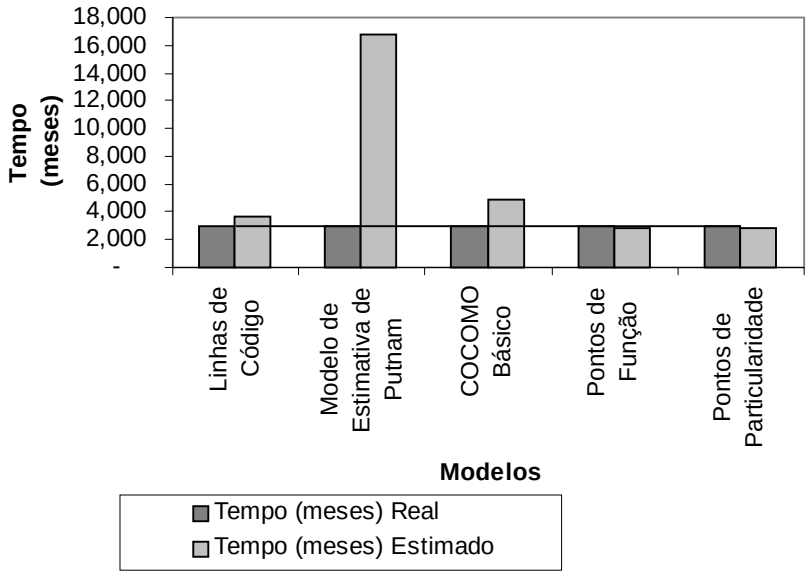

Figura 23. Tempo Real x Estimado para o Software 06

A Figura 23 mostra que, apenas o Modelo de Estimativa de Putnam é totalmente inadequado para estimar o tempo de desenvolvimento deste software.

4.1.7. Software 07

Este software é uma aplicação de Sistemas de Informação para auxiliar no controle de equipamentos de inspeção, medição e ensaios, assegurando a periodicidade e mantendo os registros das confirmações metrológicas conforme exigências das normas ISO 9000 e NBR ISO 10012. O desenvolvedor usou Delphi com Paradox. O tempo de desenvolvimento foi de 03 meses e o esforço de 03 pessoas-mês.

4.1.7.1. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO LINHAS DE CÓDIGO

A contagem de Linhas de Código foi realizada conforme detalhado no primeiro parágrafo da seção 4.1 deste trabalho, resultando para este software em 1.745 LOC (ou 1,745 KLOC).

A produtividade real foi calculada usando a equação (18):

$$\text{Produtividade} = S / E$$
$$\text{Produtividade} = 1{,}745 / 3$$
$$\text{Produtividade} = 0{,}582 \text{ KLOC/pessoas-mês}$$

A seguir serão mostradas as estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo:

a) Estimativa do Esforço

$$E = S \times fpl$$
$$E = 1{,}745 \times 2{,}94$$
$$E = 5{,}13 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S = E / fpl$$
$$S = 3 / 2{,}94$$
$$S = 1{,}020 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = S \times fpl / ted^{7}$$
$$T = 1{,}745 \times 2{,}94 / 1$$
$$T = 5{,}13 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 1{,}02 / 5{,}13$$

$$\text{Produtividade} = 0{,}199 \text{ KLOC/pessoas-mês}$$

4.1.7.2. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO DE ESTIMATIVA DE PUTNAM

Como este modelo utiliza unidades de medida diferentes, após os cálculos os valores foram convertidos para unidades padrões (pessoas-mês, KLOC e meses). As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E_{pa} = \frac{S_{loc}^{\ 3}}{C^3 \times t_d^{\ 4}}$$

$$E_{pa} = \frac{1.745^{\ 3}}{2000^3 \times 0{,}25^4}$$

$$E_{pa} = 170{,}034 \text{ pessoas-ano} \quad [8]$$

$$E = 2.040{,}41 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S_{loc} = C \times E_{pa}^{\ 1/3} \times t_d^{\ 4/3}$$

$$S_{loc} = 2000 \times 0{,}25^{1/3} \times 0{,}25^{4/3}$$

$$S_{loc} = 198{,}00 \text{ LOC} \quad [9]$$

$$S = 0{,}198 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = (S / C \times E^{1/3})^{3/4}$$

$$T = (1.745 / 2000 \times 0{,}25^{1/3})^{3/4}$$

$$T = 1{,}277 \text{ anos} \quad [10]$$

$$T = 15{,}32 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$

$$\text{Produtividade} = 0{,}198 / 2.040{,}41$$

$$\text{Produtividade} = 0{,}000097 \text{ KLOC/pessoas-mês}$$

4.1.7.3. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO O MODELO COCOMO BÁSICO

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Esforço

$$E = a \times S^{b}$$
$$E = 2,4 \times 1,745^{1,05}$$
$$E = 4,306 \text{ pessoas-mês}$$

b) Estimativa do Tamanho

$$S = (E / a)^{1/b}$$
$$S = (3 / 2,4)^{1/1,05}$$
$$S = 1,237 \text{ KLOC}$$

c) Estimativa do Tempo

$$T = c \times E^{d}$$
$$T = 2,5 \times 3^{0,38}$$
$$T = 3,795 \text{ meses}$$

d) Estimativa de Produtividade

$$\text{Produtividade} = S / E$$
$$\text{Produtividade} = 1,237 / 4,306$$
$$\text{Produtividade} = 0,287 \text{ KLOC/pessoas-mês}$$

4.1.7.4. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE FUNÇÃO

Inicialmente foram calculados os PFB-Pontos de Função Brutos e o NI-Nível de Influência:

Tipo da Função	Complexidade Funcional	Peso da Complexidade	Qtde	Subtotal de Pontos
Arquivo Lógico Interno	Simples	7 x	8	56
	Média	10 x	0	0
	Complexa	15 x	0	0
Arquivo de Interface Externa	Simples	5 x	0	0
	Média	7 x	0	0
	Complexa	10 x	0	0
Entrada Externa	Simples	3 x	17	51
	Média	4 x	0	0
	Complexa	6 x	0	0
Saída Externa	Simples	4 x	1	4
	Média	5 x	1	5
	Complexa	7 x	0	0
Consulta Externa	Simples	3 x	10	30
	Média	4 x	1	4
	Complexa	6 x	1	6
Total de Pontos de Função Brutos				156

	Características Gerais do Sistema	Nível de Influência
1.	Comunicação de dados	0
2.	Funções distribuídas	0
3.	Performance	1
4.	Configuração do equipamento	1
5.	Volume de transações	0
6.	Entrada de dados on-line	1
7.	Interface com o usuário	1
8.	Atualização on-line	3
9.	Processamento complexo	0
10.	Reusabilidade	1
11.	Facilidade de implantação	0
12.	Facilidade operacional	0
13.	Múltiplos locais	0
14.	Facilidade de mudanças (flexibilidade)	2
Total do Nível de Influência (NI)		10

Posteriormente foi calculado o FA-Fator de Ajuste:

$$FA = (NI \times 0,01) + 0,65$$

$$FA = (10 \times 0,01) + 0,65$$

$$FA = 0,75$$

E, finalmente foram calculados os PFA-Pontos de Função Ajustados:

$$PFA = PFB \times FA$$

PFA = 156 x 0,75

PFA = 117,00

A produtividade real do software, em PFA/pessoas-mês foi então calculada:

Produtividade = Tamanho / Esforço

Produtividade = 117,00 / 3

Produtividade = 39,000 PFA/pessoas-mês

As estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para este modelo serão apresentadas a seguir:

a) Estimativa do Tamanho em KLOC

$S = S_{pfa}$ x LOC_{pfa} [11]

S = 117,00 x 18

S = 2.106,00 LOC [9]

S = 2,106 KLOC

b) Estimativa do Tamanho em PFA

$S = S_{loc}$ / LOC_{pfa} [11]

S = 1.745 / 18

S = 96,944 PFA

c) Estimativa de Produtividade

Produtividade = S / E

Produtividade = 96,944 / 3

Produtividade = 32,315 PFA/pessoas-mês

d) Estimativa do Esforço

$E = S_{pfa}$ / Produtividade

E = 117,00 / 32,315

E = 3,621 pessoas-mês

e) Estimativa do Tempo

T = E / ted [7]

T = 3,621 / 1

T = 3,621 meses

4.1.7.5. MEDIÇÃO/ESTIMATIVAS CONSIDERANDO PONTOS DE PARTICULARIDADE

Como este software não apresentou nenhum algoritmo complexo, as estimativas ficam idênticas às estimativas feitas para Pontos de Função (na Seção 4.1.7.4).

4.1.7.6. ANÁLISE DAS MEDIÇÕES/ESTIMATIVAS PARA O SOFTWARE 07

Os gráficos mostrados nas Figuras 24, 25 e 26 apresentam os comparativos entre os valores reais e estimados para o Software 07.

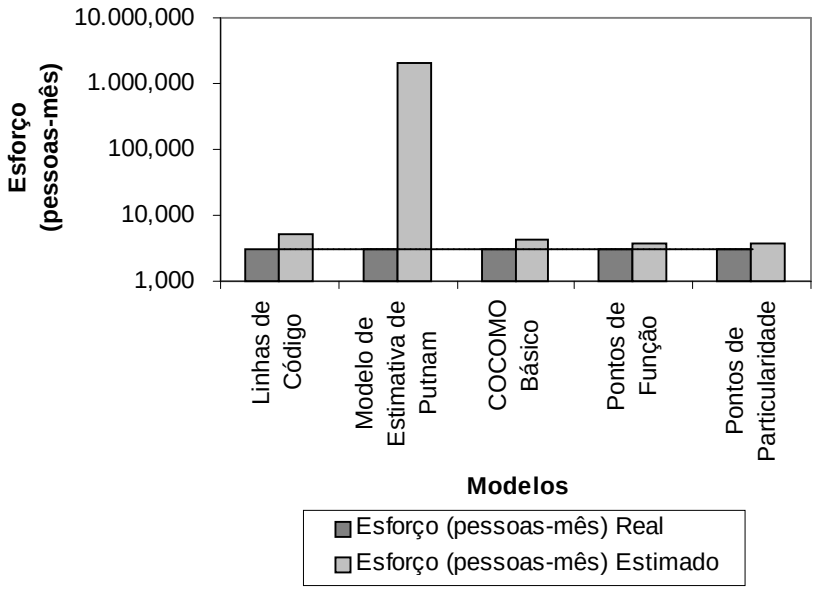

Figura 24. Esforço Real x Estimado para o Software 07

A Figura 24 mostra claramente a indicação de todos modelos para estimar o esforço para este software, exceto o Modelo de Estimativa de Putnam.

Conforme pode ser observado na Figura 25, apenas o Modelo de Estimativa de Putnam mostra-se totalmente inadequado para estimar o tamanho (em KLOC) deste software.

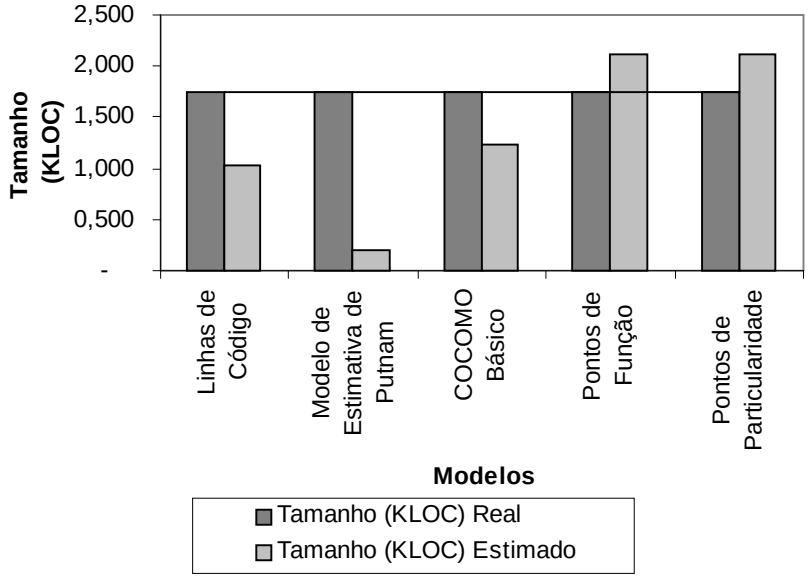

Figura 25. Tamanho Real x Estimado para o Software 07

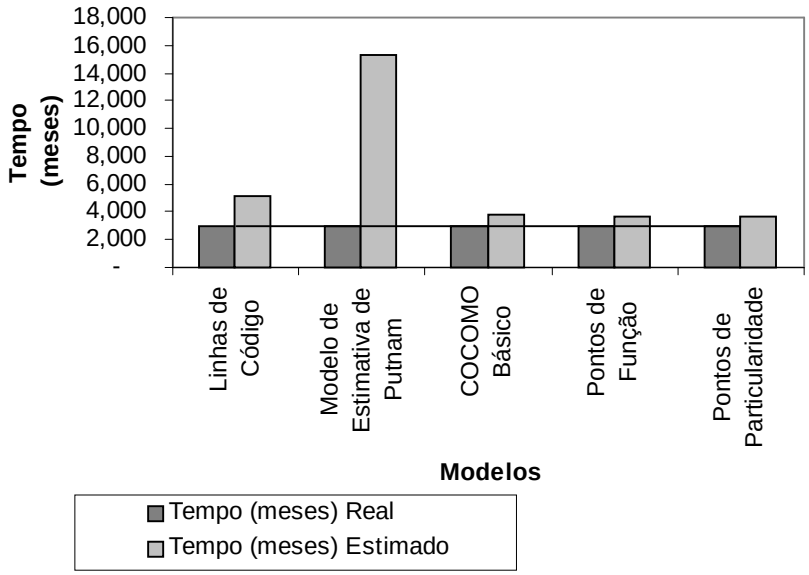

Figura 26. Tempo Real x Estimado para o Software 07

A Figura 26 mostra que apenas o Modelo de Estimativa de Putnam é totalmente inadequado para estimar o tempo de desenvolvimento deste software.

4.1.8. Síntese dos Resultados

4.1.8.1. Erro das estimativas de esforço

As medições e estimativas realizadas podem ter seus resultados sintetizados por intermédio do cálculo dos erros das estimativas de esforço apresentados na Tabela 23:

Tabela 23. Erro das Estimativas de Esforço (em %)

Software	Linhas de Código	Modelo de Estimativa de Putnam	COCOMO Básico	Pontos de Função	Pontos de Particularidade
01	78%	166891%	25%	61%	61%
02	161%	296840%	77%	23%	23%
03	73%	36%	79%	117%	136%
04	64%	62%	55%	59%	51%
05	18%	22185%	3%	73%	73%
06	22%	98229%	4%	5%	5%
07	71%	67914%	44%	21%	21%

A partir da Tabela 23 pode-se construir os gráficos mostrados na Figura 27 e na Figura 28.

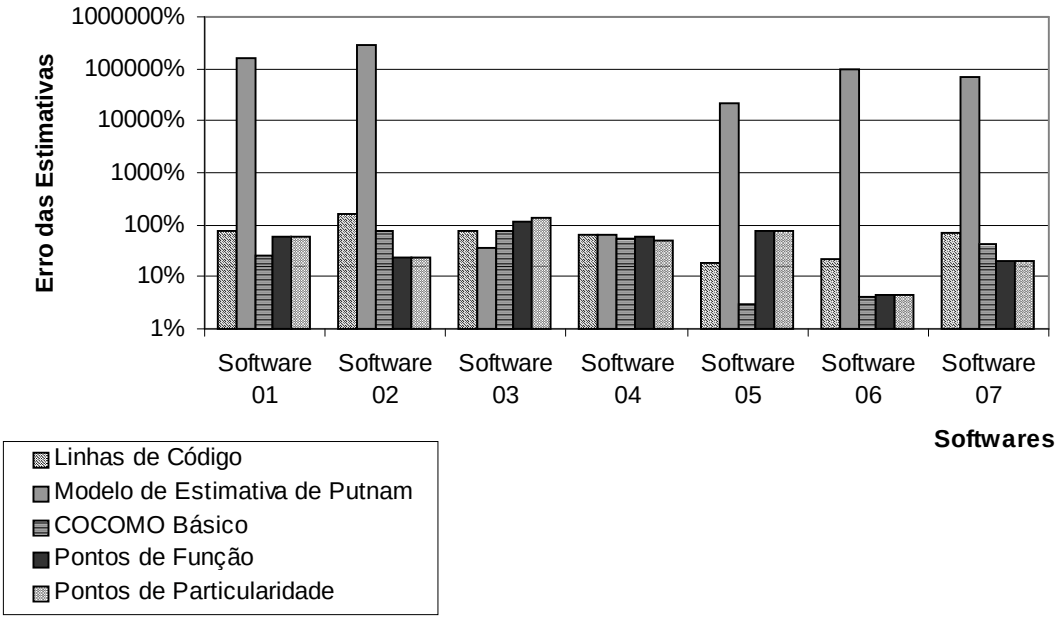

Figura 27. Erro das Estimativas de Esforço por Software

Ao se analisar o gráfico apresentado na Figura 27 observa-se que para os softwares de Inteligência Artificial (03 e 04) os erros das estimativas de esforço foram, em geral, muito

semelhantes ou muito próximos. Destaca-se ainda o fato de os menores erros terem sido observados nos softwares de Sistemas de Informação (05, 06 e 07).

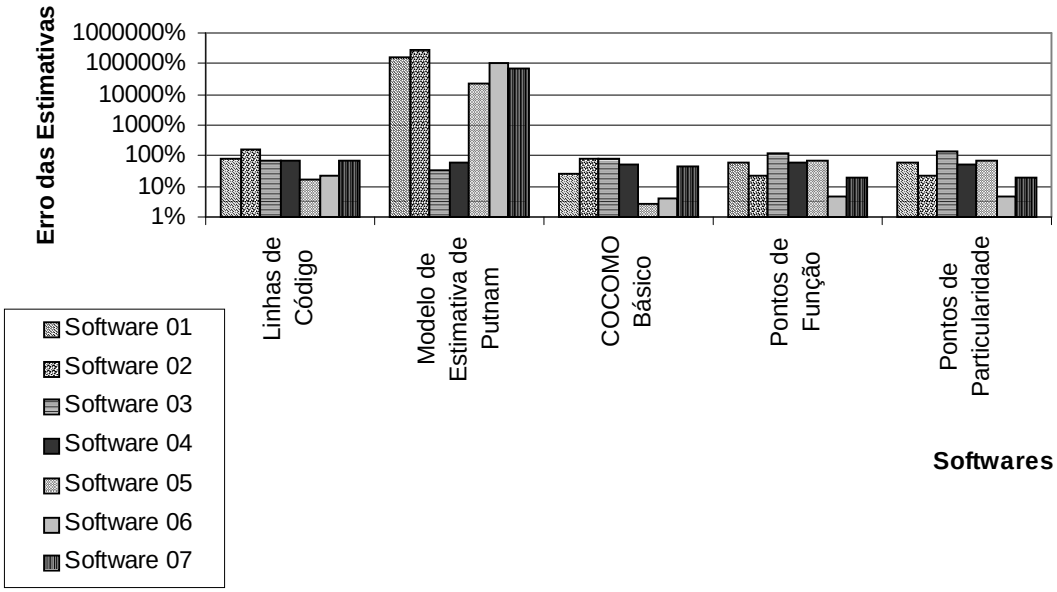

Figura 28. Erro das Estimativas de Esforço por Modelo

O gráfico mostrado na Figura 28 destaca a inadequação do Modelo de Estimativa de Putnam para estimar o esforço, exceto para softwares de Inteligência Artificial (03 e 04). Destaca-se ainda o fato de os menores erros terem sido observados nos modelos COCOMO Básico, Pontos de Função e Pontos de Particularidade.

4.1.8.2. ERRO DAS ESTIMATIVAS DE TAMANHO

As medições e estimativas realizadas podem ter seus resultados sintetizados por meio do cálculo dos erros das estimativas de tamanho apresentados na Tabela 24:

Tabela 24. Erro das Estimativas de Tamanho (em %)

Software	Linhas de Código	Modelo de Estimativa de Putnam	COCOMO Básico	Pontos de Função	Pontos de Particularidade
01	44%	92%	19%	61%	61%
02	62%	93%	42%	47%	47%
03	275%	16%	340%	117%	136%
04	181%	38%	104%	59%	51%
05	15%	84%	3%	73%	73%
06	18%	90%	4%	5%	5%
07	42%	89%	29%	21%	21%

A partir da Tabela 24 pode-se construir os gráficos mostrados na Figura 29 e na Figura 30.

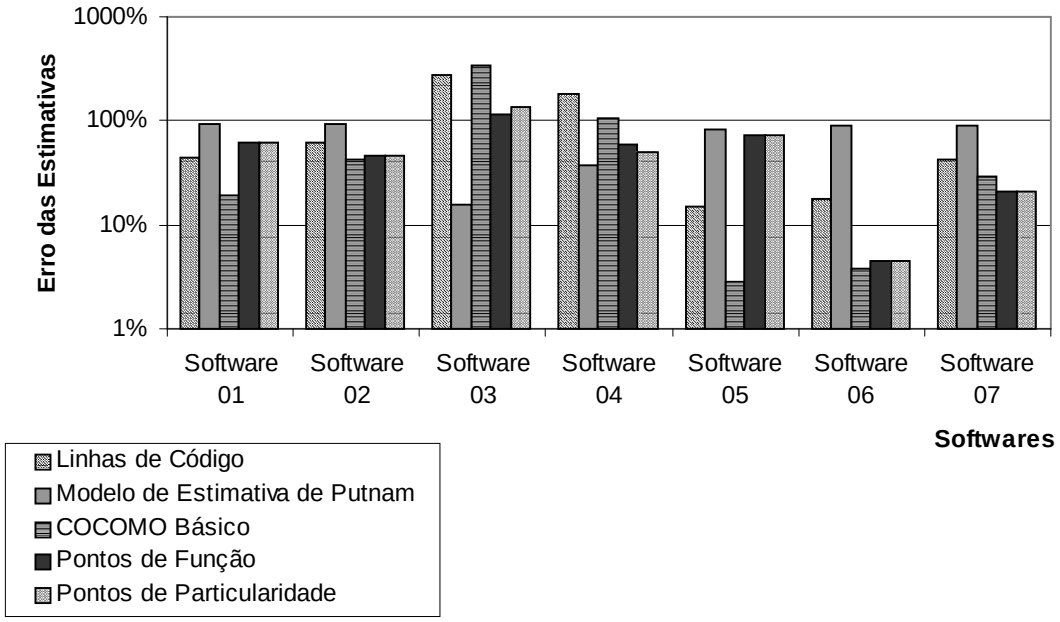

Figura 29. Erro das Estimativas de Tamanho por Software

Ao se examinar o gráfico apresentado na Figura 29 nota-se que para os softwares de Comércio Eletrônico (01 e 02) os erros das estimativas de tamanho foram, em geral, muito semelhantes ou muito próximos. Destaca-se ainda o fato de os menores erros terem sido observados nos softwares de Sistemas de Informação (05, 06 e 07).

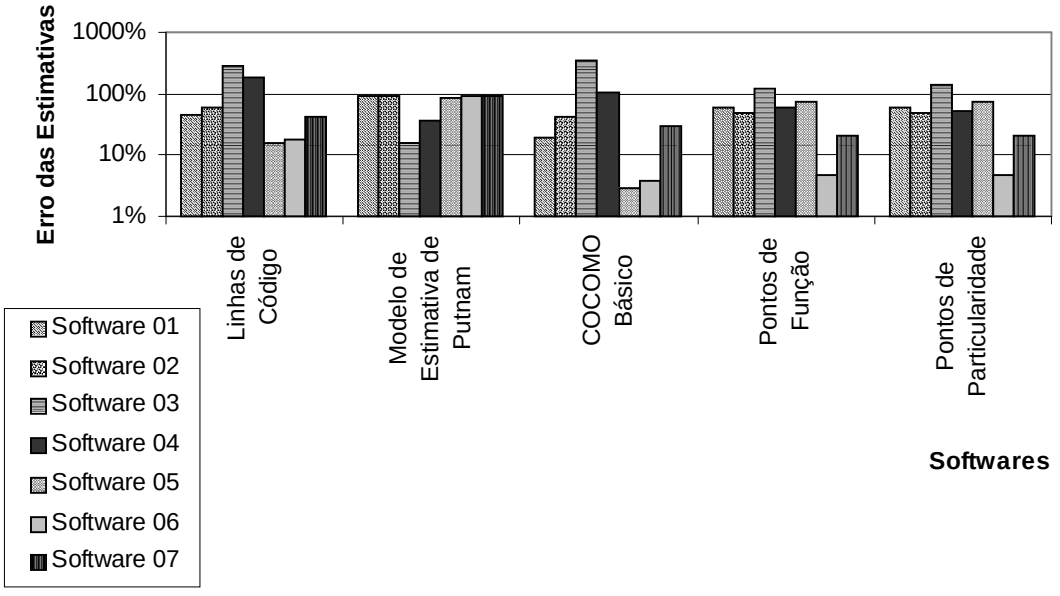

Figura 30. Erro das Estimativas de Tamanho por Modelo

O gráfico mostrado na Figura 30 destaca a homogeneidade dos erros nas estimativas de tamanho no Modelo de Estimativa de Putnam (em geral, muito próximas de 100%), exceto para os softwares de Inteligência Artificial (03 e 04). Ressalta-se ainda o fato de os menores erros terem sido observados nos modelos COCOMO Básico, Pontos de Função e Pontos de Particularidade.

4.1.8.3. ERRO DAS ESTIMATIVAS DE TEMPO

As medições e estimativas realizadas podem ter seus resultados sintetizados por meio do cálculo dos erros das estimativas de tempo apresentados na Tabela 25:

Tabela 25. Erro das Estimativas de Tempo (em %)

Software	Linhas de Código	Modelo de Estimativa de Putnam	COCOMO Básico	Pontos de Função	Pontos de Particularidade
01	78%	539%	65%	61%	61%
02	161%	638%	63%	23%	23%
03	73%	10%	18%	117%	136%
04	64%	21%	60%	59%	51%
05	18%	286%	27%	73%	73%
06	22%	460%	65%	5%	5%
07	71%	411%	27%	21%	21%

Os gráficos mostrados na Figura 31 e na Figura 32 foram construídos a partir da Tabela 25.

Ao se analisar o gráfico apresentado na Figura 31 verifica-se que para os softwares de Inteligência Artificial (03 e 04) os erros das estimativas de tempo foram, em geral, muito semelhantes, ou muito próximos, exceto para o Modelo de Estimativa de Putnam que apresentou os melhores resultados. Os menores erros foram observados nos softwares de Sistemas de Informação (05, 06 e 07).

O gráfico mostrado na Figura 32 destaca a inadequação do Modelo de Estimativa de Putnam para estimar o tempo, exceto para os softwares de Inteligência Artificial (03 e 04). Os menores erros foram observados nos modelos COCOMO Básico, Pontos de Função e Pontos de Particularidade.

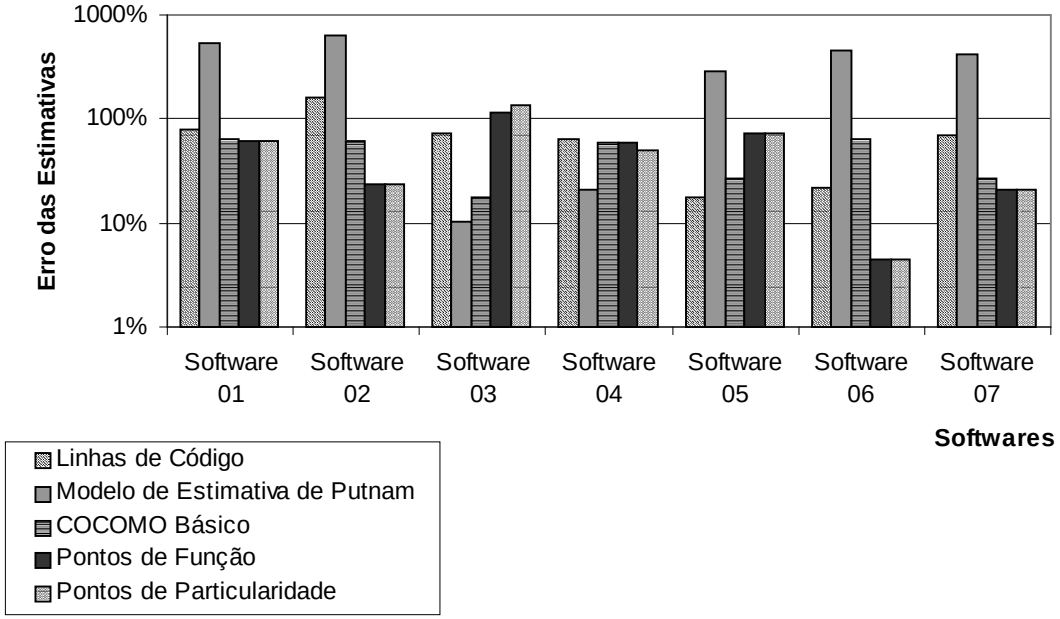

Figura 31. Erro das Estimativas de Tempo por Software

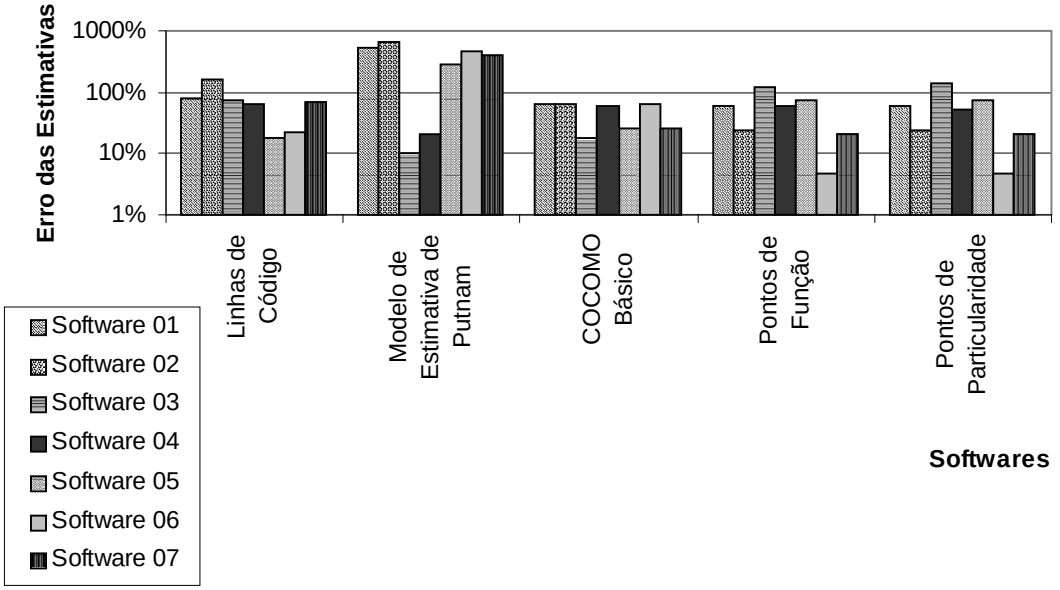

Figura 32. Erro das Estimativas de Tempo por Modelo

4.2. ANÁLISE DOS RESULTADOS POR MODELO

4.2.1. Linhas de Código (*Lines Of Code* - LOC)

Os gráficos mostrados nas Figuras 33, 34, 35 e 36 apresentam comparativos entre os valores reais e estimados para Linhas de Código.

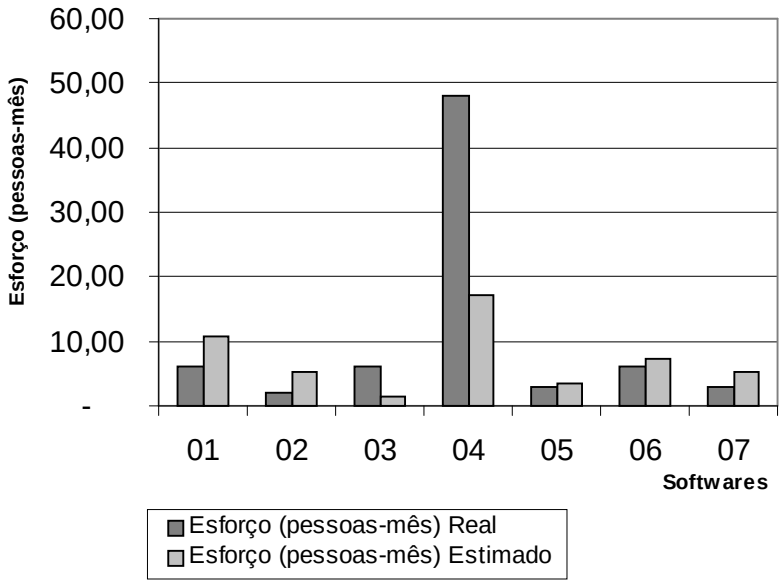

Figura 33. Esforço Real x Estimado para Linhas de Código

Conforme pode ser observado na Figura 33, o esforço estimado é compatível com o esforço real nos softwares de Sistemas de Informação (05, 06 e 07). Já os softwares de Inteligência Artificial (03 e 04) e de Comércio Eletrônico (01 e 02) mostraram relativa disparidade entre os valores reais e estimados.

Para o tamanho (em KLOC) dos softwares observa-se, na Figura 34, que os softwares de Sistemas de Informação são mais adequados a esta métrica do que os demais softwares. Nota-se, inclusive, uma tendência inversa a do esforço, ou seja, os softwares de Inteligência Artificial apresentam, na realidade, tamanhos menores que os estimados, enquanto os softwares de Comércio Eletrônico possuem tamanhos maiores que as estimativas.

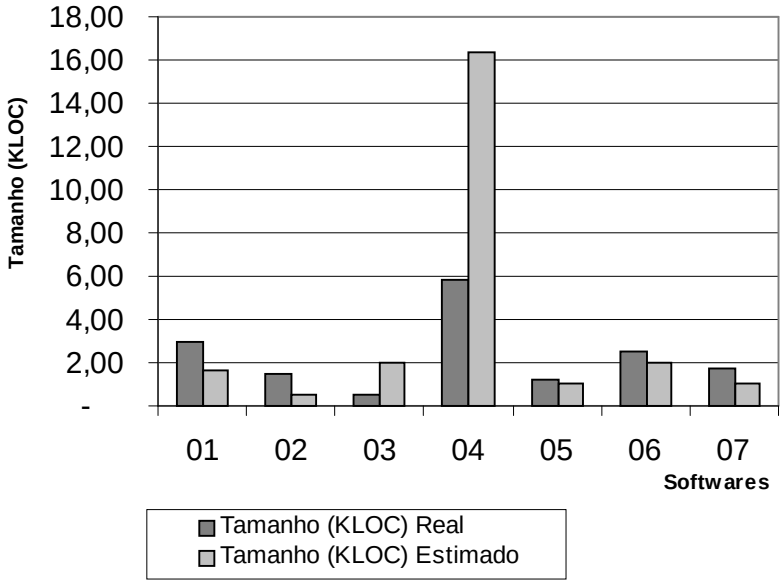

Figura 34. Tamanho Real x Estimado para Linhas de Código

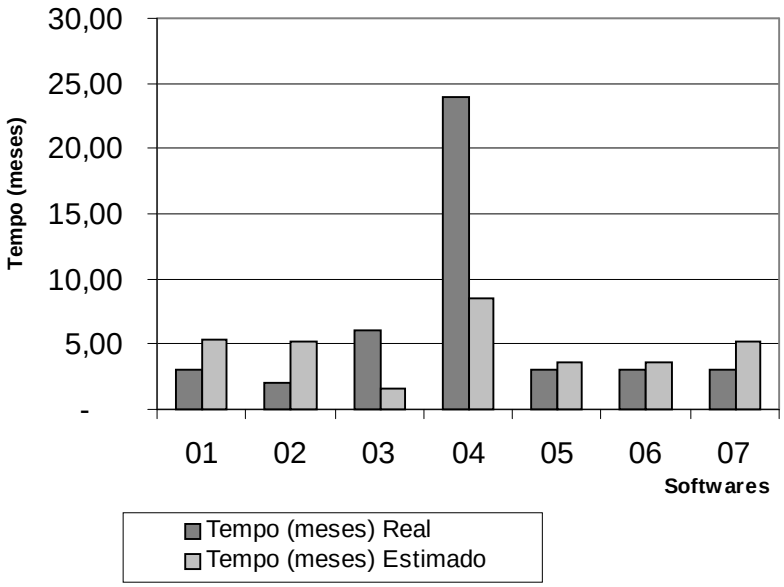

Figura 35. Tempo de Desenvolvimento Real x Estimado para Linhas de Código

Na Figura 35 pode-se observar que, em relação ao tempo de desenvolvimento, a tendência verificada na Figura 33 mantém-se a mesma, ou seja, os softwares de Inteligência Artificial apresentam valores muito menores que os reais nas estimativas realizadas. Já os softwares de Comércio Eletrônico apresentam valores estimados maiores que os reais.

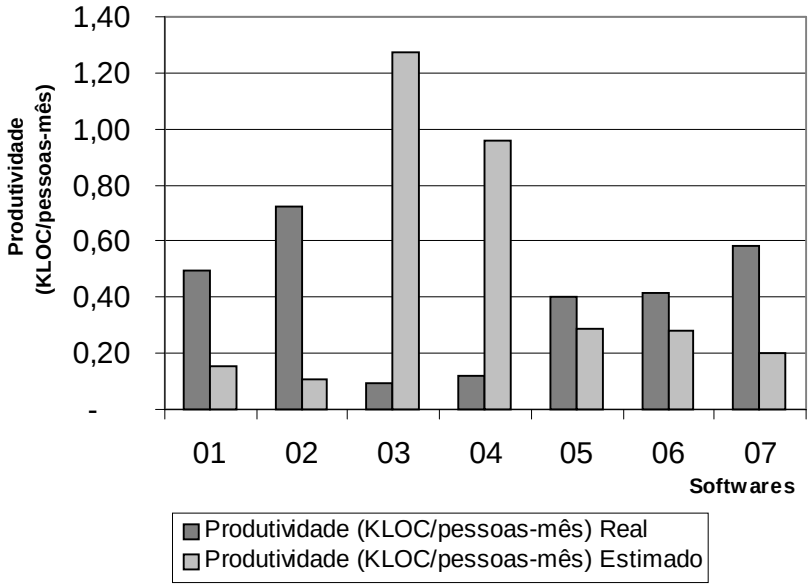

Figura 36. Produtividade Real x Estimada para Linhas de Código

Conforme mostra a Figura 36, há uma disparidade generalizada para os valores reais e estimados de produtividade. As maiores diferenças são observadas nos softwares 03 e 04 (Inteligência Artificial) que apresentam baixíssima produtividade, possivelmente devido à presença de algoritmos complexos nestes softwares. Já os demais softwares mostram uma produtividade real maior que a estimada.

4.2.2. Modelo de Estimativa de Putnam

Os gráficos mostrados nas Figuras 37, 38, 39 e 40 apresentam comparativos entre os valores reais e estimados para o Modelo de Estimativa de Putnam.

A Figura 37 mostra que o modelo de estimativa de Putnam apresenta valores muito diferentes dos reais para os softwares de Comércio Eletrônico (01 e 02) e para os softwares de Sistemas de Informação (05, 06 e 07). Os valores estimados para os softwares de Inteligência Artificial (03 e 04) estão muito próximos dos valores reais.

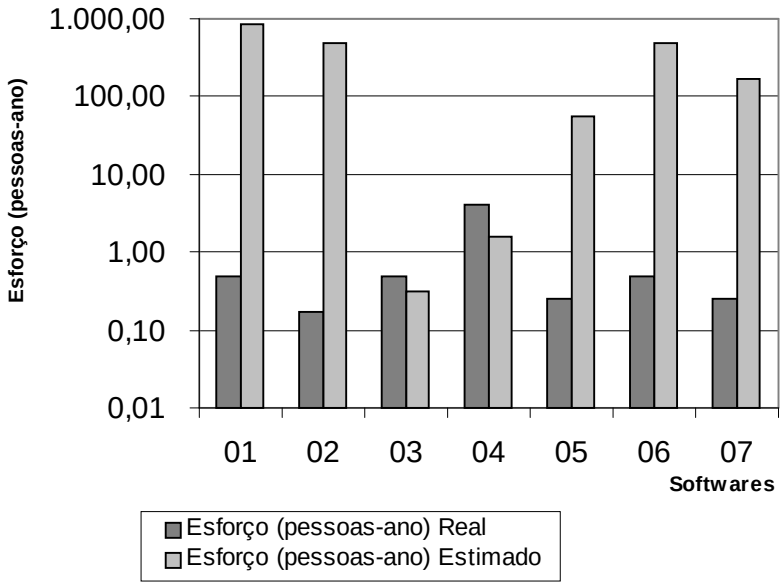

Figura 37. Esforço Real x Estimado para o Modelo de Estimativa de Putnam

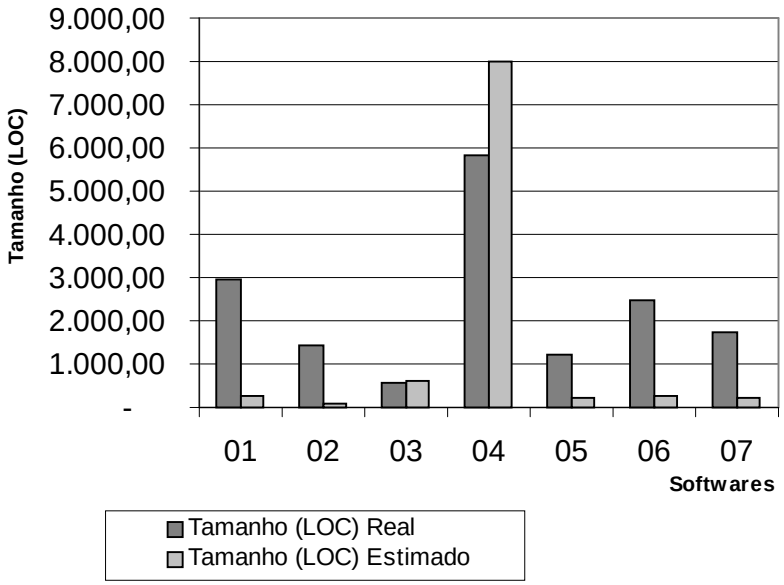

Figura 38. Tamanho Real x Estimado para o Modelo de Estimativa de Putnam

Conforme mostra a Figura 38, esta métrica não é apropriada para medir ou estimar o tamanho de softwares de Comércio Eletrônico. Para os softwares de Inteligência Artificial a métrica é adequada (a diferença no software 04 pode não ser muito significativa, visto que o software é relativamente grande se comparado aos demais). Para Sistemas de Informação, o tamanho estimado é sempre menor que o real.

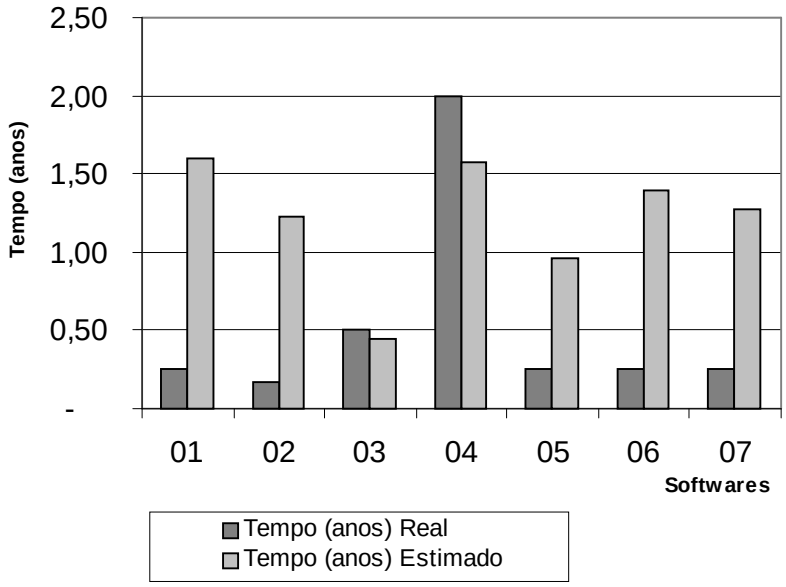

Figura 39. Tempo de Desenvolvimento Real x Estimado para o Modelo de Estimativa de Putnam

Esta métrica mostra-se adequada para estimar o tempo de desenvolvimento de softwares de Inteligência Artificial como mostra a Figura 39. Para os demais softwares, o tempo estimado é muito maior que o real.

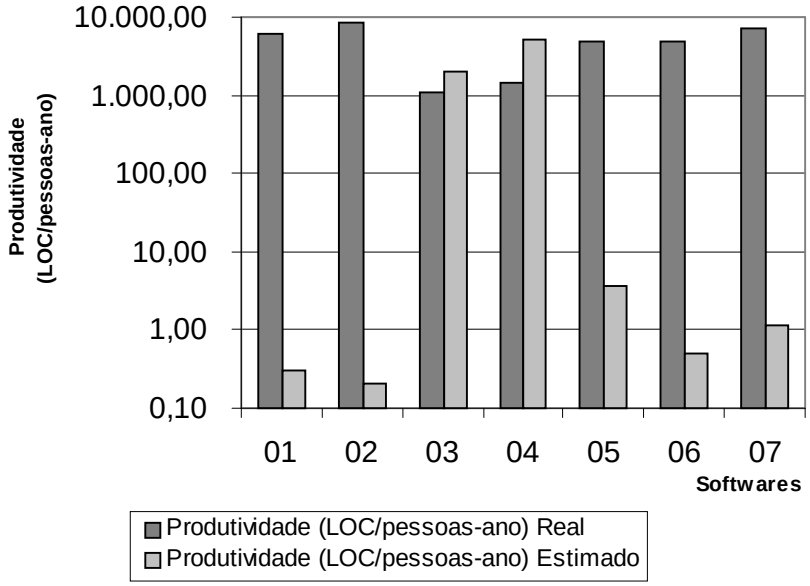

Figura 40. Produtividade Real x Estimada para o Modelo de Estimativa de Putnam

A Figura 40 mostra a não adequação desta métrica para estimar a produtividade para qualquer tipo de software estudado neste trabalho.

4.2.3. COCOMO Básico

Os gráficos apresentados nas Figuras 41, 42, 43 e 44 mostram comparativos entre os valores reais e estimados para o Modelo COCOMO básico.

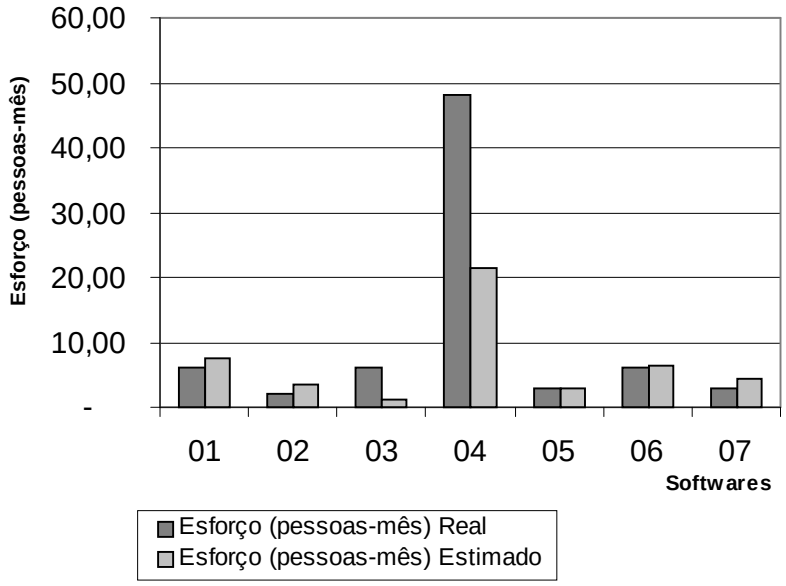

Figura 41. Esforço Real x Estimado para o Modelo COCOMO Básico

A Figura 41 indica a adequação do modelo COCOMO básico para estimar o esforço em softwares de Sistemas de Informação (05, 06 e 07) e em softwares de Comércio Eletrônico (01 e 02). Para softwares de Inteligência Artificial (03 e 04), contudo, este modelo apresentou-se pouco adequado.

Para estimativa do tamanho do software, o modelo COCOMO Básico é inadequado para os softwares de Inteligência Artificial. Para os demais tipos de softwares, o uso deste modelo é perfeitamente aconselhável. Estas observações podem ser constatadas na Figura 42.

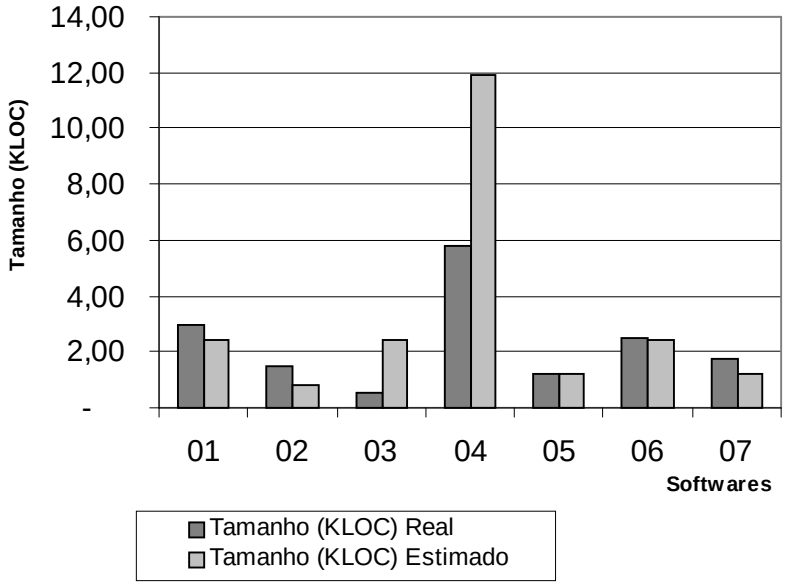

Figura 42. Tamanho Real x Estimado para o Modelo COCOMO Básico

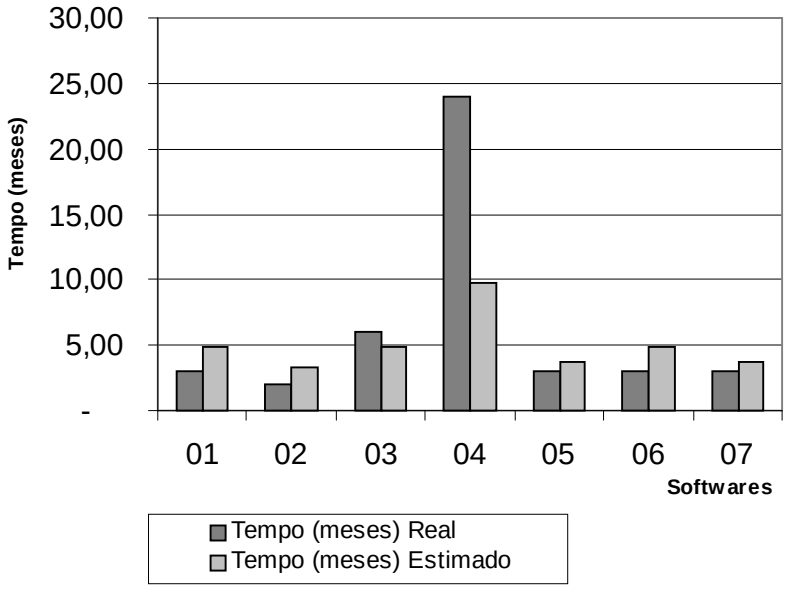

Figura 43. Tempo de Desenvolvimento Real x Estimado para o Modelo COCOMO Básico

A Figura 43 mostra que o modelo COCOMO Básico é adequado para estimar o tempo de desenvolvimento para softwares Comércio Eletrônico e de Sistemas de Informação. Para softwares muito complexos, como os de Inteligência Artificial, este modelo é pouco indicado.

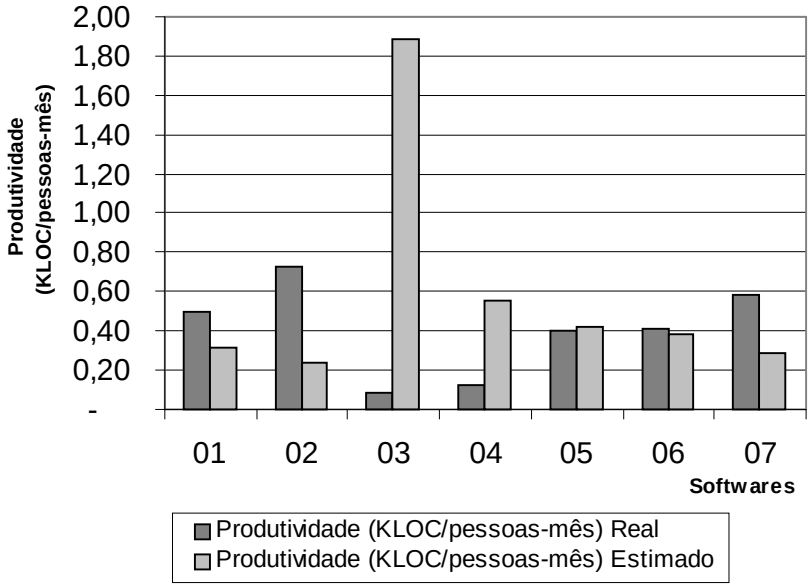

Figura 44. Produtividade Real x Estimada para o Modelo COCOMO Básico

A Figura 44 mostra que o modelo COCOMO Básico é, em geral, adequado para estimar a produtividade em softwares de Sistemas de Informação apresentando compatibilidade entre os valores reais e estimados. Este modelo é ainda contra (ou pouco) indicado para softwares de Comércio Eletrônico e Inteligência Artificial. Cabe ainda ressaltar que os valores mais discrepantes foram encontrados nos softwares 03 e 04.

4.2.4. Pontos de Função *(Function Points)*

Como este modelo não apresenta nenhuma equação para o cálculo do esforço optou-se por derivar a equação (19) a partir da equação (18). Os tamanhos estimados (em PFA e em LOC) foram calculados considerando a Tabela 19 que relaciona Linhas de Código e Pontos de Função.

$$\text{EsforçoEstimado} = \text{TamanhoReal} / \text{ProdutividadeEstimada} \quad (19)$$

Os gráficos mostrados nas Figuras 45, 46, 47, 48 e 49 apresentam comparativos entre os valores reais e estimados para Pontos de Função.

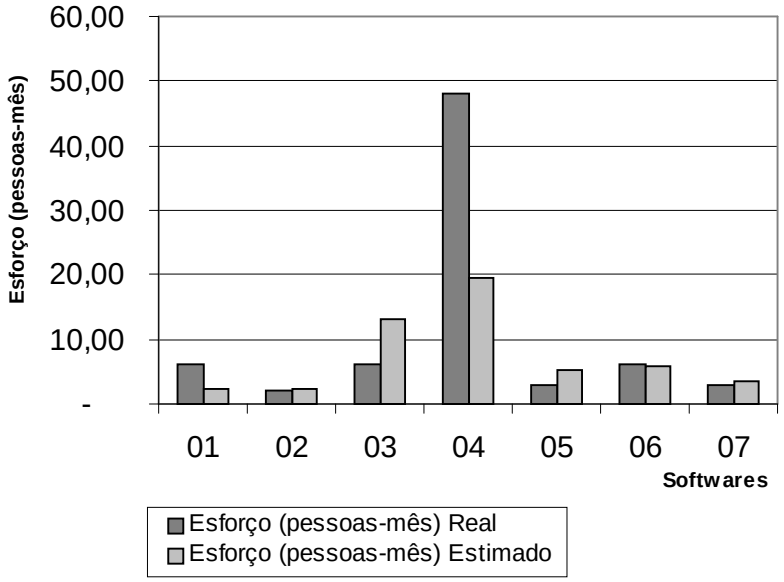

Figura 45. Esforço Real x Estimado para Pontos de Função

A Figura 45 mostra a adequação do modelo de Pontos de Função para softwares de Comércio Eletrônico e de Sistemas de Informação. Observa-se ainda que os valores estimados para softwares 03 e 04 são divergentes dos valores reais.

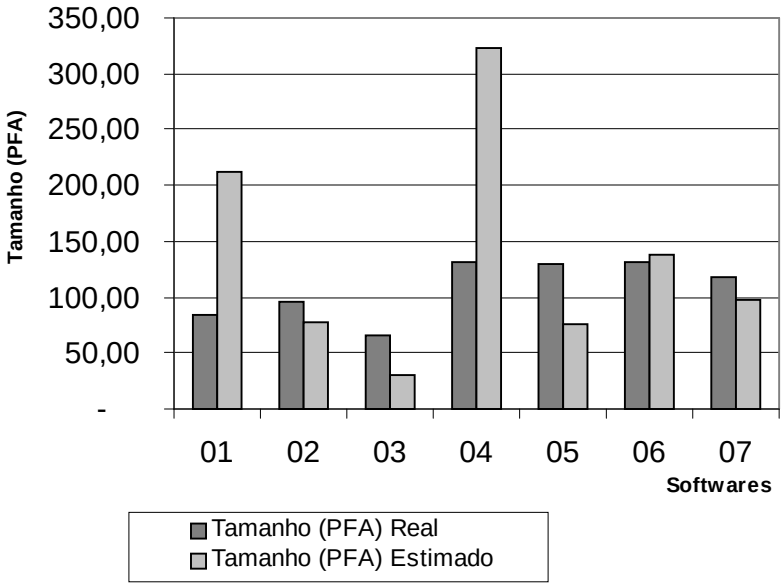

Figura 46. Tamanho em PFA Real x Estimado para Pontos de Função

Conforme pode ser observado na Figura 46, a estimativa da quantidade de PFA baseada na quantidade de LOC não é conveniente para os softwares de Comércio Eletrônico e de

Inteligência Artificial, pois os valores reais e estimados mostram-se muito distanciados. Para os softwares de Sistemas de Informação os valores estimados apresentam-se mais próximos do real, com exceção do software 05, provavelmente em decorrência da não adequação da Tabela 19 ao gerador de código utilizado.

A quantidade real de PFA serviu como base para a realização de uma estimativa do tamanho em LOC (usando a Tabela 19 que relaciona Linhas de Código e Pontos de Função). A Figura 47 ilustra a comparação destes valores.

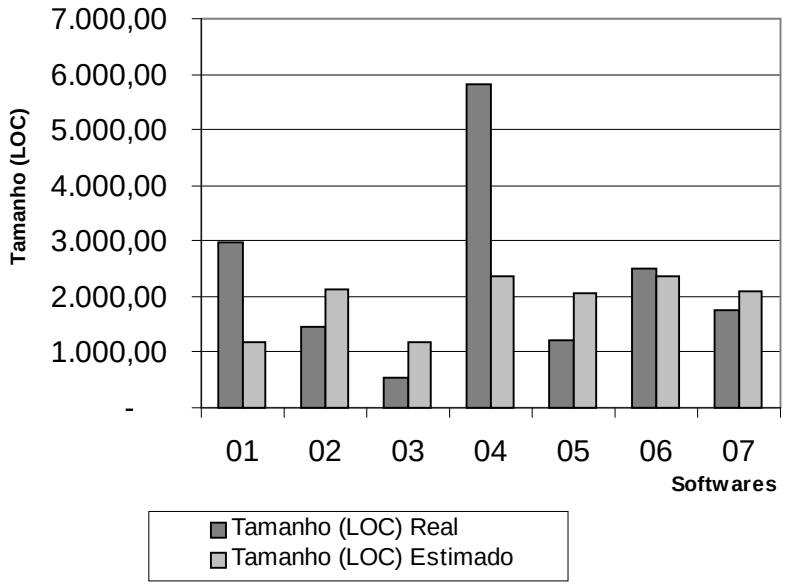

Figura 47. Tamanho em LOC Real x Estimado para Pontos de Função

Para o tamanho (em LOC) dos softwares na Figura 47, nota-se a mesma tendência do gráfico mostrado na Figura 46, ou seja, a não adequação de se relacionar LOC e PFA em softwares de Inteligência Artificial e Comércio Eletrônico.

A Figura 48 mostra que Pontos de Função não devem ser usados em estimativas de tempo de desenvolvimento em softwares de Inteligência Artificial. Para os outros tipos avaliados o modelo é adequado.

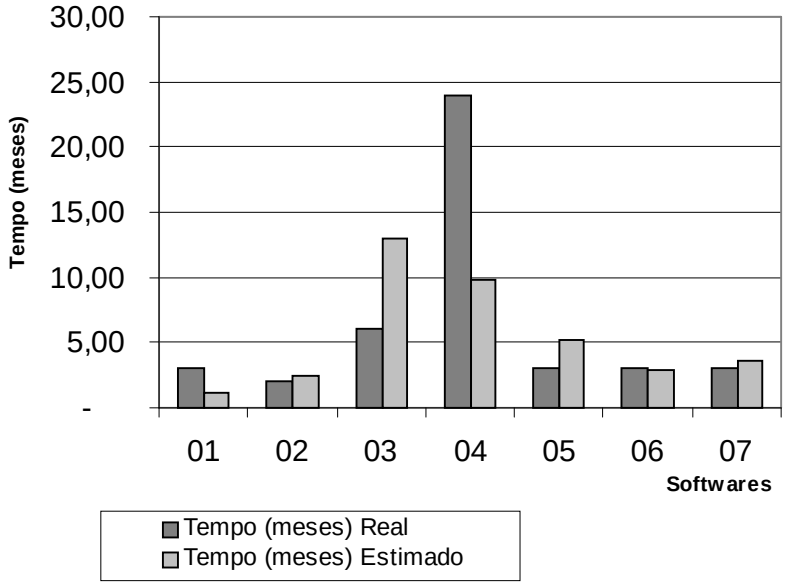

Figura 48. Tempo Real x Estimado para Pontos de Função

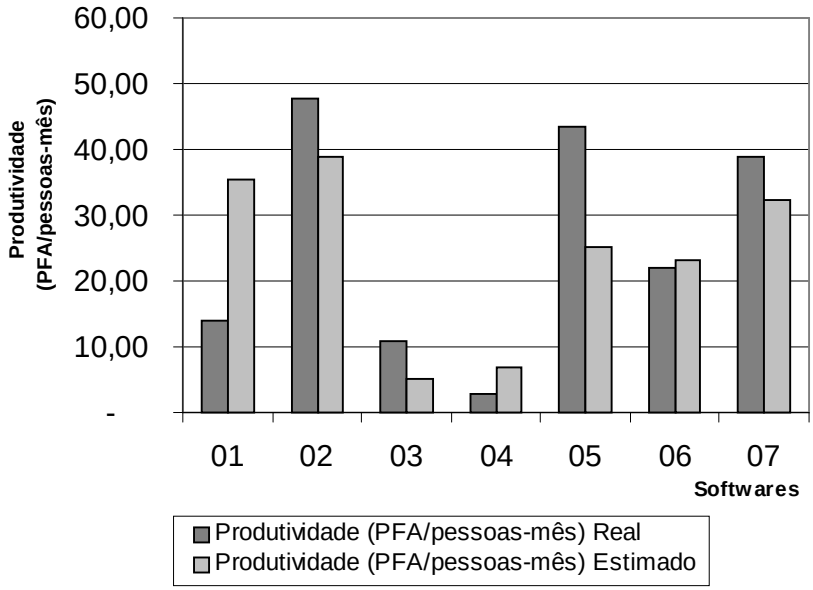

Figura 49. Produtividade Real x Estimada para Pontos de Função

Para estimativa da produtividade, o modelo de Pontos de Função é inadequado para softwares de Comércio Eletrônico e pouco adequado para softwares de Inteligência Artificial e de Sistemas de Informação, como pode ser observado na Figura 49.

4.2.5. Pontos de Particularidade *(Feature Points)*

Para software convencional e sistemas de informação esta métrica produz os mesmos resultados que a métrica de Pontos de Função (CALVERT, 1996). Assim serão analisados apenas os softwares que apresentam valores diferentes dos já apresentados em Pontos de Função, ou seja, os softwares de Inteligência Artificial (03 e 04).

Como não se dispõe de uma tabela que relacione Linhas de Código com Pontos de Particularidade, optou-se por usar novamente a Tabela 19.

Os Gráficos mostrados nas Figuras 50, 51, 52, 53 e 54 mostram que a adição dos algoritmos não foi suficiente para que os softwares de Inteligência Artificial apresentassem convergência entre os valores reais e estimados.

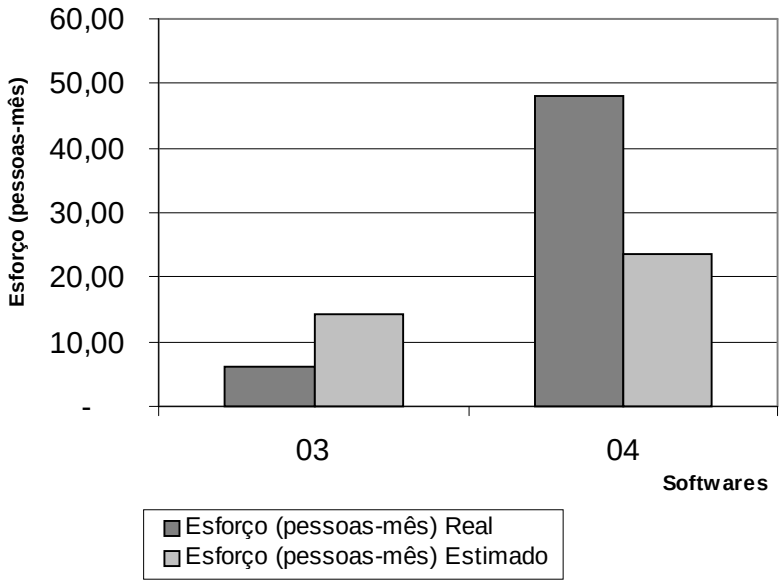

Figura 50. Esforço Real x Estimado para Pontos de Particularidade

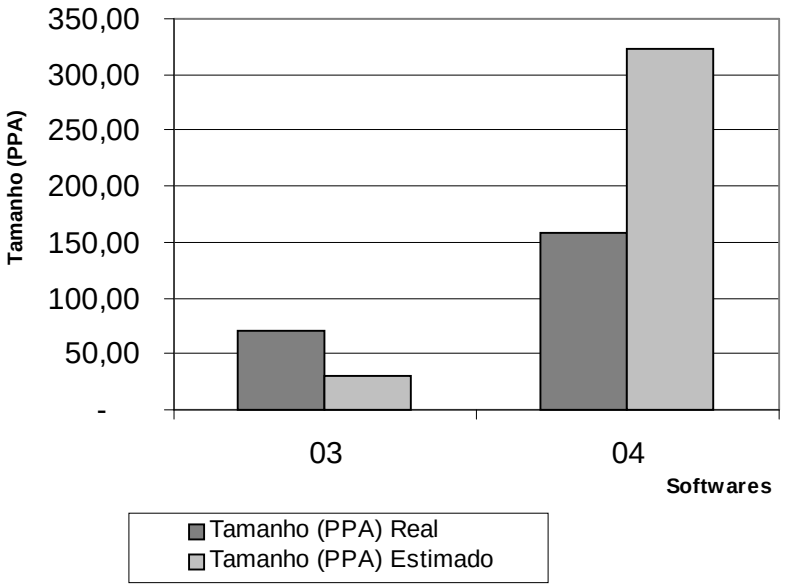

Figura 51. Tamanho em PPA Real x Estimado para Pontos de Particularidade

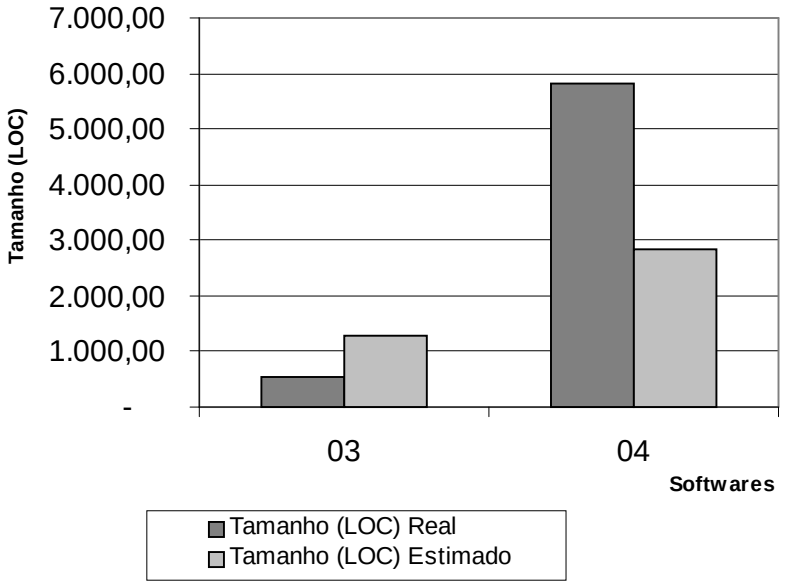

Figura 52. Tamanho em LOC Real x Estimado para Pontos de Particularidade

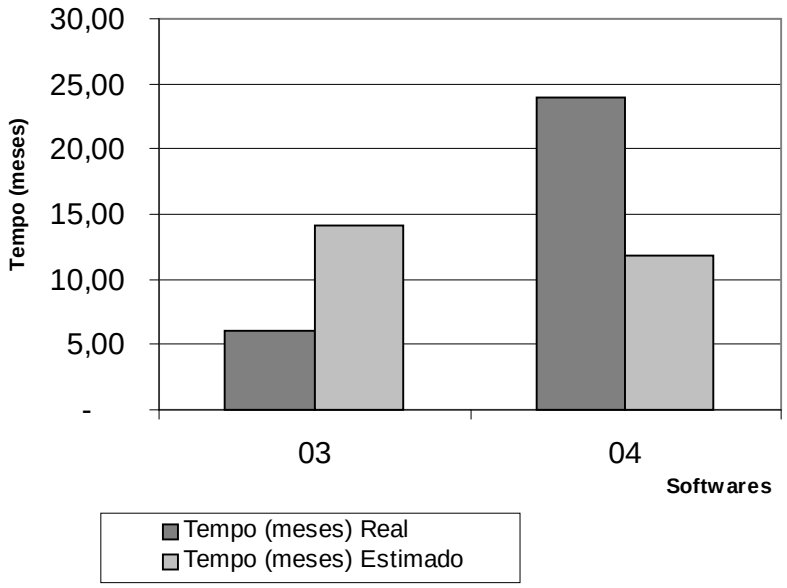

Figura 53. Tempo Real x Estimado para Pontos de Particularidade

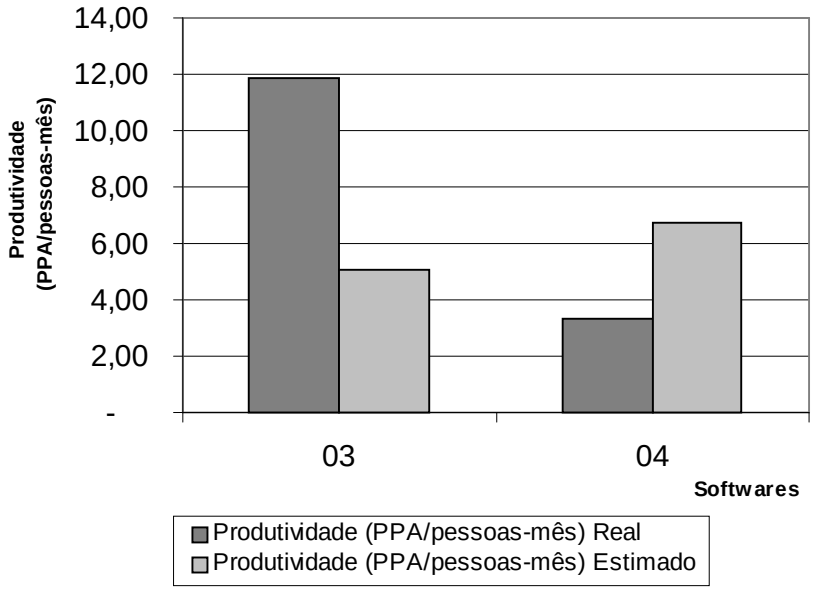

Figura 54. Produtividade Real x Estimada para Pontos de Particularidade

4.3. ANÁLISE POR TIPO DE SOFTWARE

Entre os softwares avaliados, os de Inteligência Artificial foram os que apresentaram as menores taxas de produtividade real, tanto considerando-se linhas de código, como ponto de função.

4.3.1. Softwares de Comércio Eletrônico

A Tabela 26 mostra os Modelos Algorítmicos testados mais indicados para se fazer estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade para softwares de Comércio Eletrônico. Para este tipo de software o modelo COCOMO Básico foi o que apresentou os melhores resultados.

Tabela 26. Indicação de Modelos para Estimativas de Softwares de Comércio Eletrônico

Estimativas / Modelos	Esforço	Tamanho	Tempo de Desenvolvimento	Produtividade
Linhas de Código	Pouco indicado	Pouco indicado	Pouco indicado	Não indicado
Modelo de Estimativa de Putnam	Não indicado	Não indicado	Não indicado	Não indicado
COCOMO Básico	**Indicado**	**Indicado**	**Indicado**	Pouco indicado
Pontos de Função	**Indicado**	Não indicado	**Indicado**	Não indicado
Pontos de Particularidade	**Indicado**	Não indicado	**Indicado**	Não indicado

4.3.2. Softwares de Inteligência Artificial

Conforme apresenta a Tabela 27, os Modelos Algorítmicos testados, em geral, não se adaptam muito bem para utilização em estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade de softwares de Inteligência Artificial. Apenas o Modelo de Estimativa de Putnam mostrou-se adequado para estimar o esforço nesse tipo de software.

Tabela 27. Indicação de Modelos para Estimativas de Softwares de Inteligência Artificial

Estimativas / Modelos	Esforço	Tamanho	Tempo de Desenvolvimento	Produtividade
Linhas de Código	Não indicado	Não indicado	Não indicado	Não indicado
Modelo de Estimativa de Putnam	**Indicado**	Pouco indicado	Pouco indicado	Não indicado
COCOMO Básico	Pouco indicado	Não indicado	Pouco indicado	Não indicado
Pontos de Função	Não indicado	Não indicado	Não indicado	Pouco indicado
Pontos de Particularidade	Não indicado	Não indicado	Não indicado	Pouco indicado

4.3.3. Softwares de Sistemas de Informação

Quando se trata de Sistemas de Informação, a maioria dos Modelos Algorítmicos testados manifesta-se adequada para se fazer estimativas de esforço, tamanho, tempo de desenvolvimento e produtividade, com exceção do Modelo de Estimativa de Putnam que não apresentou resultados admissíveis, como está resumido na Tabela 28.

Tabela 28. Indicação de Modelos para Estimativas de Softwares de Sistemas de Informação

Estimativas / Modelos	Esforço	Tamanho	Tempo de Desenvolvimento	Produtividade
Linhas de Código	**Indicado**	**Indicado**	**Indicado**	Pouco indicado
Modelo de Estimativa de Putnam	Não indicado	Não indicado	Não indicado	Não indicado
COCOMO Básico	**Indicado**	**Indicado**	**Indicado**	**Indicado**
Pontos de Função	**Indicado**	**Indicado**	**Indicado**	Pouco indicado
Pontos de Particularidade	**Indicado**	**Indicado**	**Indicado**	Pouco indicado

4.4. COMENTÁRIOS FINAIS

Neste capítulo foi realizado um estudo de caso com vários softwares que representam três diferentes domínios de aplicação. Para cada um dos softwares foram medidos os valores reais e estimados para esforço e prazo de desenvolvimento, entre outros. Estes valores foram

analisados e comparados para cada software, para cada modelo e para cada tipo de software. Algumas constatações muito interessantes são expostas:

- como os softwares de Sistemas de Informação são mais convencionais e sem grande complexidade, praticamente todos os modelos estudados adaptam-se, razoavelmente bem a este domínio de aplicação, exceto o Modelo de Estimativa de Putnam, isso apenas com softwares de pequeno porte;
- para softwares de Inteligência Artificial, aconteceu o contrário, ou seja, nenhum modelo pareceu adaptar-se a este domínio de aplicação, com exceção do Modelo de Estimativa de Putnam que apresentou os resultados menos ruins;
- com os softwares de Comércio Eletrônico os resultados foram mais heterogêneos: os Modelos COCOMO Básico, Pontos de Função e Pontos de Particularidade apresentaram os melhores resultados para esforço e tempo de desenvolvimento, mas não apresentaram bons resultados para estimativa de tamanho e produtividade.

5. CONCLUSÕES E RECOMENDAÇÕES

5.1. CONCLUSÕES

Neste trabalho, foram estudadas as principais técnicas para estimativa de custo de software: Julgamento Especialista ou Parecer Técnico, Analogia e Modelos Algorítmicos. Desta última técnica ainda foram estudados os principais modelos: Linhas de Código, Modelo de Estimativa de Putnam, Modelo de Custo Construtivo (mais conhecido por COCOMO), Pontos de Função, Pontos de Particularidade, Ciência do Software de Halstead e, finalmente, o Número Ciclomático de McCabe.

Realizou-se uma análise dos aspectos positivos e negativos das principais técnicas e dos principais modelos utilizados em estimativa de custo de software. Estas técnicas e modelos foram também analisados comparativamente obedecendo alguns critérios.

Desta análise, conclui-se que o modelo de Pontos de Função apresenta maior utilização comercial e, conseqüentemente, concentra um grande número de publicações sobre utilização. Este modelo conta, inclusive, com um grupo internacional de usuários, o IFPUG e com vários grupos de usuários espalhados pelo mundo (no Brasil o grupo é conhecido como BFPUG).

Outro ponto que merece destaque é que a maior parte das técnicas e modelos estudados:

- é dependente da linguagem de programação utilizada;
- não considera o uso de técnicas e ferramentas automatizadas;
- não considera a experiência da equipe de desenvolvimento;
- não está preparada para ser aplicada em projetos de manutenção de software.

Em geral, essas técnicas e modelos são aplicáveis independentemente do tamanho do projeto, com exceção do Modelo de Estimativa de Putnam que é direcionado a projetos de grande porte.

Quanto à etapa de desenvolvimento em que essas técnicas e modelos podem ser aplicados, o ideal é que a etapa de projeto (do Ciclo de Vida Clássico proposto em Pressman (1995)) esteja concluída, ou ao menos que se tenha um esboço razoavelmente claro dessa etapa.

Alguns Modelos Algorítmicos foram escolhidos para realização de um estudo de caso realizado com 07 (sete) softwares de pequeno porte selecionados, conforme alguns critérios, para representar diferentes domínios de aplicação de software: Comércio Eletrônico, Inteligência Artificial e Sistemas de Informação.

Os resultados obtidos com o estudo de caso, ou seja, os valores reais e estimados verificados em cada um dos softwares avaliados, foram analisados com o objetivo de verificar as diferenças entre estes valores para cada modelo aplicado.

Os softwares avaliados foram divididos em três tipos: Comércio Eletrônico, Inteligência Artificial e Sistemas de Informação com o objetivo de verificar, no estudo de caso, possíveis relações entre estes tipos e os modelos estudados.

Detectou-se que para softwares de Comércio Eletrônico o Modelo COCOMO Básico é indicado para se estimar esforço, tamanho e tempo de desenvolvimento, ao passo que os Pontos de Função e os Pontos de Particularidade são adequados apenas para esforço e tempo. Para softwares de Inteligência Artificial, o Modelo de Estimativa de Putnam é adequado para estimar apenas o esforço e nenhum dos outros modelos mostrou-se indicado para este tipo de software. Já em Sistemas de Informação é possível utilizar Linhas de Código, Pontos de Função e Pontos de Particularidade para estimar esforço, tamanho e tempo de desenvolvimento. O Modelo COCOMO Básico pode também ser utilizado para se estimar esforço, tamanho, tempo de desenvolvimento e produtividade.

Com exceção do Modelo COCOMO Básico aplicado em Sistemas de Informação, nenhum modelo foi adequado para se estimar a produtividade. Cabe ainda salientar que, os softwares de Inteligência Artificial apresentaram os menores valores para a produtividade real.

O Modelo de Estimativa de Putnam mostrou-se inadequado para estimar esforço e tempo de desenvolvimento, exceto para os softwares de Inteligência Artificial. Há certa homogeneidade

nas taxas de erro das estimativas de tamanho feitas pelo Modelo de Estimativa de Putnam, todas próximas de 100%, exceto, mais uma vez, para os softwares de Inteligência Artificial.

As menores taxas de erro nas estimativas de esforço, tamanho e tempo foram encontradas nos modelos COCOMO Básico, Pontos de Função e Pontos de Particularidade aplicados a softwares de Sistemas de Informação.

5.2. RECOMENDAÇÕES PARA TRABALHOS FUTUROS

A construção de um Sistema Especialista para fazer estimativas com a utilização da técnica de Julgamento Especialista ou Parecer Técnico é recomendada para futuras pesquisas. Como é pequena a disponibilidade dos especialistas em estimativas de custo de software, a construção deste sistema pode ser uma forma de se resolver este problema.

Pode-se realizar estimativas por Analogia com a construção de um Banco de Dados que armazene os dados históricos de uma determinada empresa. Para detectar os melhores casos a serem utilizados, pode-se usar alguma técnica de Inteligência Artificial, como por exemplo Redes Neurais Artificiais ou Raciocínio Baseado em Casos.

Seria importante ainda a realização de um estudo de caso sobre a aplicação das variações do Modelo COCOMO (Intermediário e Detalhado) para estimar custos de software. Poder-se-ia desenvolver um software que auxiliasse o usuário na definição dos multiplicadores de esforço para cada direcionador de custo.

Para a aplicação dos modelos Ciência do Software de Halstead e Número Ciclomático de McCabe, recomenda-se o desenvolvimento de um software que faça a varredura do código fonte de um programa (em arquivo texto, por exemplo), para calcular o Número Ciclomático de McCabe e aplicar a Ciência do Software de Halstead, inclusive automatizando a contagem de Linhas de Código, conforme recomendações de Park (1992).

Seria pertinente avaliar uma quantidade maior de softwares, em especial, de Inteligência Artificial, a fim de verificar melhor a aplicação/adequação dos modelos, principalmente de Pontos de Particularidade, que pela literatura é recomendado para estes tipos de software, mas

que, pelo estudo de caso realizado não o é. Seria conveniente realizar estudos mais minuciosos (e, com mais softwares) para todos os tipos de software e modelos de estimativa “pouco indicados” conforme consta das Tabelas 26, 27 e 28. É importante avaliar o desempenho das técnicas e dos modelos estudados, aplicados a outros tipos de software, como também a softwares corporativos.

REFERÊNCIAS

AGENA Ltd. **Quality assurance and metrics**. (s.l.), 2000. Disponível em <http://www.agena.co.uk/qa_metrics_article>. Acesso em 07/09/2000.

AZEVEDO, Douglas José Peixoto de. **Análise de pontos por função para aplicações orientadas a documentos**. 1999. 112. Dissertação (Mestrado em Ciência da Computação), Universidade Federal do Rio Grande do Sul, Porto Alegre, 1999.

BOEHM, Barry W. **Software engineering economics**. New Jersey: Prentice-Hall, 1981.

BORNIA, Antonio Cezar. **Análise gerencial de custos**: aplicação em empresas modernas. Porto Alegre: Bookman, 2002.

BOURNEMOUTH University. **Software cost estimation**. (s.l.), 1997. Disponível em <http://www.ecfc.u-net.com/cost/front.htm>. Acesso em 07/09/2000.

BRAGA, Antônio. **Análise de pontos de função**. Rio de Janeiro: Infobook, 1996.

CALVERT, David. **Software metrics**. Ontário, 1996. Disponível em <http://hebb.cis.uoguelph.ca/~dave/27320/new/metrics.html>. Acesso em 22/11/2001.

CHIZZOTTI, Antonio. **Pesquisa em ciências humanas e sociais**. São Paulo: Cortez, 2000.

CRC-SP - CONSELHO REGIONAL DE CONTABILIDADE do Estado de São Paulo. **Custo como ferramenta gerencial**. São Paulo: Atlas, 1995.

COST XPERT Group. **About Cost Xpert**. Califórnia. Disponível em <http://www.costxpert.com/about/about.html>. Acesso em 26/03/2002.

COCKBURN, Craig. Estimating software projects using *ObjectMetrix*^{TM}. **Methods & Tools**, Vevey, v. 7, n. 3, p. 8-13, Fall/1999. Disponível em <http://www.martinig.ch/mt/dmt0399.pdf>. Acesso em 24/04/2001.

CRUZ, Cláudia Dib; WERNER, Cláudia Maria Lima; SOARES, Jeferson Ferreira. **Em direção a um modelo de custos de desenvolvimento de software orientado a objetos**. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 13., 1999, Florianópolis. **Anais...** Florianópolis: UFRGS e UFSC, 1999.

DEKKERS, Carol A. **Pontos de função e medidas**: o que é um ponto de função? Flórida, 1999. Disponível em <http://www.bfpug.com.br/Artigos/Dekkers-PontosDeFuncaoE Medidas.htm>. Acesso em 30/08/2001.

FENTON, Norman E.; PFLEEGER, Shari Lawrence. **Software metrics**: a rigorous and pratical approach. Boston: PWS Publishing, 1997.

HAZAN, Cláudia. **Análise de pontos por função**: uma abordagem gerencial. In: CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO /

JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, 20.19., 2000, Curitiba. **Anais...** Curitiba: Champagnat, 2000. p.287-326.

_____. Medição da qualidade e produtividade em software. In: WEBER, Kival Chaves; ROCHA, Ana Regina Cavalcanti da; NASCIMENTO, Célia Joseli do. **Qualidade e produtividade em software**. São Paulo: Makron Books, 2001.

HERRON, David; GARMUS, David. Estimating software earlier and more accurately. **Methods & Tools**, Vevey, v. 7, n. 3, p. 2-7, Fall/1999. Disponível em <http://www.martinig.ch/mt/dmt0399.pdf>. Acesso em 24/04/2001.

JONES, Capers. **Estimating software costs**. New York: McGraw-Hill, 1998.

_____. **What are function points?**. (s.l.), 1997. Disponível em <http://www.spr.com/library/0funcmet.htm>. Acesso em 01/03/2001.

KADODA, Gada; CARTWRIGHT, Michelle; CHEN, Liguang; SHEPPERD, Martin. **Experiences using case-based reasoning to predict software project effort**. Poole, March/2000. Disponível em <http://dec.bournemouth.ac.uk/ESERG/mshepperd/EASE00. pdf>. Acesso em 04/05/2001.

KAN, Stephen H. **Metrics and models in software quality engineering**. Boston: Addison-Wesley, 1995.

LONGSTREET Consulting, Inc. **A utilidade dos pontos de função**. (s.l.), 2000. Disponível em <http://www.bfpug.com.br/Artigos/MuitosUsos.htm>. Acesso em 30/08/2001.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. **Técnicas de pesquisa**. São Paulo: Atlas, 1999.

PARK, Robert E. **Software size measurement**: a framework for counting source statements. Pittsburgh, September/1992. {Technical Report} Disponível em <http://www.sei.cmu.edu/ publications/documents/92.reports/92.tr.020.html>. Acesso em 01/03/2001.

PAULA Filho, Wilson de Pádua. **Engenharia de software**: fundamentos, métodos e padrões. Rio de Janeiro: LTC, 2001.

PEREZ Jr, José Hernandez; OLIVEIRA, Luís Martins de; COSTA, Rogério Guedes. **Gestão estratégica de custos**. São Paulo: Atlas, 1999.

PETERS, James F.; PEDRYCZ, Witold. **Engenharia de software**. Rio de Janeiro: Campus, 2001.

PETERS, Kathleen. Software project estimation. **Methods & Tools**, Vevey, v. 8, n. 2, p. 2-15, Summer/2000. Disponível em <http://www.martinig.ch/mt/dmt0200.pdf>. Acesso em 18/10/2000.

PRESSMAN, Roger S. **Engenharia de software**. São Paulo: Makron Books, 1995.

REZENDE, Denis Alcides. **Engenharia de software e sistemas de informações**. Rio de Janeiro> Brasport, 1999.

ROETZHEIM, William. Estimating software costs. **SD Magazine**, (s.l.), October/2000. Disponível em <http://www.sdmagazine.com/articles/2000/0010/0010d/0010d.htm>. Acesso em 16/10/2000.

_____. Project cost adjustments. **SD Magazine**, (s.l.), November/2000. Disponível em <http://www.sdmagazine.com/articles/2000/0011/0011g/0011g.htm>. Acesso em 14/03/2001.

SAKURAI, Michiharu. **Gerenciamento integrado de custos**. São Paulo: Atlas, 1997.

SANTOS, Flavia Cerqueira et al. **Implantação da métrica de análise de pontos de função segundo uma abordagem de aprendizagem organizacional**. In: CONFERÊNCIA INTERNACIONAL DE TECNOLOGIA DE SOFTWARE, 12., 2001, Curitiba. **Anais...** Curitiba: CITS, 2001.

SHEPPERD, Martin; SCHOFIELD, Chris; KITCHENHAM, Barbara. Effort estimation using analogy. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 18., 1996, Berlin. **Anais...** Berlin: IEEE Computer Press, 1996. Disponível em <http://xanadu.bournemouth.ac.uk/ComputingResearch/ChrisSchofield/.../ICSEanalogy.htm>. Acesso em 04/05/2001.

SCHOFIELD, Chris; SHEPPERD, Martin. **Software support for cost estimation by analogy**. Rolduc, 1995. Disponível em <http://dec.bournemouth.ac.uk/ESERG/ANGEL/ ESCOM95.html>. Acesso em 04/05/2001.

_____. **Effort estimation by analogy**: a case study. Wilmslow, 1996. Disponível em <http://dec.bournemouth.ac.uk/ESERG/ANGEL/ ESCOM96.html>. Acesso em 04/05/2001.

SPR Software Productivity Research, Inc. **What are feature points?** (s.l.), 2002. Disponível em <http://spr.com/products/feature.htm>. Acesso em 25/03/2002.

_____. **What are function points?** (s.l.), 2002. Disponível em <http://spr.com/products/feature.htm>. Acesso em 25/03/2002.

TRINDADE, André Luiz. **Métricas de software**. (s.l.), 2000. Disponível em <http://metricas.tw.eng.br>. Acesso em 19/04/2001.

WEBER, Kival Chaves; ROCHA, Ana Regina Cavalcanti da; NASCIMENTO, Célia Joseli do. **Qualidade e produtividade em software**. São Paulo: Makron Books, 2001.

WIEGERS, Karl. Stop promising miracles. **SD Magazine**, (s.l.), February/2000. Disponível em <http://www.sdmagazine.com/articles/2000/0002/0002e/0002e.htm>. Acesso em 14/03/2001.