MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-15144-TDI/1276

UMA METODOLOGIA BASEADA EM MODELOS ESTATÍSTICOS E REDES NEURAIS PARA A ESTIMATIVA DE ESFORÇO DE DESENVOLVIMENTO DE SOFTWARE

Íris Fabiana de Barcelos Tronto

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. José Demisio Simões da Silva e Nilson Sant'Anna, aprovada em 5 de setembro de 2007.

INPE
São José dos Campos
2008

Publicado por: **esta página é responsabilidade do SID**

Instituto Nacional de Pesquisas Espaciais (INPE)
Gabinete do Diretor – (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 – CEP 12.245-970
São José dos Campos – SP – Brasil
Tel.: (012) 3945-6911
Fax: (012) 3945-6919
E-mail: pubtc@sid.inpe.br

Solicita-se intercâmbio
We ask for exchange

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-15144-TDI/1276

UMA METODOLOGIA BASEADA EM MODELOS ESTATÍSTICOS E REDES NEURAIS PARA A ESTIMATIVA DE ESFORÇO DE DESENVOLVIMENTO DE SOFTWARE

Íris Fabiana de Barcelos Tronto

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. José Demisio Simões da Silva e Nilson Sant'Anna, aprovada em 5 de setembro de 2007.

INPE
São José dos Campos
2008

681.3.06

Tronto, I. F. B.
Uma metodologia baseada em modelos estatísticos e redes neurais para a estimativa de esforço de desenvolvimento de software / Íris Fabiana de Barcelos Tronto. - São José dos Campos: INPE, 2007.
159 p. ; (INPE-15144-TDI/1276)

1. Estimativa de esforço de software. 2. Redes neurais artificiais. 3. Análise de regressão.
4. Análise de variância. 5. Análise de resíduos.
I. Título.

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de Doutor(a) em

Computação Aplicada

Dr. Solon Venâncio de Carvalho

Presidente / INPE / SJCampos - SP

Dr. José Demisio Simões da Silva

Orientador(a) / INPE / SJCampos - SP

Dr. Nilson Sant'Anna

Orientador(a) / INPE / SJCampos - SP

Dr. Acioli Antonio de Olivo

Membro da Banca / INPE / SJCampos - SP

Dr. Jaelson Freire Brelaz de Castro

Convidado(a) / UFPe / Recife - PE

Dra. Sandra Camargo Pinto Ferraz Fabbri

Convidado(a) / UFSCAR / São Carlos - SP

Aluno (a): Iris Fabiana de Barcelos Tronto

São José dos Campos, 05 de Setembro de 2007

A Nossa Senhora Aparecida, que esteve presente em todos os momentos durante a realização deste trabalho de doutorado iluminando, dando força, discernimento, compreensão e coragem, a mim e aos meus orientadores.

AGRADECIMENTOS

Agradeço ao Professor José Demísio e ao Professor Nilson pela ótima orientação do trabalho, pela paciência e amizade dedicada nestes anos de trabalho.

Agradeço à Professora Rosely Sanches, do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo – ICMC/USP, por sua colaboração, com discussões e opiniões tão importantes para a condução deste trabalho.

Ao Dr. Acioli Antônio de Olivo, pesquisador do Instituto Nacional de Pesquisas Espaciais, pelas discussões e ajuda em análise estatística, as quais foram fundamentais na condução deste trabalho.

Ao Instituto Nacional de Pesquisas Espaciais – INPE e ao Laboratório Associado de Matemática e Computação Aplicada – LAC, por terem permitido a execução deste trabalho.

Ao Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS) de l'Université Blaise Pascal, por ter me acolhido e disponibilizado a infra-estrutura e pessoal necessário para a condução de um estágio de doutorado no período de 01 de abril a 31 de agosto de 2005.

Aos Professores Michael Schneider e Jean Marc Petit pela colaboração e apoio na realização do estágio de doutorado na Universidade Blaise Pascal.

À Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, CAPES, pela bolsa e auxílios concedidos.

Agradeço a meu marido Jairo Tronto, pelo apoio, compreensão, companheirismo, amor e carinho que sempre teve comigo.

A meu pai José Gonzaga de Barcelos e minha mãe Zamita de Oliveira Barcelos pelo apoio e incentivos irrestritos dedicados em todos os momentos de minha vida.

A meus irmãos José Geraldo e Wesli; minhas irmãs Maria, Sueli e Márcia; meus cunhados e cunhadas: Marques, Helvécio, Flávio, Berenice e Ana Paula; meus sobrinhos e sobrinhas: Vitor, João Pedro, Carolina, Rafael, Julia, Lucas e Edson pela amizade e carinho.

A meu sogro Jayme, minha sogra Dalva, e meu cunhado Juliano pela amizade e consideração. A todos os parentes.

Ao Paulinho e à Luciana do grupo de desenvolvimento de software do Departamento de Sistemas de Solos do Instituto Nacional de Pesquisas Espaciais – DSS/ INPE, pela grande contribuição com informações importantes e dados de projetos desenvolvidos pelo INPE, para testar os modelos.

Agradeço ao Vanderlei e à Danielle, da Companhia de Desenvolvimento de Software do Estado do Paraná – CELEPAR, pela colaboração com dados de projetos para testar os modelos.

Ao amigo Orlando de Andrade Figueiredo pela ajuda com opinião e discussão do tema.

Ao Professor Ubiraci e ao Professor Fabio, da Faculdade de Filosofia Ciências e Letras de Ribeirão Preto – FFCLRP/ USP, pela consultoria prestada em estatística.

Aos meus colegas e amigos do LAI/ ITA: Daniela, André, Alexandre, Eveline, Jeane, Inaldo, Robson, pelo apoio, compreensão e paciência.

Agradeço aos meus colegas: Dawilmar, Daniela, Giani, Elias, Roberta, Adriana, Andréia e Bruno, pela amizade.

Agradeço aos servidores: Cristina, Neuza, Amanda, Márcio, Carol, Eunice, Silvia, Yolanda e todos os amigos da gráfica do INPE, pela atenção, amizade, compreensão e pelo pronto atendimento.

A todos que de alguma forma contribuíram para a realização desse trabalho.

RESUMO

Estimativa de esforço de software é uma parte importante do desenvolvimento de software e fornece um guia essencial para a análise de viabilidade, licitação, orçamento, planejamento e acompanhamento de projeto. As conseqüências de estimativas imprecisas podem resultar em perdas significativas ou mesmo em perda de contratos. Em geral as estimativas de projetos são excedidas, principalmente porque as estimativas são muito otimistas. Nesta tese, o principal objetivo é apresentar uma metodologia baseada em métodos estatísticos e de redes neurais para realizar estimativas de esforço mais precisas e de forma mais simples. Esta pesquisa contribui para a redução de erros de estimativa de projeto de desenvolvimento de software, permitindo que o público interessado tenha um melhor entendimento das várias classes de modelos e técnicas de estimativa de esforço de software e da expressividade das variáveis de projeto disponíveis. São utilizadas redes neurais artificiais, técnicas de raciocínio baseado em casos, modelos baseados em regressão, e técnicas para integrar análise de resíduos, análise de variância e modelos baseados em regressão. Vários estudos de casos foram conduzidos para validar os diferentes métodos. Os resultados indicam que os métodos propostos apresentam resultados realistas, para os dados das empresas disponíveis na base de dados, e que o uso de redes neurais implica em um processo simples de calibração de modelos locais. Entretanto, observa-se que as técnicas são dependentes dos dados disponíveis, exigindo a re-calibração dos modelos em função do surgimento de novas tecnologias para o desenvolvimento de software.

A METHODOLOGY BASED ON STATISTICS MODELS AND NEURAL NETWORKS TO THE SOFTWARE PROJECT DEVOLOPMENT EFFORT ESTIMATION

ABSTRACT

Software effort estimation is an important part of software development work and provides essential input to project feasibility analyses, bidding, budgeting and planning. The consequences of inaccurate estimates can be severe. Optimistic estimates may cause significant losses while the pessimistic estimates may lead to loss of exiting and future contracts. Unfortunately, it is common for software development projects to overrun their effort estimates, typically because the estimates are too optimistic. This thesis presents a methodology based on statistical and neural networks methods to provide more accurate effort estimates in a simpler way. The goal of this research is to contribute to reduce estimation error in software development projects by better understanding the different software effort estimation models and techniques that include: artificial neural networks, case-based reasoning techniques, regression-based models, and techniques for integrating analysis of residuals, analysis of variance and regression-based models. Several case studies have been conducted. The results show all the proposed models lead to realistic estimations, however, neural networks based models emerge as a very easy tool for local models calibration processes due to its simpler implementation. The case studies show all the models are sensitive to the available data, thus requiring recalibration processes every since new project data area gathered to the database.

SUMÁRIO

Pág.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ACAP	Capacidade do analista
ADJKDSI	Número de linhas de código
AEXP	Experiência com aplicações
ANN	Redes neurais artificiais (Artificial Neural Networks)
APF	Análise de Pontos por Função
COCOMO	Modelo de estimativa de custo (Constructive Cost Model)
CMMI	Modelo integrado de maturidade e capacidade
CPLX	Complexidade do produto
DATA	Tamanho da base de dados
GQM	Objetivo, questão, métrica (Goal/Question/Metric)
INPE	Instituto Nacional de Pesquisas Espaciais
KPA	Área chave do processo
LANG	Linguagem de Programação
LEXP	Experiência com linguagem de programação
LOC	Linhas de Código
MMACT	Quantidade de esforço para desenvolvimento de software
MMRE	Magnitude média do erro relativo
MODP	Uso de práticas modernas de programação
PCAP	Capacidade do programador
PMBOK	Corpo de conhecimento para gerenciamento de projeto
PMI	Instituto de gerenciamento de projeto
RBC	Raciocínio Baseado em Casos
RELY	Confiabilidade requerida do software
RNA	Redes Neurais Artificiais
RVOL	Volatilidade dos requisitos
SCHED	Cronograma de desenvolvimento requerido
SEI	Instituto de Engenharia de Software
SPI	Melhoria de processo de software
STOR	Restrição de armazenamento principal

TIME	Restrição de tempo de execução
TOOL	Uso de ferramentas de software
TURN	Tempo de execução da máquina
TYPE	Tipo de aplicação
UCP	Pontos por caso de uso
VEXP	Experiência com máquina virtual
VIRT	Volatilidade da máquina virtual

1 INTRODUÇÃO

1.1 Contexto

O computador faz parte da vida diária das pessoas na comunicação com amigos, na realização de compras pela Internet, na realização de tarefas importantes relacionadas à previsão de tempo e de clima de forma mais segura e mais eficiente, e em inúmeras outras aplicações de trabalho e entretenimento. O crescente uso dos sistemas de computador em todos os segmentos da sociedade tem contribuído para o aumento da competitividade entre empresas produtoras e prestadoras de serviços de software. Assim, é preciso desenvolver software de qualidade, em tempo hábil e com baixo custo.

No entanto, nos últimos anos têm sido relatadas várias histórias de insucesso no desenvolvimento de software através de estudos de casos e experimentos documentados. Uma grande parte dos projetos ainda usa mais recursos do que aqueles que foram planejados, eles levam mais tempo para serem concluídos, fornecem menos funcionalidade e menos qualidade que o esperado. Muitos estudos relacionam tais falhas a um gerenciamento de projeto inadequado, apontando problemas de comunicação, alocação errada de habilidade da equipe, treinamento insuficiente e falta de habilidade de gerentes para predizer e ajustar o comportamento do projeto.

Um nível de qualidade e produtividade internacional pode ser conseguido através da gestão efetiva de seus processos de software, focalizando pessoas, produto, processo e projeto. Em relação a projeto, é preciso que haja planejamento e acompanhamento através de um conjunto de atividades, dentre as quais as estimativas de esforço são consideradas fundamentais, pelo fato de fornecerem um guia para as demais atividades (Agarval, 2001; Jones, 1986; Lai e Huang, 2003; Hasting e Sajeev, 2001; Briand e Wieczorek, 2002).

Através das estimativas de esforço, um gerente pode tomar decisões importantes, por exemplo, na fase inicial de um projeto ele pode decidir por desenvolver um software ou comprá-lo; durante todo o projeto ele pode estimar esforço para fazer o acompanhamento e avaliação dos riscos. A equipe de projeto deve ser capaz de produzir estimativas precisas para que a organização possa permanecer no mercado competitivo, pois o aumento inesperado no orçamento diminui a confiança nas estimativas e na equipe de desenvolvimento de software (Subramanian e Breslawski, 1995). A precisão é determinada medindo-se a diferença entre o esforço real e o estimado.

Diante da necessidade de se realizar estimativas de esforço de software consistentes e precisas muitas pesquisas têm sido conduzidas a fim de construir, avaliar e recomendar técnicas de predição (Boehm, 1981; Albrecht e Gaffney, 1983; Shepperd e Schofield, 1997; Zhong et al, 2004; Sentas et al, 2005; Bisio e Malabocchia, 1995; Myrtveit et al, 2005; Samson et al, 1997), que se enquadram em três categorias gerais:

1) Julgamento de Especialistas – tem sido a técnica mais amplamente utilizada, entretanto, os meios de se derivar uma estimativa não são explícitos e conseqüentemente não repetíveis. A opinião do especialista, embora seja sempre difícil de se quantificar, pode ser uma ferramenta de estimativa efetiva como um fator de ajuste para modelos algorítmicos (Gray et al. 1999).

2) Modelo Algorítmico – estes modelos são os mais populares na literatura e precisam ser calibrados ou ajustados a circunstâncias locais; são desenvolvidos usando dados de companhias que podem ser bastante diferentes daquela interessada nas estimativas; e tentam representar o relacionamento entre esforço e uma ou mais características. Geralmente, o principal multiplicador de esforço utilizado em tais modelos é o tamanho do software (por exemplo, o número de linhas de código fonte, o número de pontos por função, o número de pontos por

caso de uso, dentre outros). Os modelos mais conhecidos e utilizados são COCOMO (Boehm, 1981), modelos baseados em Pontos por Função (Albrecht e Gaffney, 1983) e o modelo SLIM (Putnam,1978).

3) Aprendizagem de Máquina – na última década, essas técnicas têm sido utilizadas como complemento ou alternativa para as duas categorias anteriores. Exemplos incluem modelos de lógica nebulosa (Kumar et al, 1994), árvores de regressão (Selby e Porter, 1998), redes neurais artificiais (Srinivasan e Fisher, 1995) e raciocínio baseado em casos (Shepperd et al, 1996). Um resumo destas técnicas é apresentado por Gray e MacDonell, (1997).

Apesar de existirem várias ferramentas e abordagens para estimativa de esforço de software, realizar predições precisas ainda representa um problema para as empresas públicas e para a indústria. A literatura contém informações abundantes sobre projetos que excedem o esforço e o custo estimado (Heemstra, 1992; Gibbs, 1994; Ingram, 1995; General Accounting Office, 1992; 1994). Heemstra (1992) lista vinte e nove modelos de custo que têm sido desenvolvidos desde 1966. Esses modelos são baseados no tamanho da aplicação e produzem resultados que variam muito. Nos últimos anos, têm sido conduzidos vários estudos a fim de comparar o poder de predição de diferentes técnicas (Gray e MacDonell, 1997; Briand et al, 2000; Jeffery et al, 2001; Shepperd et al, 1996; Angelis e Stamelos, 2000; Finnie et al, 1997), mas, ainda não se chegou a uma convergência sobre qual técnica se deve utilizar.

Para alguns autores uma forma de se obter vantagem dessa diversidade seria utilizar vários modelos e comparar os resultados (Goodman, 1992; Mukhopadhyay e Sunder, 1992; Kitchenham e Taylor, 1985). No entanto, nenhum estudo demonstra que estas sugestões melhoram a precisão das estimativas e poucas organizações fazem uso desses modelos (Kemerer, 1987; Heemstra, 1992; Abdel-Hamid, 1993). De acordo com uma pesquisa relatada pelo Ministério da Ciência e Tecnologia, no Brasil, em 2001, apenas

45,7% das empresas realizam estimativa de esforço de software (MCT, 2001). Não há um estudo específico que identifique as causas do baixo índice de realização de estimativas de esforço, mas o nível de confiabilidade dos modelos pode ser uma das possíveis causas.

De acordo com entrevistas realizadas com representantes do grupo de desenvolvimento de software da Divisão de Desenvolvimento de Sistemas de Solo do Instituto Nacional de Pesquisas Espaciais – DSS/INPE, a complexidade envolvida no ajuste dos modelos à realidade da organização é um dos fatores responsáveis pela não utilização de modelos de estimativa como COnstructive COst Model (COCOMO), modelos baseados em análise de pontos por função, modelos baseados em análise de casos de uso, dentre outros disponíveis no mercado e na literatura. Essa dificuldade se deve muitas vezes à falta de dados históricos e à falta de tempo suficiente para realizar os ajustes, uma vez que a complexidade inerente a estes modelos requer um tempo considerável para sua calibração. Além disso, o nível de precisão que se pode alcançar através desses modelos, não justifica o tempo despendido para sua utilização.

Assim, grande parte das empresas continua realizando suas estimativas de forma ad-hoc, baseando-se puramente na opinião de especialistas. Para os entrevistados, essa estimativa baseada em experiência também representa um problema, pois a rotatividade da equipe é uma característica presente no processo de desenvolvimento de software e nem sempre os membros da equipe têm experiência suficiente em estimativas de esforço.

O exposto nos parágrafos anteriores mostra a necessidade de se ter uma abordagem alternativa para realizar estimativas mais confiáveis e de forma mais simples.

1.2 Objetivos

O principal objetivo dessa tese é a concepção de uma metodologia para a estimativa de esforço de desenvolvimento de software baseada em modelos estatísticos e de Redes Neurais Artificiais - RNA. Desse modo acredita-se estar contribuindo para a realização de estimativas de forma mais simples e mais precisas e, conseqüentemente, colaborar para a melhoria do processo de desenvolvimento de software. Através desta metodologia espera-se contribuir para que uma organização possa realizar a análise de seus próprios dados e calibrar seu próprio modelo de estimativa de esforço ou ainda, se julgar necessário, utilizar um dos modelos propostos.

Para este propósito, várias abordagens foram experimentadas incluindo as seguintes técnicas: raciocínio baseado em casos (Barcelos Tronto et al., 2006a); análise de regressão simples (Barcelos Tronto et al., 2007a); análise de regressão múltipla (Barcelos Tronto et al., 2006b); redes neurais artificiais (Barcelos Tronto et al., 2007b); análise de variância, análise de regressão e análise de resíduo. Após realizar os vários experimentos utilizando estas técnicas verificou-se que para determinados conjuntos de dados, uma técnica pode ser melhor que outra, embora na maioria dos casos os resultados obtidos com a aplicação de redes neurais artificiais apontam essa técnica como a mais apropriada.

No contexto de estimativa de esforço surge uma questão: uma vez que a organização coletou os dados, porque a própria empresa não os analisa? A metodologia proposta é constituída de um conjunto de passos que permite uma pessoa realizar estimativa utilizando uma ou mais dessas técnicas sobre os dados da própria organização. Esse conjunto de passos é constituído da definição dos dados, do pré-processamento dos dados, da geração do modelo e análise de precisão das estimativas. Através dessa metodologia, uma pessoa poderá realizar estimativa ajustando os modelos aos dados da própria organização (caso esta tenha uma base de dados de projetos desenvolvidos).

Verificou-se que os modelos de redes neurais, além de fornecerem estimativas com um bom nível de precisão, podem ser calibrados através de um procedimento bastante simples, o que possibilita maior facilidade e rapidez no processo de estimativa.

Outra importante contribuição deste trabalho é fornecer subsídio a uma organização que ainda não possui uma base de dados de histórico de projetos e que deseja implementar um programa de medidas para dar apoio ao processo de estimativa de esforço de software. O conjunto de variáveis presentes nos modelos obtidos através dos experimentos representa um guia sobre os fatores que podem influenciar no esforço de desenvolvimento de software. Assim, a organização poderá coletar medidas para estes fatores (métricas) e implementar um programa de medição para apoiar o processo de estimativa de esforço de projetos futuros, baseando-se nos seus dados históricos. Assim, ela estará contribuindo para a melhoria do processo de software.

1.3 Organização do trabalho

Esta tese está organizada em cinco capítulos. O Capítulo 1 apresenta uma breve introdução sobre estimativa de projetos de software, assim como a hipótese de trabalho e os objetivos a serem atingidos. No Capítulo 2 é apresentada a caracterização da área de estudo. O Capítulo 3 contém a fundamentação teórica na qual a pesquisa foi baseada, com uma descrição do estado da arte em estimativas de esforço de software e das técnicas utilizadas no desenvolvimento desta tese. Os resultados obtidos e a discussão dos mesmos estão apresentados no Capítulo 4. O Capítulo 5 contém uma discussão da significância dos resultados obtidos e sobre perspectivas de trabalhos futuros.

2 MÉTRICAS E ESTIMATIVAS EM GESTÃO DE PROJETO DE SOFTWARE

2.1 Considerações iniciais

Este capítulo trata da base teórica sobre métricas e estimativas de software através de uma revisão destes conceitos no contexto de gerenciamento de projeto de software. Discute-se a importância de se ter uma gestão de projeto efetiva em desenvolvimento de software e como as métricas e estimativas, sobretudo a estimativa de esforço de projeto, se relacionam à qualidade do processo de gestão de software. Para isso, inicia-se com os conceitos relacionados à gestão de projetos de software (que envolve métricas, planejamento e acompanhamento de projetos de software) e finaliza-se com a caracterização das diferentes metodologias para se realizar estimativas de desenvolvimento.

2.2 Gerenciamento de projetos

O processo de construção de software é uma atividade complexa que afeta as características essenciais do produto final, como a qualidade e o custo. No modelo ISO/ IEC 15504 (1998), o termo processo de software é definido como um conjunto de processos utilizados por uma organização ou projeto para planejar, gerenciar, executar, monitorar, controlar e melhorar as atividades relacionadas à construção de software. Na Norma ISO IEC/12207 (ISO, 1997) os processos que envolvem o ciclo de vida do software são agrupados nas classes fundamentais, de apoio e organizacionais, conforme Figura 2.1. Cada processo é definido em termos de suas próprias atividades e cada atividade é adicionalmente definida em termos de suas tarefas. Dentre os processos organizacionais uma atenção especial é dada aos processos de gerência e de melhoria de processos.

O gerenciamento de projeto é um processo do ciclo de vida de software, que consiste de um conjunto de procedimentos que podem ser usados por gerentes em qualquer tipo de desenvolvimento de software. Por exemplo, organizações

que têm como meta terminar seus projetos dentro do cronograma, atender seus orçamentos e manter a qualidade, devem implementar uma gestão mais eficaz do processo de desenvolvimento de software (Amescua et al., 2004). A gestão de projetos proporciona às empresas ferramentas poderosas que melhoram a habilidade da organização para planejar, organizar, executar e controlar as atividades de maneira a conseguir atingir os resultados esperados dentro do prazo e custo previstos, mesmo em projetos de grande complexidade.

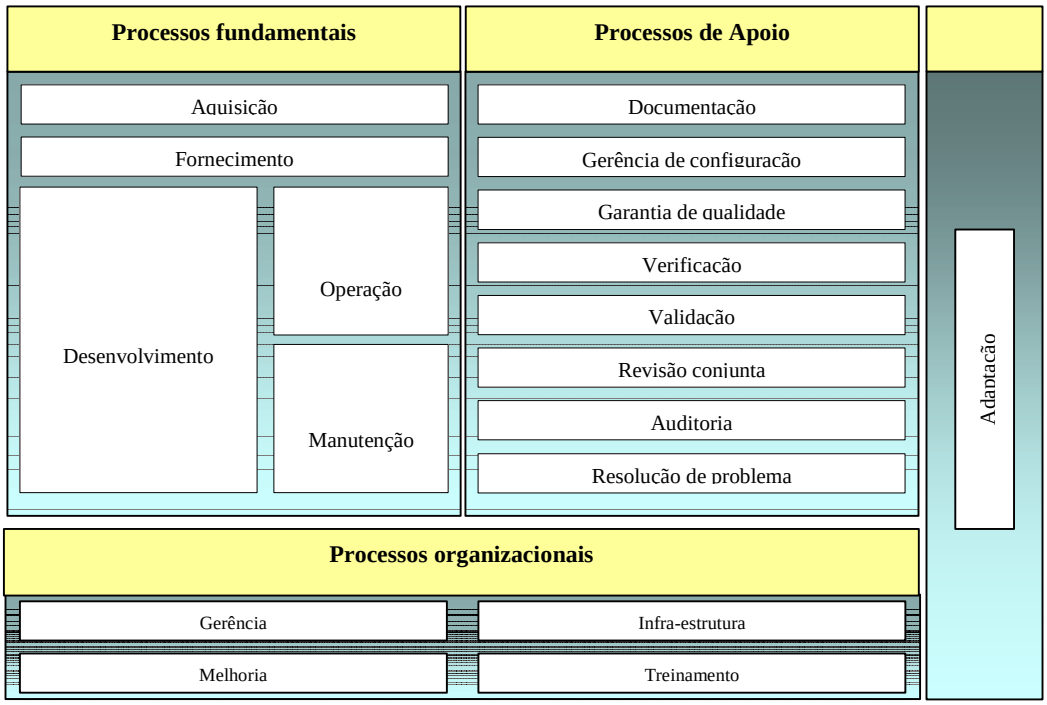

Figura 2. 1 - Ciclo de Vida de Desenvolvimento de Software.

FONTE: Machado (2001).

O Project Management Institute (PMI) é uma instituição dedicada ao progresso e disseminação das melhores práticas da atividade de Gestão de Projetos e define uma abordagem para os processos de gerenciamento de projetos através do livro Project Management Body of Knowledge (PMBOK) (PMI, 2000). Para o PMI, a gerência de projetos é a aplicação de conhecimentos, habilidades, ferramentas e técnicas para projetar atividades que visam atingir aos requisitos do projeto.

O PMBOK (PMI, 2000) descreve a gerência de projetos em termos de seus processos e de suas interações. Esses processos são agrupados em categorias, das quais destacam-se três: 1) Planejamento: define os objetivos e estabelece o melhor curso de ação para alcançar os objetivos que o projeto se compromete a atender (previsão de tamanho, esforços, recursos, prazos e riscos de um projeto de software); 2) Execução: coordena pessoas e recursos para realizar o plano; 3) Controle: monitora e mede regularmente o processo a fim de garantir que os objetivos do projeto sejam alcançados. Uma descrição sobre planejamento e acompanhamento é apresentada na Seção 2.2.1.

As categorias estabelecidas pelo PMI, não levam em consideração o tipo de projeto, ou seja, se o projeto em desenvolvimento é um projeto de software ou de outro tipo. Sant'Anna (2000) adota as bases estabelecidas pelo PMI, porém ajustando alguns processos e acrescentando outros para melhor se adequar às necessidades específicas de projeto de software. Dentre esses processos está o processo de gerenciamento de custo, em que uma das atividades é a gestão dos recursos.

No planejamento de recursos deve-se utilizar, dentre outras entradas, sempre que disponíveis, as informações históricas relativas ao esforço que foi requerido em projetos similares anteriores. Então, uma atividade fundamental deste processo é a estimativa de esforço, que utiliza a base de dados de históricos, que deve conter as informações de esforço de projetos passados disponíveis em arquivos de projetos, bases de dados comerciais ou são referentes ao conhecimento da equipe de projeto.

Nesse contexto de estimativa, o processo de medição é fundamental. Gerentes experientes são capazes de perceber alguns aspectos de um processo de desenvolvimento de software que um gerente novato não percebe. Esse conhecimento gerencial contido na base mental de um gerente especialista é um recurso de grande valor que, caso não seja medido e armazenado em uma base de dados de históricos, pode facilmente ser perdido, por várias razões.

Portanto, é importante utilizar técnicas para representar e proporcionar uma reutilização deste conhecimento, de forma que um gerente novato possa compartilhar as experiências dos especialistas, no planejamento e acompanhamento de projetos.

Esse conhecimento é registrado a partir de um programa de medição implantado na organização e armazenado em um repositório de experiência (Basili et al., 1994a), que aumenta à medida que projetos vão sendo terminados. Exemplos de métricas são: densidade de defeitos, tamanho da aplicação, quantidade de esforço, custos e medidas relacionadas ao cronograma de projeto, dentre muitas outras. Assim, pode-se dizer que o gerenciamento de projetos de software é uma atividade fortemente baseada em conhecimento.

2.2.1. Planejamento e acompanhamento de projetos

O gerenciamento eficaz de um projeto de software depende de um bom planejamento do projeto. O gerente deve prever os problemas que podem surgir e preparar soluções experimentais para esses problemas. Um plano traçado no início deve ser utilizado como guia para o projeto. Deve-se buscar desenvolver o plano inicial da melhor forma possível, porém ele deve evoluir à medida que o projeto é desenvolvido e melhores informações se tornem disponíveis. É preciso tentar prever todas as possíveis eventualidades no plano de projeto, para que as restrições e os marcos do projeto não tenham de ser renegociados a todo tempo ou a cada iteração de planejamento.

O planejamento é a função que determina, antecipadamente, que objetivos devem ser atingidos e o que se deve fazer para alcançá-los. O plano de projeto define os recursos disponíveis para o projeto, a estrutura analítica do trabalho e uma programação para realizar o trabalho. Mais especificamente, ele inclui: detalhamento do escopo; definição das atividades e da seqüência, estimativa da duração das atividades; desenvolvimento do cronograma; planejamento da

gerência de risco; planejamento dos recursos; e estimativas do esforço e dos custos.

Planejar é um esforço contínuo durante toda a vida do projeto. Estes processos estão sujeitos a freqüentes interações antes da complementação do plano. Nas fases iniciais do projeto, por exemplo, durante o levantamento de requisitos, ainda não se conhece completamente as características do produto de forma que se possa equacionar um conjunto complexo de elementos, entre eles: a quantidade de esforço (horas/homem, homem/mês, etc.); o tempo (em semanas, meses, anos, etc.); e os riscos. Assim, é preciso estimar. E as estimativas de tempo e esforço podem ser consideradas como as mais importantes.

Vários modelos de estimativa foram criados para fornecer métricas que permitam atender com menor margem de erro às necessidades de comunicação e informação do projeto, principalmente na fase de planejamento do projeto. Modelos como análise de pontos de função, pontos de caso de uso, COCOMO, modelos de redes neurais, modelos estatísticos e modelos de raciocínio baseado em casos permitem estimar o esforço de projetos de software. A manutenção de registros de outros projetos semelhantes, com a evolução das estimativas iniciais até a medição final, permite um acompanhamento nos vários estágios do desenvolvimento do produto.

O desempenho do projeto deve ser monitorado e medido regularmente para identificar as variações do plano. Estes desvios são analisados, dentro do processo de acompanhamento (ou controle), nas diversas áreas de conhecimento. Na medida que são identificados desvios significativos (aqueles que colocam em risco os objetivos do projeto), são realizados ajustes ao plano através da repetição dos processos de planejamento que sejam adequados àquele caso. Por exemplo, ultrapassar a data de término de uma atividade pode requerer ajustes nos recursos humanos, na necessidade ou não de horas-extras, ou no balanceamento entre o orçamento e os objetivos de prazo

do projeto. Acompanhar um projeto também inclui tomar ações corretivas, antecipando-se aos problemas.

O controle é uma das principais atividades envolvidas na gerência de projetos. Porém, não se consegue controlar o que não se pode medir. A falta de métricas de projeto prejudica de forma geral seu acompanhamento, uma vez que, apesar de o problema estar lá, ele não é percebido por aqueles que podem direcionar esforços em sua solução. O papel das métricas é permitir uma rápida identificação e correção dos problemas. A seguir é apresentada uma breve descrição sobre o uso de métricas no gerenciamento de software.

2.2.2. Métricas na gestão de software

Métricas são elementos chave no processo de gerenciamento de projeto, que ajudam a entender o processo técnico usado para desenvolver o produto. O processo é medido para avaliar as atividades no desenvolvimento do software (com a intenção de melhorar tais atividades) e para habilitar a organização a criar decisões estratégicas sobre como melhorar as atividades do processo.

Já o software é medido para se gerar práticas para aumentar sua qualidade, avaliar a produtividade das pessoas que o desenvolvem, determinar os benefícios derivados de novos métodos e as ferramentas de engenharia de software, formar uma base para as estimativas e ajudar na justificativa de aquisição de novas ferramentas ou de treinamentos adicionais. As métricas de produto correspondem a medidas como, esforço e custo para desenvolver, número de linhas de código (LOC), quantidade de erros, tamanho de memória, velocidade de execução, qualidade do produto, complexidade, eficiência, confiabilidade, manutenibilidade e funcionalidade.

As métricas de projeto são utilizadas para: a) avaliar o status do projeto de software; b) evitar atrasos no cronograma e delinear riscos de projeto; c) monitorar progresso durante o desenvolvimento do software e controlar a

qualidade do produto. Além disso, as métricas servem como base para se realizar estimativas. Por exemplo, o modelo de estimativa COCOMO (Boehm, 1981) utiliza o tamanho da aplicação, em número de linhas de código, para estimar o esforço requerido para o desenvolvimento de um projeto.

Alguns estudos indicam que a implementação de um programa de métricas ajuda na obtenção de melhores resultados do ponto de vista da gestão, quer seja em curto prazo (para um dado projeto) ou em longo prazo (projetos futuros), melhorando a estimativa, a produtividade e a qualidade. Com efeito, um número cada vez maior de organizações tem obtido resultados promissores com a implementação de programas de métricas. Estes indicadores positivos têm alertado a comunidade de gestores de projeto. Espera-se que, em médio prazo, também as empresas com projetos de menor dimensão possam vir a colher benefícios idênticos.

O resultado da medição tem o papel de permitir a comunicação efetiva não só entre os vários indivíduos e organizações envolvidas no projeto, mas também com os que possam ser afetados por ele. A penalidade pela falta de mecanismos de medição ou pela sua não-utilização freqüente é enfrentar problemas como o adiamento da conclusão do projeto, orçamento estourado, dentre outros.

O adiamento da conclusão do projeto interfere negativamente na motivação da equipe, que tem suas expectativas frustradas. A qualidade interna do produto é prejudicada e, na maioria das vezes, afeta também a qualidade externa. O papel das métricas é permitir uma rápida identificação e correção de problemas. Assim, a falta de métricas de projeto prejudica de forma geral seu acompanhamento, pois o gerente não terá conhecimento do problema.

O desvio da situação atual em relação ao que foi projetado é um dos problemas freqüentemente encontrado no gerenciamento de projetos e é percebido através da análise das métricas definidas e coletadas. Essa análise

permite que aspectos como esforço, custo, cronograma, qualidade, riscos ou escopo sejam monitorados. Para um gerente é mais fácil justificar e defender as decisões tomadas quando se utiliza métricas, pois as decisões são tomadas baseando-se não somente em sua experiência individual, mas também na avaliação de indicadores que refletem uma tendência de comportamento futuro. Essa tendência deve ser derivada de experiências passadas no projeto e de experiências semelhantes de outros projetos de dentro e de fora da organização.

A medição é importante também no contexto da terceirização e gestão de contratos. Tanto no corpo de conhecimento em gerência de projetos (PMBOK) do *Project Management Institut* (PMI, 2000) – quanto no *Capability Maturity Model Integration* (CMMI, 2002) do *Software Engineering Institute* (SEI) a aplicação de métricas é um aspecto determinante no relacionamento entre a empresa que contrata e a empresa contratada. Através da análise de indicadores é possível comparar as alternativas disponíveis e decidir se é mais vantajoso desenvolver o produto internamente ou a contratação de um produto específico no mercado. Por exemplo, com a análise de pontos de função é possível dimensionar a funcionalidade solicitada por uma organização e, baseando-se na quantidade de pontos de função, em indicadores de esforço, de produtividade e de custo é possível extrapolar essas informações, que de outra forma seriam inacessíveis ou de difícil apuração ou justificativa.

Além disso, há uma urgência das empresas por mecanismos que permitam o aumento dos níveis de qualidade de seus processos e produtos. No ramo de desenvolvimento e manutenção de sistemas, essa urgência se materializa no crescente número de empresas de desenvolvimento e manutenção de sistemas interessadas na obtenção de certificados de qualidade ISO/CMMI ou de melhorar seus processos através do empreendimento de iniciativas de melhoria de processos de software (*Software Process Improvement*-SPI*).* Pode-se definir SPI como o procedimento sistemático para melhorar a performance de um sistema composto por um conjunto de processos

existentes, pela modificação desses processos existentes ou a atualização com novos processos, objetivando corrigir ou evitar problemas identificados no sistema anterior. Existem vários modelos que visam orientar a condução dessas ações, como por exemplo:

SEI/ CMMI – Capability Maturity Model Integration (CMMI, 2002).

ISO/IEC 15504 – Information Technology - Software Process Assessment (ISO, 1998)

Independentemente do modelo, todos procuram tornar a atividade de desenvolvimento e manutenção de sistemas mais previsíveis, ou seja, que as estimativas sobre o comportamento das principais variáveis envolvidas, esforço, custo, tempo e qualidade tenham maior probabilidade de se confirmar nas suas respectivas medições.

O modelo do Software Engineering Institute (SEI) é o mais difundido e discutido no Brasil, sendo atualmente o que representa a maior tendência em termos de iniciativa de SPI. Ele é estruturado em cinco níveis de maturidade, constituídos de um conjunto de atividades relacionadas, que quando executadas em conjunto, atingem um conjunto de objetivos considerados relevantes para a melhoria da capacidade dos processos. Esse conjunto de atividades é chamado *Process Area* (PA) (anteriormente chamado área-chave de processo *Key Process Area* (KPA). O foco do segundo nível de maturidade (chamando *Managed)* está em gerência de projetos. Uma das PAs desse nível de maturidade é “**Medição e Análise”.**

O propósito da PA, denominada “Medição e Análise”, é desenvolver e manter a capacidade de medição a fim de fornecer as informações gerenciais importantes para a organização. As práticas específicas relacionadas aos objetivos específicos dessa PA são: estabelecer objetivos de medição, especificar medidas, especificar procedimentos de coleta de dados e armazenamento, especificar procedimentos de análise, coletar dados de medição, analisar dados de medição, armazenar dados e resultados e comunicar resultados.

Enfim, não se tem dúvida de que medir é preciso, mas muitas vezes tem-se a dúvida sobre o que medir. Para responder a esse questionamento, Vazquez et al. (2006) sugerem que devem ser medidas características, propriedades e eventos, cujas quantificações sejam relevantes para responder a objetivos definidos. Basili e Rombach (1988) apresentam uma abordagem denominada Goal/Question/Metric (GQM), bastante ilustrativa e útil no processo de identificação dessas características. Essa abordagem estabelece que, para cada um dos objetivos que se deseja acompanhar é possível estabelecer um conjunto de perguntas que verifique o seu cumprimento; para muitas dessas perguntas é possível identificar uma métrica que possa quantificar a resposta.

O método GQM é uma tecnologia que tem sido aplicada em várias companhias (Rombach, 1990; Basili et al., 1994b) para ajudar a definir e implementar metas de software mensuráveis e operacionais de forma a facilitar sistematicamente o planejamento e a condução de medidas. Nick et al. (1999) mostram que GQM atende os requisitos para se realizar boas medidas e é útil na avaliação de um sistema baseado em conhecimento, por exemplo, para uma base de experiência. Briand et al. (2002) emprega o paradigma GQM e um conjunto de hipóteses empíricas para definir medidas válidas para os atributos de interesse.

Existem vários modelos que apóiam o processo de medição, mas independentemente do modelo utilizado para implementar um programa de métricas, a coleta de métricas é apenas um dos requisitos envolvidos. A combinação dessas medidas coletadas é o que gera visibilidade quanto a algum aspecto ou conceito relevante ao desenvolvimento ou manutenção de sistemas. Esses indicadores, baseados em medidas diversas permitem o monitoramento dos vários processos envolvidos.

Para responder aos objetivos definidos, deve-se medir características, propriedades e eventos, cujas quantificações sejam relevantes. A medição não só deve ser feita, mas também pode ser estimada. Ao explorar a motivação e

os objetos de medição, chega-se à conclusão de que o esforço é uma das propriedades que deve ser medida e estimada (Vazquez et al., 2003).

Em 1981, Dr. Barry Boehm (1981) publicou a primeira versão do modelo COCOMO para estimativa de esforço e de custo e em 2000 (Boehm et al., 2000) publicou uma nova versão do modelo COCOMO de forma a atender as abordagens modernas de desenvolvimento de software. No contexto de estimativa de esforço, uma importante métrica é o tamanho do software, que pode ser medido utilizando várias métricas, dentre as quais destacam-se o número de linhas de código (LOC - Line of Code) e o número de pontos de função (Albrecht, 1979). Análise de Pontos por Função é utilizada para medir sistemas do ponto de vista de seus usuários pela quantificação da funcionalidade solicitada pelo usuário.

2.3 Estimativa de software

A estimativa de software consiste em examinar atributos de um conjunto de entidades (produtos, processos e recursos) e com base nestes valores de atributos fixar valores para atributos não conhecidos que se deseja quantificar. O termo predição é às vezes usado quando uma estimativa é feita para predizer o resultado futuro de um valor de um atributo. Um exemplo típico de uma tarefa de “estimação” é o uso de atributos que caracterizam um projeto para estimar (predizer) o esforço (em termos de homens /mês) para desenvolvê-lo.

Para atender às novas demandas de desenvolvimento, manutenção ou mesmo de aquisição e customização de um software, a equipe de projeto se depara com dois questionamentos que estão sempre presentes: 1) Quanto tempo é necessário para se concluir o projeto; 2) Qual o esforço (em homens /hora) que necessário e quanto o projeto vai custar para a organização. De acordo com Addison e Vallabh (2002), o cronograma e o esforço são vistos pelos profissionais de software como sendo os fatores de risco mais problemáticos.

Os fatores relacionados às particularidades dos requisitos de um projeto de software, da equipe responsável e da tecnologia empregada muitas vezes dificultam a obtenção de respostas confiáveis a estes questionamentos.

Para Vazquez et al. (2003) estas dificuldades podem ser evidenciadas ao tentar responder um conjunto de questões como:

a) Os requisitos traduzem com fidelidade as necessidades do negócio dos usuários? Já se encontram suficientemente estabilizados? Refletem características de software transacionais de baixa complexidade ou possuem atributos que exigem conhecimentos específicos, tais como alta performance, matemática complexa, processamento distribuído?

b) A equipe de desenvolvimento possui conhecimento na área de negócio que será atendida pelo projeto de software? Possui experiência na utilização das ferramentas necessárias à conclusão do projeto? Possui todos os perfis necessários impostos pelas características do projeto? Existem conflitos internos à equipe que precisam ser solucionados? É possível identificar uma liderança entre os integrantes da equipe? Qual o grau de motivação da equipe mediante o projeto?

c) A tecnologia já faz parte da cultura da organização? Pode ser facilmente absorvida por novos integrantes da equipe de projeto? Existe material de apoio suficiente para o aprendizado da tecnologia? Sua aplicação exige pessoal com algum tipo de especialização? Suporta a implementação de todas as características do software? Atende inclusive aos requisitos não-funcionais do projeto?

Diante dessas e outras peculiaridades inerentes a cada projeto, só se pode determinar com exatidão o esforço, o custo final e a data de conclusão do projeto quando ele já está finalizado. Porém, devem ser realizadas estimativas

em fases anteriores à conclusão do projeto. Existem vários métodos que podem ser empregados para se realizar estimativas de tamanho, de esforço, de custo e de tempo. Mas, seja qual for o método empregado, é importante analisar a precisão dos resultados obtidos quando comparados aos dados reais. É importante notar que as características específicas que diferenciam um novo projeto dos anteriores adicionam um grau de incerteza às respostas. Esse grau de incerteza pode ser obtido através de uma medida do erro, que calcula a diferença entre o valor real e o valor estimado do projeto.

Diferentes medidas de erro têm sido utilizadas por vários pesquisadores (por exemplo, o erro quadrático médio, Pred25, dentre outras) mas nesta tese a principal medida para avaliar a precisão do modelo é a magnitude média do erro relativo – Mean Magnitude of Relative Error (MMRE). A escolha de MMRE deve-se ao fato desta ser a medida mais comumente utilizada em outros trabalhos relacionados à estimativa de esforço, o que facilita o estudo comparativo dos resultados. MMRE é a percentagem média de erros absolutos descrita na Fórmula 2.1.

$$MMRE = \frac{\left(\sum_{i=1}^{n} \left| \frac{M_{est} - M_{act}}{M_{act}} \right| * 100 \right)}{n} \tag{2.1}$$

em que n é o número de projetos; M_{act} é o esforço real observado; e M_{est} é o esforço estimado.

A partir da manutenção de uma base de dados históricos estimados e realizados dos projetos, a organização pode avaliar a adequação de seu processo de desenvolvimento e extrair indicadores de custo e qualidade, cada vez mais próximos de sua realidade e, portanto, cada vez mais confiáveis. Com esses indicadores as estimativas dos futuros projetos de software podem ser realizadas com segurança o mais cedo possível durante o seu ciclo de vida de desenvolvimento, ocasionando decisões mais rápidas e com um menor custo para a organização.

O processo de estimativa de um projeto de software envolve, basicamente, quatro atividades: 1) Estimar o tamanho do produto a ser gerado; 2) Estimar o esforço empregado na execução do projeto; 3) Estimar a duração do projeto; 4) Estimar o custo do projeto. O conjunto de procedimentos envolvidos no processo de estimativa é apresentado na Figura 2.2.

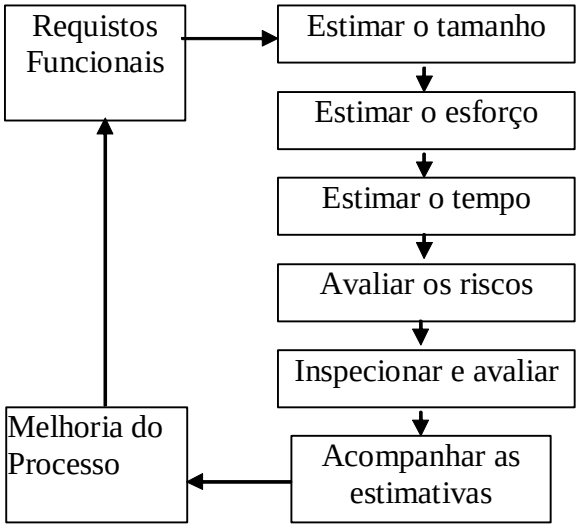

Figura 2. 2 - Processo de estimativa de projeto de software.
FONTE: Agarval (2001).

Este processo inicia-se nos primeiros níveis de abstração dos requisitos de projeto. Para ser completo e eficiente, devem ser considerados as experiências e os dados históricos de projetos passados; os recursos disponíveis dentro ou fora da organização, os dados de custo e os fatores de risco que cercam os projetos. A cada etapa realizada, as estimativas obtidas devem passar por um processo de aprovação antes de serem registradas na base de dados históricos. As estimativas poderão ser utilizadas como fonte de informação para projetos futuros e como base para as estimativas das etapas seguintes.

Refazer as estimativas deve ser uma atividade constante durante todo o ciclo de vida do desenvolvimento do software.A metodologia apresentada nesta tese pode ser aplicada em qualquer fase do ciclo de vida de desenvolvimento de software, a partir da especificação dos requisitos (fase em que é possível

estimar o tamanho do software, baseando-se na contagem de Pontos por Função, Pontos por Caso de Uso, dentre outros). É importante ressaltar que à medida que se evolui nas fases do ciclo de vida de desenvolvimento de software, é possível obter estimativas mais confiáveis.

Na etapa final do processo, com o produto concluído, as medidas reais de tamanho, esforço, duração e custo também devem ser devidamente registradas na base histórica, servindo também para a validação e o aprimoramento de todo o processo de estimativa. Independentemente da etapa em que se esteja no processo, existem várias técnicas de estimativas que podem ser utilizadas. Dentre elas os métodos diretos e os métodos derivados (Vazquez et al., 2003).

Os métodos diretos são aqueles baseados na opinião de especialistas. O procedimento padrão consiste em reunir a opinião de um ou mais especialistas, que fornecem uma suposição direta da estimativa baseando-se principalmente em suas intuições e experiências passadas.

Os métodos derivados, também conhecidos como métodos de modelo algorítmico ou métodos paramétricos, fornecem um ou mais algoritmos de transformação, que produzem a estimativa desejada em função de variáveis que se relacionam com alguns atributos de software.

O uso de técnicas de aprendizagem de máquina é outra abordagem que tem sido utilizada por pesquisadores em estimativas de esforço de software. Dentre estas técnicas podem ser citadas o raciocínio baseado em casos, redes neurais artificiais e os algoritmos genéticos.

Cada um destes métodos pode ser aplicado usando uma abordagem de estimativa *bottom-up* ou *top-down*. A estimativa *bottom-up* começa com os menores níveis de componentes e fornece uma estimativa para cada um. Essa abordagem combina estimativa em nível mais baixo dentro de estimativas de

nível mais alto. A estimativa *top-down* começa com o produto global. Estimativas para partes de componente são calculadas como porções relativas da estimativa completa (Fenton e PFleeger, 1997).

2.4 Considerações finais

Neste Capítulo o objetivo foi contextualizar estimativa de esforço em gerenciamento de projetos, ressaltando a importância dessa métrica em planejamento e acompanhamento de projetos. O próximo capítulo apresenta o estado da arte em estimativa de esforço de software e faz uma breve descrição das técnicas utilizadas nesta tese.

3 ESTIMATIVA DE ESFORÇO DE SOFTWARE

3.1 Considerações iniciais

Neste Capítulo apresenta-se o estado da arte em estimativas de esforço de software, apontando os desafios para se estimar. Além disso, é apresentada uma breve descrição das abordagens experimentadas nesta tese, que são: redes neurais artificiais, análise de regressão, análise de variância e análise de resíduos.

3.2 Revisão da literatura sobre estimativa de esforço

Dominar a arte de estimativa de projeto de software é um grande desafio para engenheiros e gerentes de software. Nesta tese, o conceito “Estimativa de Projeto de Software” é definido como o processo de predizer o esforço requerido para implementar uma solução de software com base nas especificações de requisitos. Esforço de desenvolvimento de software tipicamente inclui o esforço humano gasto para desenvolver o projeto de alto nível, o projeto detalhado, a codificação, o teste de unidade, o teste de integração e o teste de aceitação do cliente. Uma vez que o custo com pessoas é o custo dominante no desenvolvimento de software, o esforço está freqüentemente relacionado ao custo.

Assim, realizar estimativas precisas e consistentes do esforço requerido é crucial no gerenciamento efetivo de projetos de software. Apesar da quantidade de pesquisas nos últimos anos, a comunidade de software ainda é significativamente desafiada quando se trata de estimativas efetivas de esforço. A tarefa de estimar esforço de projeto de software precisamente é um desafio hoje como foi há trinta anos atrás (Moløkken-Østvold et al., 2004; Jorgensen e Shepperd, 2007). Várias questões em pesquisas sobre estimativa de projetos de software ainda não foram resolvidas. Uma dessas questões é o tipo de método apropriado para estimativa de esforço de projeto de software. Com o objetivo de responder a essa e outras questões, estimativa de esforço de

software se tornou uma importante área de pesquisa e tem sido tópico de muitos artigos, livros e seminários (Boehm, 1981; Albrecht e Gaffney, 1983; Shepperd e Schofield, 1997; Zhong et al, 2004; Sentas et al, 2005; Bisio e Malabocchia, 1995; Myrtveit et al, 2005; Samson et al, 1997; Subramanian et al., 2006; Jorgensen e Shepperd, 2007).

O método mais antigo e mais utilizado para estimativa de esforço é a “Opinião de Especialista”. Esta abordagem se refere a predições feitas por especialistas baseando-se em experiência passadas, mas geralmente quem realiza a estimativa são os próprios responsáveis pelo desenvolvimento do projeto. Para Moløkken-Østvold et al. (2004) deve-se ter um especialista em estimativa, pois em geral o pessoal de desenvolvimento conhece a expectativa do cliente sobre o custo e isso interfere na precisão das estimativas. Assim, esta abordagem é inteiramente dependente da disponibilidade de pessoas experientes no processo de estimativa, no domínio, em desenvolvimento e com registros de estimativas anteriores com boa precisão. Embora possa resultar em estimativas precisas, estudos mostram que 60% a 80% dos projetos terminam com estimativas acima do valor real e que a magnitude do erro varia de 30% a 40% (Bergeron e St-Arnaud, 1992 ; Jenkins et al., 1984 ; Phan, 1990; Boehm, 1981; Kitchenham, B., et al., 2005), enquanto gerentes consideram aceitável até 20% (Moores e Edwards 1992; Moløkken-Østvold et al, 2004).

As vantagens são que o método incorpora conhecimento de diferentes experiências de projetos passados. Para Jorgensen (2006) uma das razões que levam à preferência das empresas por este tipo de estimativa é que muitas das organizações de software se sentem desconfortáveis usando modelos que não sejam completamente entendidos. As desvantagens deste método são: a qualidade das estimativas depende da experiência e poder intuitivo do pessoal responsável pela estimativa; os próprios responsáveis não confiam em suas estimativas; as estimativas de especialistas são inconsistentes, ou seja, a mesma entrada pode gerar estimativas muito diferentes; as estimativas podem ser influenciadas e podem não ser analisáveis. Além disso, se há grande

rotatividade na equipe ou há perda de algum especialista, o processo de estimativa utilizando este método pode ser ineficiente senão impossível. Para Boehm (1981) as técnicas baseadas em experiência ou em conhecimento de especialista obtido previamente no campo, são úteis na ausência de dados empíricos quantificados.

Nas empresas, como o INPE, onde há grande rotatividade de pessoal de desenvolvimento e onde cada tipo de sistema exige um nível diferente de experiência, há, portanto, a necessidade de se ter um procedimento bem definido, baseado em experiência passada que permita armazenar o conhecimento (experiência) referente a vários projetos anteriores. O objetivo é que esse conhecimento possa ser transformado de forma que a organização (os gerentes experientes e inexperientes) utilize adequadamente o conhecimento gerado por especialistas para realizar estimativas. Essa abordagem é empregada por modelos de estimativa.

As pessoas tendem a usar estratégias de estimativa que requerem o mínimo esforço computacional, deixando de lado a precisão. Assim, algumas pesquisas têm sido conduzidas a fim de reverter essa situação e desenvolver modelos quantitativos, que reduzam a subjetividade do processo de estimativa. Os modelos algoritmos surgiram a partir da observação de que especialistas esquecem facilmente as atividades realizadas e subestima o esforço requerido para resolver eventos inesperados. O uso de modelos de estimativa aumenta o uso de dados históricos e conseqüentemente, remove a potencialidade de erros (ruídos) da visão do especialista.

Modelos de estimativa de esforço são algoritmos que relacionam alguma medida de entrada, geralmente uma medida de tamanho do produto, para uma medida de saída, neste caso, esforço. As técnicas mais utilizadas por estes modelos incluem abordagens de regressão múltipla. No que se refere à regressão, vários trabalhos discutem os prós e contras de regressão em

relação a outras técnicas e apresentam os resultados (Briand et al, 1992; Khoshgoftaar et al, 1995).

Os modelos matemáticos fornecem estimativas diretas de esforço. Eles usam o tamanho como a principal variável independente, o esforço como a variável de saída e incluem um número de fatores de ajuste chamados multiplicadores de esforço. Os multiplicadores de esforço, geralmente, são representados como medidas em escala ordinal que são atribuídas subjetivamente. Por exemplo, a experiência do programador pode ser representada por muito bom, bom, média, baixa, muito baixa. A principal vantagem de modelos de esforço é que eles podem ser utilizados por não especialistas.

Exemplos destes trabalhos incluem modelos paramétricos e baseados em regressão. O modelo COCOMO (Boehm, 1981) foi desenvolvido para estimar esforço e custo de desenvolvimento de software. Os modelos de estimativa de custo tais como COCOMO utilizam principalmente o número de linhas de código fonte (Source Line Of Code (SLOC) como a base para estimativa de esforço. Com isto, o esforço em pessoas /mês é expresso como uma função de Kilo Source Lines Of code (KSLOC). O modelo COCOMO II, que é a versão atual de COCOMO usa 17 multiplicadores de esforço e cinco fatores de escala para estimar esforço de desenvolvimento baseado no tamanho do projeto. Para Agrawal e Chari (2007), multiplicadores de esforço como experiência com a aplicação, linguagem de programação e experiência com as ferramentas, não são significantes. Este é um dos mais conhecidos modelos para estimativa de esforço de software.

Uma métrica alternativa para SLOC é pontos de função (Albrecht, 1979; Vazquez et al., 2006), em que a estimativa é realizada multiplicando-se a quantidade de pontos de função pelo ajuste da complexidade processada Os pontos de função passam a ter maior significado quando utilizados como parâmetros na obtenção de outras variáveis relevantes para a gerência de projetos, por exemplo, para estimar o esforço (Matson et al, 1994). O esforço

estimado para um projeto é medido pela alocação dos recursos humanos em termos de homem/mês ou homem/hora. A inferência a ser realizada pode ser direta caso o conjunto de dados contenha o dado sobre a produtividade média do desenvolvimento conforme a plataforma de hardware/software e respectiva metodologia ou se houver o valor de Ponto de Função similar ao estimado. A interpolação também pode ser empregada nos casos de valores de pontos de função estimados, intermediários aos existentes no conjunto de dados.

A análise de pontos de função permite sua utilização em estimativas de esforço já na atividade de projeto, pois é baseada na funcionalidade. Porém para se ter uma melhor precisão das estimativas, esta técnica depende da disponibilidade de informações mais detalhadas do sistema. Mas tal informação é definida na especificação de projeto detalhado do software e isso pode exigir um tempo que o cliente não está disposto a esperar, no atual contexto de desenvolvimento de software. Além disso, apresenta uma subjetividade inerente ao modelo, já que diferentes profissionais podem obter estimativas diferentes para a mesma entrada. De acordo com Heemstra e Kusters (1991) as estimativas que utilizam a métrica de pontos por função têm um esforço excedido maior que aquelas utilizando outros modelos.

Por isso, é difícil aplicar a métrica de pontos por função para estimativa de esforço de software. O método de estimativa por Pontos por Caso de Uso (Use Case Point - UCP) é recente, tendo se originado da evolução do método de estimativa por Pontos por Função, desenvolvido principalmente para projetos que utilizam metodologias orientadas a objetos. O uso deste método tem aumentado no mercado devido ao uso da modelagem orientada a objeto e por permitir um maior grau de precisão. A estimativa de esforço é realizada multiplicando-se o valor de UCPs por um valor específico de homens/mês ou homens/hora. Karner (1993) sugere que seja utilizado um fator de 20 homens/hora por UCP para um projeto, enquanto que para Schneider e Winters (1998) o número de homens/hora por UCP depende dos fatores ambientais.

Uma vez que os pontos por caso de uso são medidos a partir do modelo de caso de uso que define o escopo funcional do sistema de software a ser desenvolvido, o objetivo é o de possibilitar estimativas de esforço em fases anteriores ao projeto detalhado. Porém, como em contagem de pontos de função, a precisão depende do nível de detalhes contido no modelo, além da experiência do responsável pela estimativa e da experiência do profissional que desenvolve o modelo de caso de uso. Assim, é um método difícil de aplicar em fases iniciais do ciclo de vida do software, pois depende de uma boa especificação de requisitos. A metodologia de raciocínio baseada em casos representa uma abordagem alternativa, a qual pode ser utilizada quando ainda não se tem uma especificação completa dos requisitos, pois ela se baseia em características de mais alto nível para realizar as estimativas.

A precisão das estimativas de esforço utilizando pontos por caso de uso ainda está aquém da desejada. Várias pesquisas e estudos têm sido relatados sobre estimativa de esforço baseado em modelo de casos de uso. Anda et al. (2002) aplicam o método de pontos por caso de uso para estimar esforço para três tipos de projetos e os resultados mostraram que o esforço estimado para cada projeto é quatro ou cinco vezes maior que o real. Os autores sugeriram que o método pontos por caso de uso deve ser utilizado com outro método de estimativa (como por exemplo, o COCOMO).

A desvantagem da abordagem de estimativa baseada em modelo é que a fórmula básica deve ser modificada quando houver mudanças em métodos de desenvolvimento. Quando estimativas são realizadas a partir de modelos previamente calibrados, sem um procedimento de re-calibração para a situação atual, significa que o responsável pelas estimativas assume que o futuro é o mesmo que o passado. Para Kitchenham et al. (1997), em tais circunstâncias, será possível obter resultados razoáveis somente quando se tratar de projetos médios. A necessidade de ajuste de fórmula pode ser solucionada pela estimativa baseada em analogia, que é uma abordagem formal para opinião de especialista. O responsável por realizar a estimativa compara o projeto

proposto com um ou mais projetos passados. As diferenças e similaridades são identificadas e usadas para ajustar a estimativa. Este responsável tipicamente identificará o tipo de aplicação, estabelecerá uma predição inicial e então refinará as predições.

Há pesquisas substanciais nesse sentido. Por exemplo, o Raciocínio Baseado em Casos (RBC) é uma abordagem semelhante àquela utilizada por um especialista (que é a preferida pela indústria de software), com a vantagem de dispensar a necessidade de um especialista e de oferecer uma precisão maior. O modelo é gerado em tempo de estimativa, evitando problemas relacionados a grandes desvios quando não se tem experiência em um domínio. A desvantagem dessa abordagem se refere à necessidade de uma base de dados históricos com muitos projetos. Mas grande parte das empresas não tem um sistema de registro de experiência implantado. Uma abordagem baseada em analogia chamada ESTOR foi desenvolvida para estimativa de esforço de software. Vicinanza et al (1990) mostraram que a abordagem ESTOR é comparável a um especialista e tem um desempenho significativamente melhor que o COCOMO e pontos de função quando utilizada sobre amostras restritas de problemas. Porém, Barcelos Tronto et al. (2006a) em um estudo experimental utilizando raciocínio baseado em casos verificou que a precisão da abordagem baseada em analogia é dependente da disponibilidade de informações sobre projetos históricos, conforme sugerido em (Fenton e Pfleeger, 1997). Para Jeffery et al. (2001) os resultados de estimativa de desenvolvimento de software indicam que o uso semi-automatizado de dados leva a uma melhor precisão de estimativa. Para eles as pessoas são boas para encontrar analogias, mas não se ajustam apropriadamente quando se tem grande diferença entre as tarefas a serem estimadas e as tarefas similares. Esta abordagem poderá ser utilizada no desenvolvimento de softwares como um método alternativo para estimativa de esforço em fases anteriores ao projeto, quando ainda não se conhece o tamanho do software.

Entretanto, outras técnicas para análise exploratória de dados, como redes neurais artificiais (RNA) têm se mostrado efetivas para predizer esforço de software, apresentando boa precisão quando comparado com modelos algorítmicos: como Análise de Pontos de Função, COCOMO, SLIM, etc. Alguns trabalhos de pesquisa têm usado RNA a fim de produzir estimativas de recursos mais precisas (Gray e MacDonnell, 1999; Witting e Finnie, 1997). Em Karunanithi et al (1992) as redes neurais são utilizadas para estimar a confiabilidade de software. Eles conduziram experimentos utilizando redes com retro propagação de erro e com redes de Jordan. Samson et al. (1997) utilizaram um perceptron de mútiplas camadas para predizer o esforço de software sobre o conjunto de dados COCOMO e compararam os resultados com aqueles obtidos utilizando regressão linear simples.

Porém, a precisão dos resultados obtidos com as duas abordagens não é animadora. Isso pode ser atribuído à forma com que os dados foram preparados, pois não foi relatada nenhuma atividade de preparação dos dados. Srinivasan e Fisher (1995) também relatam o uso de uma rede neural com um algoritmo de aprendizagem com retro propagação de erro. Eles argumentam que a rede neural teve um desempenho melhor que outras técnicas e levou a resultados cujo MMRE = 70%. Entretanto, eles não deixam claro como o conjunto de dados foi dividido para treinar e avaliar o modelo. A rede neural artificial é uma técnica promissora para construir modelos de predição, porque elas são capazes de modelar relacionamentos não-lineares.

Em geral, os trabalhos relacionados com o uso de redes neurais para estimar esforço de desenvolvimento de software têm como principal objetivo comparar o grau de precisão de modelos algorítmicos, ao invés de investigar a conveniência da abordagem para construir sistemas de predição. Um exemplo é o trabalho de Finnie et al. (1997), em que se explora o uso de uma rede *perceptron* de múltiplas camadas sobre o conjunto de dados Desharnais e da Associação de Métricas de Software Australiana (Australian Software Metrics Association - ASMA). Para o conjunto de dados Desharnais eles dividiram os

projetos randomicamente três vezes, sendo 10 projetos para o conjunto de teste e 71 projetos para o conjunto de treinamento (um procedimento semelhante ao que foi seguido nesta tese, no que se refere à divisão do conjunto de dados em amostras para teste e amostras para treinamento). Os resultados destes três conjuntos de validação são agregados e apresentam um bom nível de precisão, quando dados de alguns projetos são excluídos (dados que foram considerados exceções). Barcelos Tronto et al (2007a) apresentam resultados significativos ao comparar o nível de precisão obtido com redes neurais e outras abordagens, como análise de regressão simples e múltipla, quando estes dados (exceções) são considerados na análise.

Finalmente, nos últimos anos tem crescido o interesse pelo uso de redes neurais artificiais. As RNAs têm sido aplicadas com sucesso em vários domínios de problemas, em áreas tais como medicina, engenharia, geologia e física para projetar soluções para estimativas, classificação, problemas de controle, dentre outros. As RNAs podem ser usadas como modelos de predição porque são técnicas capazes de modelar funções complexas. Os algoritmos de aprendizagem de máquina tais como as RNAs oferecem recurso para tratar o problema de muitos fatores. Elas são efetivas quando se tem um número relativamente grande de dados numéricos bem como quando se tem um grande número de fatores.

Esta tese apresenta os resultados de estudos de caso usando redes neurais artificiais – RNA, um procedimento utilizando análise de regressão, análise de variância e análise de resíduos para estimar o esforço de software a partir do tamanho do projeto (dado pela quantidade de linhas de código fonte) e outras variáveis direcionadoras de esforço e de custo. Os resultados obtidos com cada abordagem são comparados e discutidos.

3.3 Redes neurais artificiais

As redes neurais artificiais são modelos inspirados na arquitetura de redes neurais biológicas, que compreendem unidades simples interconectadas (neurônios artificiais). A arquitetura resultante resolve problemas através do aprendizado das características dos dados disponíveis relacionados ao problema.

Redes neurais é a técnica mais indicada na construção de modelo de estimativa de software, como uma alternativa para a técnica de regressão de mínimos quadrados (Gray e McDonell, 1997). Elas permitem a geração de modelos de estimativa que podem ser treinados usando dados históricos para produzir melhores resultados, ajustando-se automaticamente os valores de parâmetros do algoritmo para reduzir o desvio entre as estimativas do modelo e os valores reais conhecidos.

3.3.1. Modelo geral de neurônio

Os primeiros trabalhos desenvolvidos na área de redes neurais artificiais datam de 1943, quando o neurofisiologista, filósofo e poeta americano Warren McCulloch, e o lógico Walter Pitts desenvolveram o primeiro modelo matemático de um neurônio. A estrutura do neurônio artificial proposto por McCulloch e Pitts (1943) é baseada no neurônio biológico.

O neurônio artificial representa um simples dispositivo capaz de calcular o somatório de N entradas (x_1, x_2,..., x_m) ponderadas pelos pesos w_1, w_2, ..., w_m, cujo resultado passa através de uma função não-linear. Basicamente, este dispositivo é caracterizado por um *bias* (polarização) ou limiar interno e por algum tipo de função não-linear. O modelo de neurônio apresentado em (Haykin, 1999) inclui um “bias” aplicado externamente (denotado por b_k), que se comporta como uma entrada estimulante (positivo) ou como um inibidor (negativo) para outros neurônios na rede, dependendo se ele é positivo ou

negativo, respectivamente. Um neurônio *k* pode ser descrito pelas Equações 3.1 e 3.2.

$$u_k = \sum_{j=1}^{m} \omega_{kj} x_j \tag{3.1}$$

$$y_k = \varphi(u_k + b_k) \tag{3.2}$$

em que x_1, x_2,..., x_m são os sinais de entrada; w_{k1}, w_{k2}, ..., w_{km} são os pesos do neurônio *k*; u_k é a saída da combinação linear; b_k é o "bias"; $\varphi(\ )$ é a função de ativação e y_k é a saída do neurônio.

A função de ativação, denotada por $\varphi(v)$, define a saída do neurônio. Três tipos básicos são descritos:

1. *Função Limiar*: Esta função é descrita matematicamente na Equação 3.3.

$$\varphi(v_k) = \begin{cases} 1 \ se \ v_k \geq 0 \\ 0 \ se \ v_k < 0 \end{cases} \tag{3.3}$$

Da mesma forma, a saída do neurônio *k*, pode ser definida pela Equação 3.4.

$$y_k = \varphi(v_k) = \begin{cases} 1 \ se \ v_k \geq 0 \\ 0 \ se \ v_k < 0 \end{cases} \tag{3.4}$$

em que o valor de v_k é descrito pela Equação 3.5.

$$v_k = \sum_{j=1}^{m} \omega_{kj} x_j + b_k \tag{3.5}$$

Esta representação descreve a propriedade de tudo-ou-nada do modelo de McCulloch e Pitts.

2. *Função Semi-Linear.* Esta função é descrita na Equação 3.6.

$$y_k = \varphi(v_k) = \begin{cases} 1 & v_k \geq +\frac{1}{2} \\ v_k & +\frac{1}{2} > v_k > -\frac{1}{2} \\ 0 & v_k \leq -\frac{1}{2} \end{cases} \quad (3.6)$$

em que o fator de amplificação dentro da região de operação linear é unitário.

3. *Função Sigmóide.* Esta função tem a forma de um "S" e é a função de ativação mais comumente utilizada na construção de redes neurais. A função logística é um exemplo da função sigmóide, definida pela Equação 3.7.

$$y_k = \varphi(v_k) = \frac{1}{1 + \exp(-a\, v_k)} \quad (3.7)$$

em que *a* é o parâmetro de inclinação da função sigmóide. Variando-se o parâmetro *a* obtém-se funções sigmóides de diferentes inclinações, como mostrado em (Haykin, 1999). Neste trabalho de doutorado, os dados de esforço de desenvolvimento de software disponíveis apresentam um comportamento não- linear. Assim, foi utilizada a função sigmóide, a qual ajusta melhor a esta realidade.

A rede neural resulta do arranjo de neurônios em camadas, que são interconectadas umas às outras. Uma rede neural pode ser caracterizada de acordo com a sua topologia (isto é, pelo número de camadas, de elementos de processamento e de conexões), com as características de seus elementos de processamento e com as leis de aprendizagem a que foram submetidas.

3.3.2. Arquiteturas de redes neurais

A maneira através da qual os neurônios de uma rede neural são estruturados é intimamente relacionada com o algoritmo de aprendizagem usado para treinar a rede. A classificação dos algoritmos de aprendizagem é abordada na próxima seção. Nesta seção, são apresentadas três diferentes classes de arquiteturas de rede, que são consideradas fundamentais: redes de uma única camada, redes de múltiplas camadas e redes recorrentes.

Do ponto de vista da topologia da interligação entre neurônios, uma rede pode ser de alimentação para frente (do inglês, "*feedforward*") ou recorrentes. Nas redes de alimentação para frente, os neurônios são organizados em camadas e a informação se desloca em um único sentido, entre camadas adjacentes. Nas redes recorrentes, não existe, direção privilegiada para a propagação da informação, podendo haver retroalimentação.

A forma mais simples de arquitetura é a rede de uma única camada (do inglês, "*Single-Layer Feedforward Networks"*). Nesta classe de rede, uma camada de entrada (de nós de origem) é projetada sobre a camada de neurônios de saída (denominados nós de computação). Esta rede é estritamente acíclica. Toda a computação é realizada sobre a camada de saída e nenhuma computação é realizada sobre a camada de entrada, por isso esta última não é considerada uma camada.

A rede de múltiplas camadas (do inglês, "*Multilayer Feedforward Networks")* se distingue pela presença de uma ou múltiplas camadas escondidas, em que os nós de computação são chamados neurônios escondidos ou unidades escondidas. A função dos neurônios escondidos é intervir entre a entrada externa e a saída da rede, de alguma maneira útil. Ao adicionar uma ou mais camadas escondidas, a rede se torna capaz de extrair estatísticas de ordem mais alta. A habilidade dos neurônios escondidos extrair estatísticas mais altas é particularmente importante quando o tamanho da camada de entrada é

grande. Os nós de origem na camada de entrada da rede fornecem os respectivos elementos do modelo de ativação (vetor de entrada), que constituem os sinais de entrada aplicados aos neurônios (nós de computação) na segunda camada (ou seja, a primeira camada escondida). Os sinais de saída da segunda camada são usados como entradas para a terceira camada, e assim por diante para o resto da rede. Tipicamente os neurônios em cada camada da rede têm como suas entradas os sinais de saída da camada precedente. O conjunto de sinais de saída dos neurônios na camada de saída (final) da rede constitui a resposta geral da rede para o modelo de ativação fornecido pelos nós de origem na camada de entrada (primeira camada).

A rede neural recorrente distingue de uma rede neural *feedforward* pelo fato de possuir pelo menos um laço de realimentação. Por exemplo, uma rede recorrente pode consistir de uma única camada de neurônios, sendo que cada neurônio envia seu sinal de saída de volta para as entradas de todos os outros neurônios. A presença de laços de realimentação pode exercer forte impacto na capacidade de aprendizagem da rede e sobre seu desempenho.

O modelo mais comum de rede neural usada no contexto de estimativa de software é a rede *perceptron* de múltiplas camadas, que é treinada usando o algoritmo de retro-propagação do erro (*error back-propagation)*. O desenvolvimento de tal modelo neural requer uma estrutura de neurônios ou conexão entre os nós da rede apropriada. Isto inclui definir o número de camadas de neurônios, o número de neurônios dentro de cada camada e a maneira através da qual eles são todos ligados. Uma breve descrição desta arquitetura é apresentada na Seção 3.3..4, tendo em vista que este é o modelo utilizado nesta tese.

3.3.3. Processos de aprendizagem

O processo de aprendizagem (ou treinamento) consiste em pesquisar pelos melhores valores de pesos entre os nós, uma vez que a rede tenha sido

construída. Um conjunto de regras bem definidas para a solução de um problema é chamado algoritmo de aprendizagem. Existem diversos algoritmos de aprendizagem e eles diferem uns dos outros pela formulação de ajuste dos pesos de um neurônio. Haykin (2001) apresenta cinco regras de aprendizagem básicas: aprendizagem baseada na correção do erro, aprendizagem baseada em memória, aprendizagem de Hebbian, aprendizagem competitiva e aprendizagem de Boltzmann.

A aprendizagem baseada na correção do erro é fundamentada em um mecanismo de controle do erro, que aplica uma seqüência de ajustes corretivos nos pesos do neurônio *k* de forma a minimizar o erro (ou seja, a diferença entre o valor calculado pelo neurônio de saída e o valor esperado). Os ajustes corretivos são aplicados de forma a tornar o sinal de saída y_k (n) o mais próximo possível à resposta desejada $d_k(n)$. Os algoritmos Hebbian e aprendizagem competitiva são inspirados pelas considerações neurobiológicas. A aprendizagem de Boltzmamm é diferente porque é baseada em idéias adotadas de mecanismos estatísticos.

Uma rede pode ser classificada de diversas maneiras, por exemplo, quanto à forma de aprendizado: supervisionado e não-supervisionado.

No aprendizado supervisionado os conjuntos de padrões de entrada e seus correspondentes padrões de saída são apresentados à rede, sucessivamente. Durante este processo, a rede realiza um ajustamento dos pesos das conexões entre os elementos de processamento, segundo uma determinada lei de aprendizagem, até que o erro entre os padrões de saída gerados pela rede alcance um valor mínimo desejado. Por exemplo, perceptron, adaline e madaline, backpropagation (Hecht-Nielsen, 1990; Freeman e Skapura, 1991; Beale e Jackson, 1992), são algumas dentre as dezenas de leis de aprendizagem supervisionada.

Um outro tipo de aprendizagem similar à supervisionada é a aprendizagem por reforço. Neste tipo de aprendizagem, ao invés de fornecer as saídas corretas para a rede, relativo a cada treinamento individual, a rede recebe somente um valor que diz se a saída está correta ou não (Hecht-Nielsen, 1990; Freeman e Skapura, 1991; Beale e Jackson, 1992).

No aprendizado não-supervisionado a rede “analisa” os conjuntos de dados apresentados a ela, determina algumas propriedades dos conjuntos de dados e “aprende” a refletir estas propriedades na sua saída. A rede utiliza padrões, regularidades e correlações para agrupar os conjuntos de dados em classes. As propriedades que a rede vai “aprender” sobre os dados podem variar em função do tipo de arquitetura utilizada e da lei de aprendizagem. Por exemplo, Mapa Auto-Organizavél de Kohonen, Redes de Hopfield e Memória Associativa Bidirecional, são algumas métodos de aprendizado não-supervisionado (Hecht-Nielsen, 1990; Freeman e Skapura, 1991; Beale e Jackson, 1992).

O algoritmo mais conhecido para treinamento de redes neurais é a retropropagação (do inglês, “*backpropagation*”). Esse algoritmo pode ser considerado como uma generalização da regra delta para redes de alimentação para frente com mais de duas camadas. A retropropagação é um algoritmo de treinamento supervisionado. Seu funcionamento pode ser descrito da seguinte forma: (i) apresenta-se um exemplo à rede e obtém-se a saída correspondente; (ii) calcula-se o vetor de erro que consiste na diferença entre a saída obtida e a esperada; (iii) calcula-se o gradiente do vetor de erro e atualiza-se, utilizando a regra delta, os pesos da camada de saída e, finalmente (iv) propaga-se para trás (origem do nome do algoritmo) os valores desejados de modo a atualizar os pesos das demais camadas.

Como o algoritmo de retropropagação requer o cálculo do gradiente do vetor de erro, é interessante que a função de ativação seja derivável em todos os pontos. Isso explica o sucesso da função sigmóide como função de ativação, pois ela apresenta esta propriedade. Outros algoritmos de treinamento de

redes neurais são: contrapropagação (do inglês, "*counterpropagation"*); aprendizado competitivo, utilizado nas redes de Kohonen; algoritmos genéticos.

3.3.4. Rede percetron multicamadas

Esta é uma importante classe de redes neurais. Tipicamente, a rede consiste de um conjunto de unidades (nós de origem) que constitui a camada de entrada, uma ou mais camadas escondidas de nós de computação, e uma camada de saída de nós de computação. O sinal de entrada propaga pela rede em uma direção para frente, camada por camada. Estes redes são comumente referenciadas como perceptrons multicamada (do inglês, *multilayer perceptrons - MLPs*), que representa uma gereneralização da perceptron de uma única camada (do inglês, single-layer percetron).

MLPs são aplicadas com sucesso para resolver algumas dificuldades e diversos problemas, sendo treinadas de uma maneira supervisionada com um algoritmo conhecido como algoritmo de retropropagação do erro. Este algoritmo utiliza a regra de aprendizagem baseada na correção do erro (do inglês, *error-correction learning*).

Basicamente, o algoritmo de retropropagação do erro consiste de duas passagens pelas diferentes camadas da rede: uma passagem adiante e uma passagem de volta. Na passagem adiante, um padrão de atividade (vetor de entrada) é aplicado aos nós da rede, e seu efeito se propaga pela rede, camada a camada. Finalmente, um conjunto de saídas é produzido como a resposta real da rede. Durante a passagem adiante os pesos da rede são fixados. Durante a passagem de volta, por outro lado, os pesos são todos ajustados de acordo com uma regra baseada na correção do erro. De acordo com a definição desta regra, a resposta real da rede é subtraída da resposta desejada para produzir o erro. Este erro é então propagado de volta através da rede em direção contrária às conexões, por isso o nome retropropagação do erro. Os pesos são ajustados para tornar a resposta real da rede mais próxima

da resposta desejada, estatisticamente. O processo de aprendizagem com este algoritmo é chamado aprendizagem com retropropagação (do inglês, *back-propagation* learning).

Uma rede perceptron multicamadas tem três características distintas:

- O modelo de cada neurônio na rede inclui uma função de ativação não-linear. Uma forma de linearidade comumente utilizada é a não-linearidade sigmoidal definida pela função logística mostrada na Equação 3.7.
- A rede contém uma ou mais camadas de neurônios escondidas, que não são parte da entrada ou saída da rede. Estes neurônios escondidos tornam a rede capaz de aprender funções complexas, extraindo progressivamente, mais características importantes dos padrões (vetores) de entrada.
- A rede exibe um alto grau de conectividade. Uma mudança na conectividade da rede requer uma mudança na população de conexões ou em seus pesos.

Com base nos estudos realizados, as redes multicamadas do tipo feedfoward com método de aprendizagem supervisionado são consideradas as mais adequadas. A seguir, o algoritmo de backpropagation descrito em (Medeiros, 1999; Haykin, 2001) é mostrado, uma vez que ele é utilizado no contexto deste trabalho.

Pode-se dizer que o algoritmo de backpropagation é uma generalização do algoritmo do método dos mínimos quadrados, que utiliza técnicas de gradiente descendente interativo para minimizar uma função de custo igual a diferença média quadrada entre a saída desejada e a saída real da RNA.

Para esclarecer sobre este processo de treinamento considere o exemplo de uma rede neural de quatro camadas, mostrada na Figura 3.1 (Medeiros, 1999).

Esta rede utiliza N valores contínuos como as suas entradas, M valores como suas saídas e duas camadas de unidades ocultas. Escolhe-se uma função não-linear (sigmóide) para cada unidade de processamento. A regra de decisão escolhida é: “selecione classe correspondente ao elemento de processamento com a maior saída”. Nas fórmulas, X'_j, X''_k correspondem às saídas dos elementos de processamento da primeira e segunda camada oculta; θ_i, θ'_k, θ''_l são os “bias” internos dos elementos de processamento das camadas ocultas e de saída, w_{ij} denota os pesos das conexões entre a camada de entrada e a primeira camada oculta, w_{ij}' e w_{ij}'' refere-se aos pesos entre as duas camadas ocultas e aos pesos entre a última camada oculta e a camada de saída.

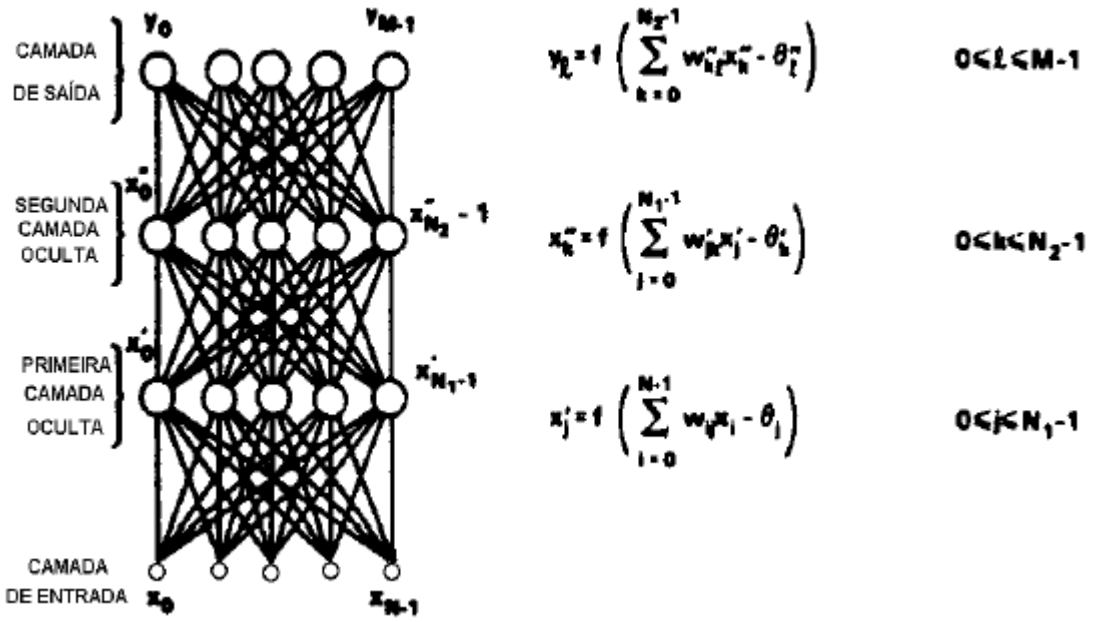

Figura 3. 1 – Rede percetron multicamadas
Fonte: Medeiros (1999)

No processo de treinamento, inicialmente os elementos de processamento da RNA são iniciados aleatoriamente com diferentes limiares internos e, as conexões entre eles com pequenos pesos (são comuns, valores que estão entre -1.0<w<1.0, onde w denota o peso de uma conexão qualquer). Em

seguida, os dados de treinamento são apresentados à RNA repetidamente e a cada ciclo de treinamento os pesos são ajustados através de uma informação complementar que indica a correta classe de saída, até que a função de custo seja reduzida a um valor aceitável. A parte principal do algoritmo de backpropagation é a maneira interativa pela qual os erros utilizados para adaptar os pesos são propagados para trás, isto é, a partir da camada de saída para as camadas anteriores (Lippmann, 1987).

No processo de treinamento com algoritmo de backpropagation (retropropagação) são realizados os seguintes passos:

Passo 1 - Inicializa-se todos o pesos e todos os "bias" dos elementos de processamento com pequenos valores aleatórios.

Passo 2 - Apresenta-se um vetor x_0, x_1, ..., x_n (de entrada) de valores contínuos e o vetor d_0, d_1, ..., d_n de saída desejado.

Passo 3 – Calcule as saídas y_0, y_1, ..., y_n utilizando as fórmulas relacionadas a cada camada.

Passo 4 – Faz a adaptação dos pesos. Usa-se um algoritmo recursivo iniciando nos elementos de processamento de saída, trabalhando para trás no sentido da primeira camada, ajustando os pesos através da fórmula $w_{ij}(t + 1) = w_{ij}(t) + \eta\delta_j x_i$' em que w_{ij} é o peso do elemento de processamento oculto j no tempo t; x_i pode ser um elemento de processamento de saída ou uma entrada; η denota um termo de ganho (velocidade da aprendizagem) e; δ_j um termo de gradiente para o elemento de processamento j; se j for um elemento de saída então $\delta_j = y(1-y_j)(d_j - y_j)$, onde d_j denota a saída desejada e y_j é a saída real da rede; se o elemento j for um elemento oculto então $\delta_j = x'_j (1-x'_j) \sum_k \delta_k w_{jk}$, onde k denota todos os elementos à frente dos elementos j; os limiares dos elementos internos são ajustados de forma semelhante; a convergência algumas vezes

pode ser mais rápida se um termo de momento for adicionado e os pesos alterados de forma mais suave.

Passo 5 - Retorne para o passo 2.

Uma demonstração mais detalhada do algoritmo de retropropagação do erro pode ser vista em Rich e Knight (1993).

3.4 Análise de regressão

A análise de regressão tenta encontrar um relacionamento linear entre uma ou mais variáveis independentes e uma variável dependente (nesse caso, esforço); fornece uma equação descrevendo a natureza do relacionamento entre as variáveis e permite avaliar a precisão do modelo.

Em uma análise de regressão simples, o objetivo é predizer o valor da variável dependente baseando-se no valor de uma única variável independente. Por exemplo, pode-se predizer o esforço de software baseando-se somente no tamanho da aplicação. Nesse caso, o esforço é a variável dependente e o tamanho é a variável independente.

Na análise de regressão múltipla, o objetivo é predizer o valor da variável dependente baseando-se em várias variáveis independentes. Por exemplo, pode-se predizer o esforço de software baseando-se no tamanho da aplicação, confiabilidade requerida, complexidade, tamanho da equipe, dentre outros fatores. Nesse trabalho de doutorado foram utilizadas as duas abordagens: regressão simples e regressão múltipla.

As técnicas baseadas em regressão são as formas mais comumente utilizadas para construir modelos. As razões para a popularidade são a facilidade de usar e a simplicidade. Elas estão disponíveis como uma opção nos mais diversos pacotes estatísticos comerciais .

Um modelo de custo baseado em análise de regressão é derivado seguindo alguns passos, dentre eles: 1) os dados de projetos passados são coletados; 2) o relacionamento entre as medidas dos atributos é capturado; 3) os engenheiros de software assumem que alguns fatores podem ser relacionados por uma equação. O atributo tamanho é o mais relacionado ao esforço; 4) a equação básica obtida é então ajustada por outros fatores direcionadores de esforço secundários (Fenton e Pfleeger, 1997). Porém, se o tamanho fosse uma variável de estimativa de esforço perfeita, todo ponto do gráfico deveria ficar sobre a linha da equação, com um erro residual igual a zero, mas na realidade, existe um erro residual significante e ela não é perfeita.

3.4.1. Regressão simples

O método de mínimos quadrados ajusta uma reta sobre os dados de forma a minimizar a soma dos erros quadráticos. Os erros são as diferenças entre os valores reais e os valores estimados (referenciados como os resíduos).

Na Figura 3.1, os três pontos, (x_1, y_1), (x_2, y_2) e (x_3, y_3), representam os valores reais. Os valores estimados, (x_1, $\hat{y}_1$), (x_2, $\hat{y}_2$) e (x_3, $\hat{y}_3$), estão sobre a linha. Os erros são as diferenças entre y e $\hat{y}$ para cada observação. O objetivo é encontrar a linha reta que minimiza: $erro_1^2 + erro_2^2 + erro_3^2$.

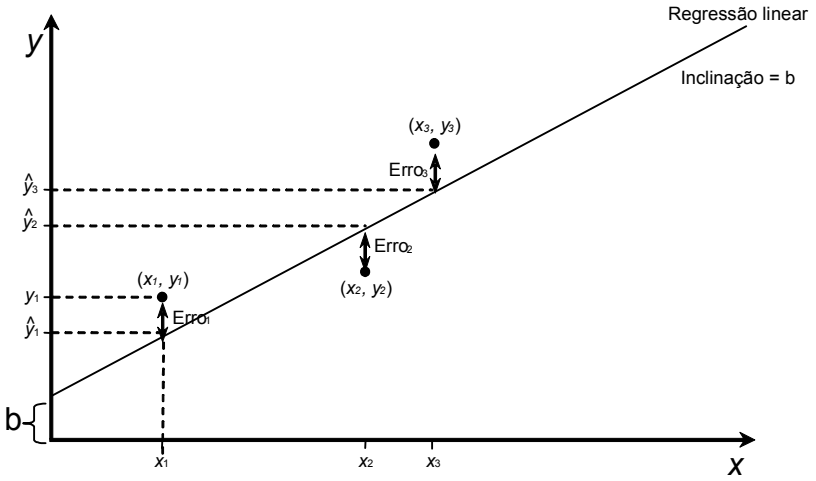

Figura 3. 2- Erros da regressão

Relembra-se a partir da álgebra que a equação para a linha reta é da forma dada pela Equação 3.8:

$$\hat{y} = a + bx \tag{3.8}$$

em que *ŷ* é o valor estimado da variável dependente, y, dados o valor da variável independente, x. A constante *a* representa o valor de *ŷ* quando *x* é zero. A constante *b* representa a inclinação da linha. Ele será positivo quando existir um relacionamento positivo e negativo quando existir um relacionamento negativo.

Para encontrar os valores de *a* e *b* de uma reta ajustada pelo método de mínimos quadrados, as Equações 3.9 e 3.10 devem ser resolvidas, simultaneamente:

$$\sum y = na + b\sum x \tag{3.9}$$

$$\sum xy = a\sum x + b\sum x^2 \tag{3.10}$$

em que n é o número de observações. Substituindo os valores conhecidos de *x* e *y*, os valores de *a* e *b* podem ser calculados.

3.4.2. Precisão da regressão

Uma linha de regressão é somente uma medida do relacionamento entre as variáveis dependentes e independentes. A menos que exista correlação perfeita, na qual todas as observações estejam sobre uma reta, existirão erros nas estimativas. Quanto mais distantes os valores reais estiverem da linha de regressão, maior o erro de estimativa. Como se pode traduzir isto em uma medida que possa dizer se o ajuste da linha de regressão é bom? Para responder esta questão considere que um gerente de projetos tenha que convencer ao seu chefe sobre as vantagens de se utilizar uma regressão ao invés de se calcular o esforço médio de todos os projetos passados. Nesse

caso, o gerente de projeto poderá comparar os resultados obtidos pela equação de regressão com aqueles obtidos através do cálculo da média dos esforços passados.

A Figura 3.2 mostra um exemplo que utiliza três projetos. Considere que y é o esforço do projeto e x é o tamanho. Pode-se ver que para o Projeto 1, o valor médio de esforço, $\bar{y}$, é maior que o valor real e o valor estimado, $\hat{y}_1$, é menor que o valor real. Para o Projeto 2, o valor médio e o valor estimado de esforço, $\hat{y}_2$, são maiores que o valor real. Para o Projeto 3, o valor médio e o valor estimado, $\hat{y}_3$, são menores que o valor real de esforço. É necessário comparar as diferenças entre os valores reais, os valores estimados e o valor médio para cada projeto, a fim de se calcular a precisão global do modelo.

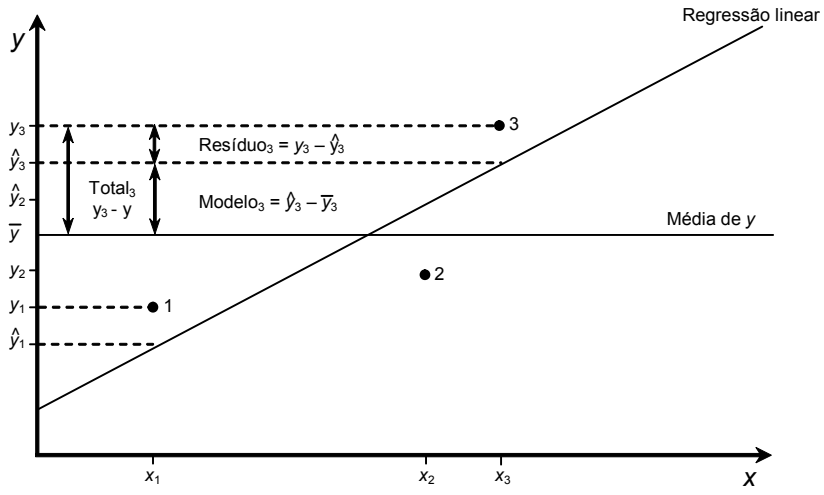

Figura 3. 2- Precisão da regressão.

O erro quadrático total entre o valor real de esforço e o valor médio de esforço para cada projeto é dado pela Equação 3.11.

$$Total\ SS = \sum \left(y_i - \bar{y}\right)^2 \tag{3.11}$$

Isto é a variação total dos dados. Se o esforço realmente depende do tamanho, então os erros (resíduos) deveriam ser pequenos quando comparados à variação total dos dados. O erro (resíduo SS) é a soma das diferenças

quadráticas entre o valor real de esforço e o valor de esforço estimado para cada projeto, conforme mostrado na Equação 3.12.

$$Resíduo\ SS = \sum \left(y_i - \hat{y}_i \right)^2 \tag{3.12}$$

Isso pode ser também pensado como a variação total dos dados não explicados pelo modelo. A variação total dos dados é igual a variação explicada pelo modelo mais a variação não explicada pelo modelo, que é, Total SS = Modelo SS + Resíduo SS. A variação nos dados explicada pelo modelo é calculada através da Equação 3.13.

$$Modelo\ SS = \sum \left(\hat{y}_i - \bar{y} \right)^2 \tag{3.13}$$

Com isso, se o esforço realmente depende do tamanho, o Resíduo SS será menor e as diferenças entre os valores preditos de esforço e o valor médio de esforço (Modelo SS) será próximo ao Total SS.

O valor dado por r^2, é a lógica subjacente à medida de precisão do modelo de regressão. O valor de r^2 é calculado pela Equação 3.14.

$$r^2 = \frac{Modelo\ SS}{Total\ SS} \tag{3.14}$$

Esse é o valor r^2, que significa a fração da variação nos dados que pode ser explicada pelo modelo; pode variar entre 0 e 1; e mede o ajuste da equação de regressão. Se o *Modelo SS* é quase o mesmo que *Total SS*, então r^2 será próximo de 1. Um valor de r^2 próximo de 1 indica que a linha de regressão ajusta bem os dados. Em regressão simples, o r^2 é também o quadrado do coeficiente de correlação de Pearson, *r,* entre duas variáveis.

3.4.3. Regressão múltipla

A regressão múltipla é basicamente a mesma que a regressão simples, mas ao invés do modelo ser uma linha reta, ele é uma equação com mais que uma variável independente. Como resultado, os cálculos são mais complexos. Além disso, quando tiver mais que três dimensões, não será possível visualizar o

relacionamento (por exemplo, com duas variáveis independentes). Em regressão múltipla, a estatística R^2 é chamada de coeficiente de determinação múltipla.

A desvantagem com esta técnica é sua vulnerabilidade para valores extremos (exceções), embora técnicas de regressão robustas, que são menos sensíveis a tais problemas, tenham sido aplicadas com sucesso (Briand e Wieczorek, 2002). Outro problema é o impacto da co-linearidade – a tendência de variáveis independentes serem fortemente correlacionadas umas com as outras – sobre a estabilidade de um sistema de predição baseado em regressão.

3.4.4. Significância dos resultados

Tanto em regressão simples como na regressão múltipla, o passo final é determinar se o resultado é significante. O modelo é significante? Os coeficientes de cada variável e a constante são significantes? O que significa significante? Significância é melhor explicada como segue: Quanto mais baixa a probabilidade que os resultados sejam conseqüência do acaso, maior será sua significância. A probabilidade está relacionada ao tamanho da amostra (ou seja, o número de projetos que a base de dados de históricos contém) e ao número de variáveis utilizadas para modelar a variável dependente. Diferentes distribuições, denominadas distribuição-F e distribuição-t, são utilizadas para determinar essas probabilidades. Geralmente, considera-se significante um valor de probabilidade menor ou igual a 0.05. Este foi o valor considerado nesse trabalho.

Uma das saídas da análise de regressão é apresentada em uma tabela, que é conhecida como a tabela de análise de variância (ANOVA). Os títulos das colunas são definidos como: SS = soma dos quadrados; df = grau de liberdade; e MS = quadrado médio. Assim, são apresentados o erro quadrático médio (resíduos MS), a soma total dos quadrados (Total SS) e a soma de quadrados

contados para o modelo (Modelo SS). Este último é definido como a soma dos quadrados (Residual SS), dividido pelos correspondentes graus de liberdade (Resíduos df).

Outras saídas da análise de regressão incluem: o número de projetos; a estatística F associada com a tabela ANOVA (considerando-se os graus de liberdade do modelo e dos resíduos); R^2, R^2-ajustado. A estatística F testa a hipótese nula de que todos os coeficientes, excluindo a constante, é zero. F é calculada com a Equação 3.15.

$$F = \frac{Modelo\ SS / grau\_de\_liberdade\_total}{Modelo\ SS / grau\_de\_liberdade\_residual} \tag{3.15}$$

O número de observações (ou seja, projetos) deve ser significativamente maior que o número de variáveis para que os resultados sejam confiáveis. O valor de R^2-ajustado é calculado pela Equação 3.16.

$$R^2\ ajustado = 1 - \frac{\left(1 - R^2\right)\left(grau\_de\_liberdade\_total\right)}{\left(grau\_de\_liberdade\_residual\right)} \tag{3.16}$$

Na análise de regressão, os graus_de_liberdade_total é *n – 1* e os graus_de_ liberdade_residual é *n-k*, em que *n* é o número de observações e *k é* o número de variáveis independentes +1 (para o termo constante).

Um importante passo em modelagem baseada em regressão é identificar os fatores que causam variação entre o esforço estimado e o esforço real. Uma análise de fator identifica esses parâmetros. São atribuídos pesos a estes parâmetros e eles são adicionados ao modelo como direcionadores de esforço. O modelo COCOMO original de Barry Boehm continha 15 multiplicadores de esforço, para os quais Boehm forneceu pesos multiplicadores relevantes. Barcelos Tronto et al (2007b) sugerem um modelo de estimativa de esforço a partir do conjunto de dados do COCOMO, que minimiza a estimativa do erro padrão nos dados. A abordagem foi experimentada sobre os dados do COCOMO e sobre outros dois conjuntos de dados de empresas brasileiras: dados do grupo de desenvolvimento de software do DSS/ NPE e da CELEPAR.

3.5 Técnicas estatísticas combinadas

Esta seção descreve um procedimento de análise de dados baseado em análise de regressão, análise de variância e análise de resíduos, proposto por Kitchenham (1998). Antes de descrever o procedimento de análise de dados é necessário apresentar alguma terminologia estatística. Análise de variância é o método comumente utilizado para decidir se os diferentes níveis de um fator afetam uma variável resposta (por exemplo, o esforço). Para um único fator, que pode ter muitos níveis, supõe-se que o esforço despendido sobre um projeto pode ser representado de acordo com o modelo dado pela Equação 3.17.

$$x_{ij} = \mu_i + \varepsilon_{ij} \tag{3.17}$$

em que, x_{ij} é o valor de uma variável resposta para o j-ésimo projeto no i-ésimo nível de um fator; μ_i é o valor médio da variável resposta para o i-ésimo nível do fator; ε_{ij} é o termo de erro, o qual tem uma distribuição normal com média 0 e variância σ^2.

O modelo representado pela Equação 3.17 corresponde a representar o impacto de cada nível do fator em termos da média do nível do fator.

3.5.1. Análise de variância

A análise de variância (ANOVA) permite testar se existe ou não diferenças significantes entre as médias dos níveis de um fator. Ela se baseia na suposição de que a variância das médias dos níveis será grande se existir diferenças entre os grupos e pequena se não existir. Com isto, a variância das médias dos grupos (chamada de variância entre grupos) é comparada com a variância dentro do grupo (que é uma estimativa de σ^2). A variância entre grupos é dividida pela variância dentro do grupo para dar o valor referido como a estatística F. Se as médias dos grupos (isto é, as médias dos níveis) não são significativamente diferentes umas das outras, a estatística F será próxima de

um. As tabelas estatísticas estão disponíveis para identificar se a estatística F é significativamente maior que um.

Essas tabelas são indexadas de acordo com o nível de significância e do grau-de- liberdade do teste (em inglês: *degree-freedon – df*). Grau-de-liberdade (df) identificam a quantidade de informação fornecida pela estatística de teste (geralmente, quanto maior o grau-de-liberdade, mais confiável será a estatística de teste). Existem dois graus-de-liberdade: um é associado com a variância entre grupos e o outro com a variância dentro do grupo. O grau-de-liberdade entre grupos é calculado pela Equação 3.18.

$$df\ \_entre\_grupos = (número\_de\_níveis - 1) \tag{3.18}$$

O grau de liberdade dentro do grupo é calculado pela Equação 3.19.

$$df\ \_dentro\_grupo = (número\_de\_projetos - 1) - (número\_de\_níveis - 1) \tag{3.19}$$

A variância dentro e entre grupos é geralmente referida como o quadrado da média dentro do grupo (erro) e entre grupos (MS-Mean Square). A variância entre grupos é calculada pela Equação 3.20 e 3.21.

$$MS(entre\_grupos) = \frac{Soma\ dos\ quadrados\ SS(entre\_grupos)}{df\,(entre\_grupos)} \tag{3.20}$$

$$Soma\ dos\ quadrados\ SS(entre\_grupos) = \sum n_i \left(m_{xi} - m_x\right)^2 \tag{3.21}$$

em que n_i é o número de projetos no i-ésimo nível do fator; m_{xi} é a média do i-ésimo nível; m_x é a média calculada sobre todos os projetos.

A variância dentro do grupo é calculada pelas Equações 3.22 e 3.23.

$$MS(dentro\_de\_grupos) = \frac{[SS(total) - Soma\ dos\ quadrados\ SS(entre\_grupos)]}{df\_dentro\_grupo)} \quad (3.22)$$

$$SS(total) = \sum (X_{ij} - m_x)^2 \quad (3.23)$$

Geralmente, os resultados de uma análise de variância são mostrados em uma tabela, como mostrado na Tabela 3.1, que é baseada na análise de variância dos cinco níveis do Fator TIME do conjunto de dados COCOMO. A estatística F é significante ao nível de probabilidade igual a 0,01, tal que, ignorando-se os outros fatores, a ANOVA indica que o Fator TIME tem um efeito significante sobre o valor de esforço.

Tabela 3. 1 - Análise de variância para o fato TIME

	SS	df	MS	F
TIME	35,64	4	8,91	3,30
ERRO	126,76	47	2,70	
TOTAL	162,40	51		

Pacotes estatísticos, como Excel e Statistica, possibilitam facilidades na construção da Tabela ANOVA a partir de um conjunto de dados.

3.5.2. Análise de resíduos

A análise de resíduos tem se mostrado útil quando se tem um modelo obtido através da aplicação de métodos estatísticos como a análise de regressão e a ANOVA. Resíduos são discutidos na maioria dos textos que tratam de modelos lineares e de regressão, por exemplo, no trabalho de Draper e Smith (1966). No caso de um modelo linear ajustado por ANOVA, os resíduos são calculados pela Equação 3.24.

$$r_{ij} = x_{ij} - m_{xi} \quad (3.24)$$

em que r_{ij} é o resíduo correspondente ao j-ésimo projeto no i-ésimo nível do fator; x_{ij} é o valor de esforço do j-ésimo projeto no i-ésimo nível do fator e; m_{xi} é o esforço médio para o i-ésimo nível do fator.

Os resíduos devem ser representados graficamente em função dos dados brutos a fim de investigar as propriedades do modelo ajustado. Se o modelo ajustado é um modelo simples baseado em dois ou mais níveis discretos e ele é um bom ajuste para os dados, o padrão dos resíduos deveria aparecer como uma série de linhas paralelas com aproximadamente o mesmo tamanho. Se parecer que os pontos estão agrupados em torno de cada uma das linhas, isto é uma indicação de que um fator importante foi omitido do modelo. Se o padrão de resíduos é uma série de linhas paralelas de tamanho crescente, isto é uma indicação de que a variável resposta (ou seja, esforço) tem uma variância instável, ou seja, a variância é maior para a maior variável resposta do que para a variável resposta menor. Isto é um sério problema uma vez que ANOVA requer uma variância estável. Para superar este problema, a variável resposta precisa ser transformada de uma maneira que estabilize a variância.

3.5.3. O procedimento de análise

Uma forma de tratar o problema da análise de um conjunto de dados com um grande número de fatores é usar um método baseado em análise de resíduos. Para a avaliação do esforço usando um conjunto de dados com vários fatores de escala nominal e ordinal, o procedimento de análise é composto dos seguintes passos:

1) Se necessário, transforme a variável correspondente ao esforço para melhorar a normalidade, em particular, a estabilidade da variância. Transformações logarítmicas, geralmente, são efetivas. Para obter mais informação sobre transformações logarítmicas consulte Hoaglin et al. (1983) ou Atkinson (1983).
2) Aplicar análise de variância (ANOVA) simples usando cada fator (para as variáveis de escala ordinal ou nominal) e análise de regressão simples para cada variáveis de escala absoluto.
3) Identificar o fator mais significante, ou seja, a partir de um conjunto fatores que têm um efeito significante estatisticamente sobre o esforço, escolher aquele que tem o menor termo de erro.

4) Remover o efeito do fator mais significante (ou seja, para cada nível do fator, subtrai-se o valor do esforço médio do valor do esforço do projeto) e obtém os resíduos.
5) Aplicar ANOVA usando cada fator restante sobre os resíduos. É preciso reduzir os graus-de-liberdade aplicados ao termo de erro porque K-1 gruas-de-liberdade foram utilizados para remover o impacto do fator mais significante, em que K é o número de níveis do fator mais significante.
6) Eliminar os fatores redundantes. Os fatores que foram considerados significantes na análise original, mas não tiveram o efeito significante sobre os resíduos são fatores que são confundidos com o fator mais significante da análise anterior. Fatores confundidos devem ser identificados e excluídos da análise subseqüente. Embora se tenha o risco de um fator ser removido sem necessidade, é importante reconhecer que através do uso deste procedimento pode-se obter um modelo aceitável. Ele reduz o risco de acontecer certos problemas de análise e pode ser a base para pesquisas futuras, mas como qualquer outra técnica de análise, ele não pode garantir a geração de um modelo real.
7) Identificar o próximo fator mais significante. Somente os fatores que têm um impacto significativo baseado na análise de variância sobre os resíduos devem ser considerados, e novamente, se existirem vários fatores candidatos, escolha aquele que tiver o menor termo de erro.
8) Remova o efeito do fator mais significante e obtenha o segundo resíduo.
9) Repita o procedimento de análise até que todos os fatores significantes sejam removidos, ou até que não existam graus-de-liberdade suficientes para continuar. Deve-se assegurar de que existem pelo menos quatro graus-de-liberdade disponíveis para o termo de erro e preferencialmente mais que dez.

Fatores do tipo absoluto e intervalo podem ser facilmente incluídos no procedimento de análise. O impacto deles sobre a variável resposta pode ser

avaliado usando regressão linear e calculando-se o quadrado médio devido à equação de regressão. Se um fato do tipo absoluto ou intervalo for considerado como o mais significante, deve-se remover seu efeito e continuar analisando os resíduos. Nesse caso, os graus-de-liberdade atribuídos ao termo de erro na análise subseqüente devem ser reduzidos por um. É importante notar que em estatística, os termos variáveis e fatores são utilizados indistintamente.

Na Seção 4.5 esse procedimento é aplicado sobre o conjunto de dados COCOMO de Boehm (1981), para ilustrar como se realiza a análise sobre um conjunto de dados real.

3.6 Considerações finais

Neste capítulo foram apresentadas algumas abordagens que podem ser utilizadas na estimativa de esforço de software. Para melhorar a precisão das estimativas é preciso seguir um procedimento específico para cada abordagem. O próximo capítulo apresenta alguns estudos de casos realizados como parte desta tese, que mostra como gerar um modelo de estimativa de esforço utilizando cada uma das abordagens aqui apresentadas.

4 DADOS, ANÁLISE E RESULTADOS DE ESTIMATIVA DE ESFORÇO

4.1 Considerações iniciais

Neste capítulo são apresentados três estudos de casos conduzidos utilizando o conjunto de dados COCOMO de forma a examinar o impacto de alguns fatores sobre o esforço de desenvolvimento de software e investigar a precisão das estimativas. A seguir são descritos os estudos de casos realizados como parte desta pesquisa e os resultados obtidos.

4.2 Metodologia de modelagem

O problema considerado é gerar um modelo que possa auxiliar um gerente de projetos a predizer a quantidade de esforço utilizado para o desenvolvimento de um novo projeto de software. Quando uma empresa participa de uma licitação de um contrato ou mesmo desenvolve um software na própria organização, realizar estimativas precisas de esforço é uma atividade crítica. Embora existam muitas ferramentas de estimativas comerciais disponíveis, não se pode esquecer de que elas foram desenvolvidas usando dados de companhias que podem ser bastante diferentes da organização em questão.

Para melhorar a precisão, é preciso calibrar estas ferramentas com os dados da própria companhia. Uma vez que a organização coletou os dados, porque a própria empresa não os analisa? Quais os fatores que influenciam o esforço de projetos em uma determinada organização? O modelo gerado com os dados da própria organização pode resultar em estimativas mais confiáveis? Através dos estudos de casos realizados nesta tese se espera contribuir para que uma organização possa realizar a análise de seus próprios dados e calibrar seu próprio modelo de estimativa de esforço ou ainda, se julgar adequado, utilizar um dos modelos propostos.

Antes de coletar dados sobre os parâmetros para um modelo de estimativa é preciso definir cuidadosamente os parâmetros. Além disso, é importante ter

uma idéia de como os parâmetros devem ser utilizados para produzir estimativas efetivas. Isto significa que é preciso determinar quais parâmetros poderão influenciar significativamente o esforço a ser estimado. Variáveis categóricas como restrição de tempo, experiência da equipe, restrição no cronograma, tipos de aplicação e volatilidade dos requisitos podem ser parâmetros importantes para explicar o esforço e o custo de projetos de software de uma companhia.

A utilização de estatística na geração destes modelos pode ser considerada tanto uma arte quanto uma ciência. Escolher os métodos estatísticos apropriados, selecionar as variáveis a usar, criar novas variáveis, remover *outliers,* gerar o melhor modelo, detectar variáveis que se confundem umas com as outras e escolher as variáveis categóricas requer a tomada de muitas decisões durante o processo de análise de dados. Freqüentemente, não se têm regras claras para estas decisões.

Assim, nesta tese os estudos conduzidos utilizam o seguinte conjunto de passos para analisar os dados de projetos e desenvolver e medir a precisão de modelos de estimativa de esforço de software: 1) definição do conjunto de dados; 2) preparação dos dados; 3) geração do modelo; 4) análise da precisão. Antes da geração do modelo é importante selecionar a técnica a ser utilizada.

Como resultado é apresentado um conjunto de parâmetros que se mostraram influenciar o esforço de software, os quais foram identificados através dos estudos de casos descritos a seguir. Antes de realizar cada estudo de caso, foi executada uma revisão da literatura de modelagem de custo de software a fim de obter um conhecimento inicial sobre parâmetros potencialmente significantes e as questões relacionadas à definição de parâmetros (por exemplo, tamanho de software). O conjunto de dados de projetos de software COCOMO (descrito na próxima seção) foi selecionado para realizar todos os estudos de caso mostrados neste capítulo e mostrar como construir modelos de estimativa de esforço utilizando as diferentes técnicas.

4.3 Definição do conjunto de dados

A análise empírica abordada nesta tese foi realizada sobre uma base de dados de projetos de software que tem sido utilizado em estudos anteriores na literatura de engenharia de software (Sentas et al, 2005; Srinivasan e Fisher, 1995; Samson et al, 1997). Este conjunto de dados foi selecionado porque está disponível para uso público e foi utilizado para descrever e testar um dos mais importantes métodos na área, que é o modelo COCOMO (Boehm,1981; Boehm et al, 2000). O conjunto de dados consiste de 63 projetos incluindo instâncias de projetos de software de negócio, científico e de sistema, escritos em uma variedade de linguagens de programação que incluem COBOL, PLI, HMI e FORTRAN, conforme mostrado no Apêndice A. As variáveis multiplicadoras de esforço de desenvolvimento de software (e uma breve descrição delas) consideradas neste estudo são apresentadas na Tabela 4.1.

Tabela 4. 1 - Multiplicadores de esforço de desenvolvimento de software

Variável	**Descrição**
TYPE	Tipo de aplicação
LANG	Linguagem de programação
RELY	Confiabilidade requerida do software
DATA	Tamanho da base de dados
CPLX	Complexidade do produto
TIME	Restrição de tempo de execução
STOR	Restrição de armazenamento principal
VIRT	Volatilidade da máquina virtual
TURN	Tempo de execução da máquina
ACAP	Capacidade do analista
AEXP	Experiência com aplicações
PCAP	Capacidade do programador
VEXP	Experiência com máquina virtual
LEXP	Experiência com linguagem de programação
MODP	Uso de práticas modernas de programação
PLATFORM	Plataforma de desenvolvimento
TOOL	Uso de ferramentas de software
SCHED	Cronograma de desenvolvimento requerido
RVOL	Volatilidade dos requisitos
MODE	Modo de desenvolvimento
ADJKDSI	Número de linhas de código ajustadas

A variável ADJKDSI é do tipo numérico contínuo enquanto que as demais variáveis são categóricas, em que cada categoria é representada por um valor numérico.

Todos os 63 projetos foram utilizados na análise realizada. A variável dependente, considerada neste estudo, é ACTMM (a quantidade de esforço total, em número de homens/ hora, despendido no desenvolvimento do projeto). De uma maneira geral, para cada um dos experimentos os dados históricos são divididos em amostras usadas para treinar o sistema de aprendizagem e amostras separadas para testar a precisão do classificador treinado para estimar o esforço de desenvolvimento.

As estimativas de precisão das predições obtidas a partir de um conjunto de dados de treinamento são sempre otimistas. De forma a investigar a precisão dos modelos construídos nesta tese, utilizou-se o mesmo procedimento de (Kitchenham, 1998). Kitchenham desenvolveu um procedimento simples para investigar a precisão das predições realizadas pelo seu modelo sobre a base de dados COCOMO. Baseando-se neste processo, omitiu-se um subconjunto de projetos (o conjunto de dados de teste), depois se desenvolveu um modelo com os projetos restantes (o conjunto de dados de treinamento) e finalmente avaliou-se a precisão das predições do modelo sobre o conjunto de dados de teste. Desta forma, foram criados seis diferentes pares de conjuntos de treinamento e de teste. Cada conjunto de treinamento foi construído removendo-se todo sexto projeto, a começar (da primeira vez) do primeiro projeto. Assim, o conjunto de dados de aprendizagem 1 foi construído removendo-se os projetos: 1, 7, 13, 19, 25, 31, 37, 43, 49, 55 e 61; o conjunto de dados de aprendizagem 2 foi construído removendo-se os projetos: 2, 8, 14, 20, 26, 32, 38, 44, 50, 56, 62 e assim sucessivamente. Cada conjunto de dados removidos representa o respectivo conjunto de dados de teste.

Sabendo-se que todos os 63 projetos da base de dados COCOMO foram utilizados para construir o modelo, dos seis pares de conjuntos criados três

conjuntos de dados de aprendizagem contêm 52 projetos e os outros três contêm 53 projetos.

4.4 Preparação dos dados

Esta seção descreve a atividade de preparação da base de dados COCOMO, a fim de ajustá-la às necessidades impostas pelas técnicas de estimativa utilizadas nesta tese. Os estudos de casos utilizaram estes dados pré-processados.

Esse pré-processamento exclui ajustes numéricos para cada fator, em cada projeto, ao invés daqueles níveis de fatores específicos usados no modelo COCOMO. O conjunto de dados foi adaptado para a análise substituindo-se os ajustes numéricos com um nível de fator equivalente. Além disso, foi aplicado um procedimento para converter os fatores de ajuste para níveis. Por exemplo, para o fator restrição de tempo "TIME", foi aplicado o procedimento apresentado na Tabela 4.2. É importante observar que na literatura de estatística os termos variáveis e fatores são freqüentemente utilizados como sinônimos.

Tabela 4. 2 - Os níveis para o fator TIME

Fator de ajuste usado em Boehm (1981)	Nível atribuído
1.0	1
1.06, 1.07, 1.08	2
1.11, 1.15	3
1.27, 1.30, 1.35	4
1.46, 1.66	5

Uma vez que análise de regressão e a análise de variância são métodos estatísticos paramétricos e requer uma distribuição normal aproximada, o primeiro passo é avaliar se o esforço (a variável resposta dado por MMACT) tem uma distribuição normal. Existem diferentes formas de decidir se um conjunto de valores tem distribuição normal: a) Os testes formais de normalidade (Shapiro e Wilk, 1965); b) Métodos visuais simples tais como

gráfico de probabilidade normal (Hogg e Ledolter, 1992). A Figura 4.1 mostra o efeito de se produzir uma representação gráfica da probabilidade normal para os valores de esforço bruto do segundo conjunto de dados de treinamento.

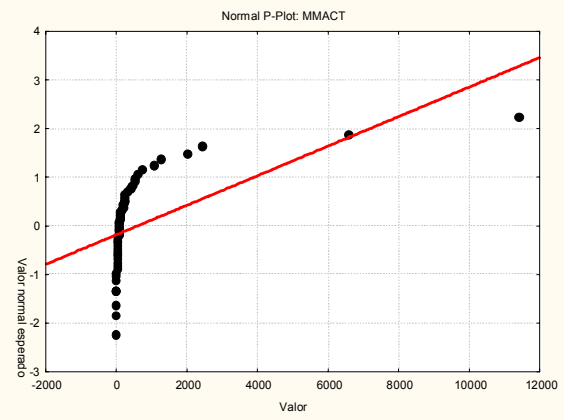

Figura 4. 1 - Diagrama de probabilidade normal para os dados brutos.

O diagrama de espalhamento dos dados é claramente não linear. Assim, é razoável assumir que os dados não têm uma distribuição normal. De forma a melhorar a normalidade é comum transformar os dados brutos. Quando se quer estabilizar a variância, uma transformação natural, ou seja, *log* na base *e,* é efetiva (para mais informações sobre transformações vejam Hoaglin et al. (1983) ou Atkinson (1983)). A Figura 4.2 mostra o diagrama de probabilidade normal após a aplicação de uma transformação logarítmica sobre os valores de esforço do segundo conjunto de treinamento.

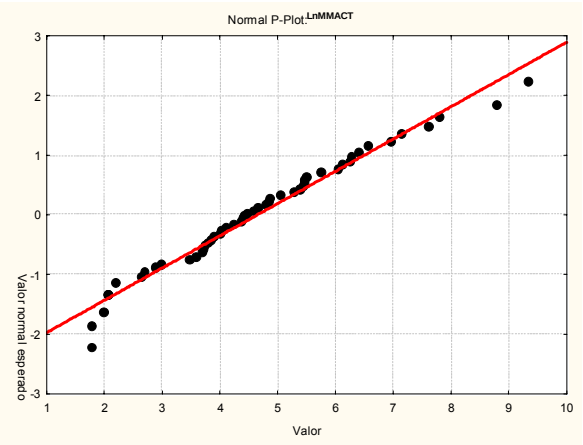

Figura 4. 2- Diagrama de probabilidade normal para os dados transformados.

É fácil notar que o diagrama de espalhamento dos dados é muito mais linear. Portanto, o procedimento de análise mostrado nos estudos de casos é realizado sobre os dados transformados. A variável que representa o esforço (MMACT) foi transformada para melhorar a normalidade. O conjunto de dados transformados é apresentado no Apêndice B.

Além disso, foi realizada uma atividade de preparação das variáveis (fatores) independentes utilizadas neste trabalho. Neste sentido, foi aplicada a abordagem de análise de variância (ANOVA), de um único fator, sobre o conjunto de dados COCOMO a fim de ajustar os níveis de cada variável independente (fator). Análise de variância é um método, geralmente aplicado para decidir se diferentes níveis de um fator afetam a variável resposta (neste caso, esforço). Esta análise permite testar se existem diferenças significantes entre as médias dos níveis de um fator. Para o propósito deste trabalho, restringe-se às médias dos níveis de cada fator. Todos os 63 projetos foram utilizados na análise e a maioria das variáveis é categórica (fatores).

Cada uma das variáveis categóricas foi submetida a um processo ANOVA de um único fator de forma a verificar o impacto de cada fator sobre a variável dependente original e identificar as categorias que precisam ser concatenadas em cada fator. Através desses testes foi possível comparar cada categoria (nível) de um fator com todas as outras categorias no mesmo fator e determinar a significância da diferença. Baseado no resultado obtido dessa análise, as categorias de cada variável independente que não são significantemente diferentes foram concatenadas e recodificadas em grupos homogêneos para cada fator. As variáveis categóricas recodificadas são apresentadas na Tabela 4.3. Esta abordagem foi baseada no trabalho de Angelis et al. (2001).

Quanto à preparação das variáveis numéricas, a única variável independente numérica considerada nos estudos de casos foi o tamanho da aplicação, identificada como ADJKDSI. Esta variável foi pré-processada através de uma transformação logarítmica, por permitir melhor ajuste e resultados de predição.

Tabela 4.3 - Variáveis categóricas recodificadas

Fator	Nível Original	Nível concatenado	Nome do nível
MODP	1	1	Muito alto
	2 e 3	2	Alto
	4	3	Médio
	5	4	Baixo
	6	5	Muito baixo
PCAP	1	1	Muito alto
	2 e 3	2	Alto
	4	3	Médio
	5, 6, 7	4	Abaixo da média
RVOL	1	1	Altamente estável
	2 e 3	2	Estável
	4	3	Instável
	5 e 6	4	Altamente instável
TOOL	1	1	Muito alto
	2 e 3	2	Alto
	4	3	Médio
	5	4	Baixo
	6	5	Muito baixo
RELY	1	1	Muito alto
	2 e 3	2	Alto
	4	3	Médio
	5	4	Baixo
	6	5	Muito baixo
ACAP	1	1	Muito alto
	2 e 3	2	Alto
	4	3	Médio
	5 e 6	4	Baixo
	7	5	Muito baixo

4.5 Estudo de caso usando análise de regressão

Esta seção descreve o estudo de caso utilizando análise de regressão *stepwise* realizado à luz da metodologia apresentada anteriormente. Existe, é claro, outros tipos de modelos de estimativa, mas a escolha de um modelo de regressão é motivada principalmente pela necessidade de se ter um modelo simples que dê suporte a uma tentativa preliminar de entender o processo de estimativa de esforço de software.

4.5.1. Geração dos modelos de regressão

Foram conduzidos vários experimentos utilizando análise de regressão sobre cada um dos seis conjuntos de dados de treinamento obtidos a partir do conjunto de dados COCOMO. A primeira etapa dos experimentos ocorreu sem

realizar o pré-processamento dos dados. Porém, por a regressão ser um método paramétrico, é necessário que os dados numéricos apresentem uma distribuição normal. Assim, a segunda etapa utilizou os dados pré-processados. Os resultados mostram que a confiabilidade e a precisão dos resultados pode ser afetada pela forma com que os dados são preparados. Foram utilizadas análise de regressão simples (em que a única variável independente é aquela que se refere ao tamanho da aplicação) e análise de regressão múltipla (em que as variáveis independentes são aquelas apresentadas na Tabela 4.1).

Primeira etapa: modelos gerados sem o pré-processamento de dados

A partir de estudos da literatura e de análises estatísticas utilizando ANOVA, verificou-se que a variável que está mais relacionada ao esforço de desenvolvimento de software é aquela que representa o tamanho da aplicação. Com base nessa informação, o primeiro experimento realizado nesta etapa foi gerar um modelo de regressão linear simples para estimar o esforço de desenvolvimento de software para cada um dos conjuntos de treinamento obtidos a partir do conjunto de dados COCOMO. Os modelos de regressão linear simples foram calibrados usando o método de regressão *stepwise* e tomando-se a variável MMACT (o esforço) como variável dependente e a variável ADJKDSI (o tamanho) como variável independente. Depois de vários experimentos obteve-se os modelos finais de regressão apresentados na Tabela 4.4.

Tabela 4. 4 - Modelos de regressão simples

Modelo	**Equação de Regressão**
Modelo 1	ACT MM = -173,273809+14,733878*ADJKDSI
Modelo 2	ACT MM =52,053251+8,957939*ADJKDSI
Modelo 3	ACT MM= 92,959369+ 9,265678*ADJKDSI
Modelo 4	ACT MM=99,139520+ 9,215841*ADJKDSI
Modelo 5	ACT MM =98,1242505272844+9,18249062297309*ADJKDSI
Modelo 6	ACT MM= 70,6468778556116+7,60319621866261*ADJKDSI

As estimativas dos modelos de regressão simples (obtidos da calibração sobre os conjuntos de treinamento) usando os conjuntos de teste são mostradas na Tabela 4.5.

Tabela 4. 5- Esforço estimado através dos modelos de regressão simples

Conjunto1	**Conjunto2**	**Conjunto3**	**Conjunto4**	**Conjunto5**	**Conjunto6**
1491,65	2283	1316	523,1	245,044	101,06
-71,61	249,1	370,9	265	281,774	351,965
180,339	78,93	129,1	133,2	115,571	2503,67
14059,7	2623	2428	1104	786,811	754,935
386,614	482	180,1	218,9	117,775	85,7012
563,42	2390	463,6	301,9	217,497	161,885
327,678	186,4	150,4	122,2	146,791	218,909
239,275	320,8	389,5	624,4	309,322	2435,24
1167,51	267	185,6	174,7	146,791	104,101
-80,45	293,9	231,9	338,8	290,957	121,588
239,275	133,6	185,6			

Apesar de se ter o conhecimento de que o tamanho da aplicação tem uma influencia de maior peso sobre o esforço, sabe-se também que algumas variáveis têm grande impacto sobre essa métrica quando consideradas juntas. O objetivo é encontrar um conjunto de variáveis que contribuam para se realizar estimativas mais confiáveis. Assim, vários experimentos foram realizados utilizando a técnica de regressão múltipla e o método *stepwise forward* (com um nível de significância igual a 0,05) para calibrar modelos para os seis conjuntos de dados de treinamento em questão. Cada um desses modelos foi treinado sobre seu respectivo conjunto de dados de treinamento, considerando ACTMM como a variável dependente e tomando-se todas as variáveis mostradas na Tabela 4.1 como variáveis independentes.

Os experimentos foram realizados utilizando um algoritmo que encontra o melhor modelo a partir de um determinado número de possíveis modelos. A estatística conhecida como R^2_ajustado, dos subconjuntos (modelos), permitiu

comparações diretas e a escolha do “melhor” subconjunto entre os vários subconjuntos encontrados (Maxwell, 2002). Para cada conjunto de dados de treinamento escolheu-se o modelo (dentre dez modelos gerados) com o maior valor de R^2_ajustado. Conseqüentemente, foram obtidos seis modelos. Apesar dos seis conjuntos de treinamento serem derivados de uma mesma base de dados de projetos (a base de dados COCOMO), os respectivos modelos encontrados são constituídos de variáveis diferentes entre si.

Os coeficientes β da regressão permitem avaliar a contribuição de cada variável independente na estimativa global da variável dependente. Quanto maior o valor de β maior é a contribuição da variável no modelo (Clark, 2001). Nesse experimento todos os modelos encontrados possuem algumas variáveis com um valor de β considerado não significativo. Como o objetivo é obter uma maior precisão de estimativa, aplicou-se uma regressão múltipla *stepwise forward* (Maxwell, 2002) tomando como variáveis independentes, aquelas contidas no modelo. Esta regressão foi utilizada para cada um dos seis modelos de forma que somente as variáveis com um valor de β significativo fossem consideradas no modelo. A regressão *stepwise forward* constrói um modelo de predição, adicionando-se ao modelo (a cada estágio) a variável com a maior correlação parcial com a variável dependente, considerando-se todas as variáveis presentes no modelo. O objetivo é encontrar um conjunto de variáveis independentes que maximizam a estatística F, que avalia se as variáveis independentes, quando consideradas juntas, são significativamente associadas com a variável dependente. O critério utilizado para adicionar uma variável é sua influência no aumento do valor de F para a regressão, por alguma quantidade *k*. Uma variável que reduz F, também por alguma quantidade *w*, deve ser removida do modelo. Os modelos são apresentados na Tabela 4.6.

Tabela 4. 6 - Modelos de regressão múltipla sem pré-processamento dos dados

Modelo	**Equação de Regressão**
Modelo 1	ACTMM = -2937,57453363+16,1182817ADJKDSI+2410,01478707*CPLX
Modelo 2	ACTMM = -5558,849858+ 7,980057*ADJKDSI+ 5654,065476*DATA
Modelo 3	ACTMM= -3709,2565+3757,96154*MODP+9,05218371*ADJKDSI
Modelo 4	ACTMM= -8835,5111+5396,35474*DATA+3638,30337*MODP+7,77161528*ADJKDSI
Modelo 5	ACTMM = -3682,0398+3802,92414*MODP+8,92926545*ADJKDSI
Modelo 6	ACTMM= -4074,3481+1394,65051*RELY+2308,47763*VEXP +2535,60316*MODP-2129,0629*TOOL+7,29998110*ADJKDSI

As estimativas dos modelos de regressão múltipla (obtidos da calibração sobre os conjuntos de treinamento), usando os conjuntos de teste, são mostradas na Tabela 4.7.

Tabela 4. 7- Estimativas usando regressão múltipla sem pré-processamento de dados

Conjunto1	**Conjunto2**	**Conjunto3**	**Conjunto4**	**Conjunto5**	**Conjunto6**
570,8	2986,9	905,4	2293,3	1176,5	-119,4
-416,3	-68,5	-17,9	15,3	299,5	286,4
582,3	-220,1	-254,2	227,9	137,8	2976,2
15042,7	2837,8	1991,6	718,8	790,6	661,3
807,9	478,3	-542,6	1095,7	-202,3	-154,2
278,4	2630,3	72,6	474,4	1149,7	-83,7
-702,5	214,9	-233,4	258,6	-174,1	-240,9
92,5	560,8	338,4	857,9	706,5	2161,0
577,7	286,7	1041,1	812,3	168,2	-373,7
-1149,1	-28,6	560,3	-250,1	-33,9	564,9
285,3	-171,4	-537,2			

Como os dados utilizados nesta fase do experimento não apresentam uma distribuição normal, estas estimativas estão todas sujeitas a erros e devem ser interpretadas com cautela. Assim, sugere-se que um novo modelo de predição seja calibrado utilizando os dados pré-processados apresentados na Seção 4.4, a fim de que esse modelo possa ser utilizado com segurança para realizar estimativas.

Segunda etapa: modelos gerados após o pré-processamento de dados

Nessa etapa é mostrado como a análise baseada em regressão múltipla trabalha sobre o conjunto de dados COCOMO após o pré-processamento dos dados. Vários experimentos foram conduzidos para predizer o esforço de software sobre os seis conjuntos de dados de treinamento derivados do conjunto de dados COCOMO.

Como análise de regressão é uma técnica paramétrica, o primeiro passo realizado foi verificar se o esforço (dado pela variável MMACT) é normalmente distribuído. Conforme descrito na Seção 4.4, a variável esforço foi transformada para melhorar a normalidade. Os modelos de regressão foram desenvolvidos aplicando-se regressão *stepwise* com eliminação *forward.* Nesse processo de geração do modelo, um valor de β igual a 0,05 foi utilizado para a entrada de variáveis consideradas importantes no modelo. Cada um dos modelos foi calibrado sobre seu respectivo conjunto de dados de treinamento, considerando o logaritmo na base neperiana da variável MMACT - Ln(MMACT) - como a variável dependente. As variáveis independentes são todas aquelas variáveis categóricas mostradas na Tabela 4.1, sendo que a variável que representa o tamanho do software é Ln(ADJKDSI).

A análise estatística foi executada usando um algoritmo que encontra o melhor modelo a partir de um determinado número de possíveis modelos. A estatística conhecida como R^2_ajustado, dos subconjuntos (modelos), permitiu comparações diretas e a escolha do "melhor" subconjunto entre os vários subconjuntos encontrados. Para cada conjunto de dados de treinamento escolheu-se o modelo (dentre dez modelos gerados) que tem o maior valor de R^2_ajustado. Conseqüentemente, foram obtidos seis modelos.

Apesar dos seis conjuntos de treinamento serem derivados de uma mesma base de dados de projetos (a base de dados COCOMO), os respectivos

modelos encontrados são constituídos de variáveis diferentes entre si. As únicas variáveis presentes em todos os modelos são Ln(ADJKDSI) e o RVOL.

Todos os modelos encontrados possuem algumas variáveis com um valor de β considerado não significativo, considerando-se um nível de significância igual a 0,05. Como o objetivo é obter uma maior precisão de estimativa, aplicou-se uma regressão múltipla *stepwise forward* tomando como variáveis independentes, somente aquelas com um valor de β significativo no modelo. A regressão *stepwise forward* constrói um modelo de predição, adicionando-se ao modelo (a cada estágio) a variável com a maior correlação parcial com a variável dependente, considerando-se todas as variáveis presentes no modelo.

O objetivo é encontrar um conjunto de variáveis independentes que maximizam a estatística F, a qual avalia se as variáveis independentes, quando consideradas juntas, são significativamente associadas com a variável dependente. O critério utilizado para adicionar uma variável é se ela aumenta o valor de F para a regressão, por alguma quantidade k. Quando uma variável reduz F, também por alguma quantidade w, ela é removida do modelo. A Tabela 4.8 mostra os modelos finais de regressão construídos para cada conjunto de aprendizado.

Os subconjuntos de teste foram utilizados para avaliar a precisão das estimativas realizadas através dos modelos mostrados na Tabela 4.8. Uma vez que os modelos são baseados nos dados transformados, os valores resultantes precisam ser transformados de volta para a escala dos dados brutos para evitar valores irrealistas. A Tabela 4.9 mostra os valores estimados para cada subconjunto de teste.

Tabela 4. 8 - Modelos de regressão múltipla após pré-processamento dos dados

	Equação de regressão
Conjunto 1	LnMMACT= -0,71831+0,208257*RELY+0,599015*RVOL +1,08353*LnADJKDSI
Conjunto 2	LnMMACT = -2,2070+0,224217*RELY+0,081843*DATA +0,218017*TIME+0,085651*STOR+0,390645*VIRT -0,23117*PLATFORM+0,204759*ACAP+0,277066*PCAP +0,301366*RVOL+1,03700*LnADJKDSI
Conjunto 3	LnMMACT= -1,6174+,286916*RELY+,332576*ACAP+,524855*RVOL +1,08734*LnADJKDSI
Conjunto 4	LnMMACT= -2,9387+0,149604* RELY+0,282402*TIME +0,524036*ACAP+0,311149*LEXP+0,188171*MODP +0,318291*RVOL+1,14161* LnADJKDSI
Conjunto 5	LnMMACT= -1,3577+0,137437*ACAP+0,616360*RVOL +0,541879*MODE+1,00858 *LnADJKDSI
Conjunto 6	LnMMACT= -2,4885+0,148765*RELY+0,282343*TIME +0,320646*ACAP+0,260250 *MODP+0,305558*RVOL +0,258748*MODE+1,07799*LnADJKDSI

Tabela 4. 9 - Estimativas usando regressão múltipla após pré-processamento de dados

Conjunto 1	**Conjunto 2**	**Conjunto 3**	**Conjunto 4**	**Conjunto 5**	**Conjunto 6**
748,1	1571,7	702,3	476,8	37,5	24,4
16,1	739,8	315,8	274,2	415,6	347,1
143,2	21,1	45,8	43,5	6,1	10284,0
7850,1	10560,5	1418,7	1072,7	459,5	263,8
780,9	342,2	124,7	141,9	5,3	3,1
390,7	926,8	340,4	227,8	166,7	52,6
61,4	34,5	18,1	7,7	12,3	27,8
137,4	148,5	118,2	209,1	41,1	2277,9
897,3	204,3	132,9	117,3	16,1	14,1
18,0	602,1	92,0	133,4	26,7	55,2
137,4	23,2	30,4			

4.5.2. Análise de precisão dos modelos de regressão

Neste trabalho o MMRE é adotado como indicador de desempenho de predição, uma vez que ele é amplamente utilizado e nos permite comparar os resultados aqui obtidos com aqueles obtidos por outros pesquisadores. A Tabela 4.10 apresenta os valores de MMRE obtidos através das estimativas realizadas sobre os seis conjuntos de dados de teste, utilizando os modelos de regressão linear simples (somente com uma variável independente, ADJKDSI) e os modelos de regressão múltipla gerados como resultado deste estudo.

Tabela 4. 10 - MMRE dos modelos de regressão sem o pré-processamento dos dados

	Conjunto1	**Conjunto2**	**Conjunto3**	**Conjunto4**	**Conjunto5**	**Conjunto6**
Regressão Múltipla	1372	354	865	843	1526	706
Regressão Simples	335	231	358	288,5	795	291

Nenhum dos modelos executa particularmente bem para estimativas de esforço de desenvolvimento de software, particularmente sob a análise do MMRE. Os resultados mostram que regressão múltipla executou pobremene em todos os experimentos (com exceção daquele realizado utilizando o conjunto de dados 2, que é constituído de dados de teste bem comportados e que todos os outros modelos também tiveram seu melhor desempenho) quando comparado com modelos anteriores utilizando somente uma única variável dependente: ADJKDSI.

A Tabela 4.11 apresenta os valores de MMRE resultantes da análise de precisão das estimativas utilizando os modelos de regressão múltipla gerados após o pré-processamento dos dados apresentados na Seção 4.4.

Tabela 4. 11 - MMRE das estimativas de regressão múltipla após pré-processamento

Conjunto1	**Conjunto2**	**Conjunto3**	**Conjunto4**	**Conjunto5**	**Conjunto6**
102	51	62	47	48	37

Esses resultados indicam que as estimativas realizadas utilizando análise de regressão múltipla após o pré-processamento apresentam um índice de precisão significativamente melhor que aquele obtido antes do pré-processamento. Com isso, pode-se inferir que o pré-processamento dos dados tem impacto substancial sobre a precisão dos modelos de estimativa de esforço de software para a abordagem de regressão múltipla.

4.6 Estudo de caso utilizando redes neurais artificiais

Esta seção descreve o estudo de caso utilizando redes neurais artificiais, realizado seguindo a metodologia apresentada na Seção 4.2. Existem, é claro, outros tipos de modelos de estimativa, mas a escolha de um modelo de rede neural é motivada principalmente pela necessidade de se ter um modelo capaz de tratar os relacionamentos não lineares presentes em um processo de estimativa de esforço de software e pela simplicidade de implementação dos modelos baseados em redes neurais.

4.6.1. Geração do modelo

Foram conduzidos vários experimentos utilizando redes neurais artificiais sobre cada um dos seis conjuntos de dados de treinamento obtidos a partir do conjunto de dados COCOMO. A primeira etapa dos experimentos ocorreu sem realizar o pré-processamento dos dados. Porém, apesar dessa técnica não exigir que os dados apresentem uma distribuição normal, a segunda etapa ocorreu utilizando os dados pré-processados, conforme descrito na Seção 4.4. O objetivo aqui é mostrar como a confiabilidade e a precisão dos resultados podem ser afetadas pela forma com que os dados são preparados para a análise. A geração dos modelos foi realizada utilizando o software MATLAB.

Primeira etapa: modelos gerados sem o pré-processamento de dados

Foram conduzidos vários experimentos utilizando redes neurais artificiais, sobre cada um dos seis conjuntos de dados de treinamento obtidos do conjunto de dados COCOMO.

Conforme apresentado na seção anterior, o tamanho da aplicação é a variável que está mais relacionada ao esforço de desenvolvimento de software, porém existem outros fatores que influenciam no esforço de desenvolvimento de software. Com base nessa informação, o objetivo nessa etapa foi gerar um modelo de redes neurais para estimar o esforço para cada um dos conjuntos de treinamento obtidos a partir do conjunto de dados COCOMO, utilizando todas as variáveis apresentadas na Tabela 4.1. Os modelos de redes neurais artificiais foram calibrados utilizando uma arquitetura multicamadas com algoritmo de aprendizado com retro-propagação do erro.

A fase de treinamento foi repetida várias vezes, em uma busca pela melhor rede para resolver o problema. Nesta fase várias arquiteturas de redes neurais foram experimentadas e os resultados aqui apresentados correspondem à rede neural com o melhor desempenho de generalização. Os dados foram normalizados em relação aos valores máximos que cada variável pode assumir. A rede neural resultante foi implementada com 17 entradas (a 18^{a} variável é o esforço a ser estimado, dado pela variável MMACT), 5 neurônios na primeira camada, 3 neurônios na segunda camada e um neurônio de saída para estimar o esforço (variável de saída). As variáveis de entrada são listadas na Tabela 4.1. As predições obtidas a partir de RNA (depois do treinamento sobre os seis conjuntos de dados de treinamento) usando os seis conjuntos de teste são mostradas na Tabela 4.12, sendo que “obs” representa os valores reais e “est” os valores estimados.

Tabela 4. 12 - Esforço estimado através de modelos de redes neurais

Conjunto1		Conjunto2		Conjunto3		Conjunto4		Conjunto5		Conjunto6	
obs	**est**	**Obs**	**est**	**obs**	**est**	**obs**	**est**	**obs**	**est**	**obs**	**est**
2040	401,9	1600	2246,3	243	265,4	240	544,9	33	111,9	43	162,6
8	224,8	1075	858,4	423	258,1	321	249,5	218	165,9	201	135,1
79	254,5	73	269,5	61	399,8	40	466,9	9	165,0	11400	6077,8
6600	10570,3	6400	10543,9	2455	523,2	724	293,3	539	119,0	453	281,9
523	432,8	387	229,2	88	223,6	98	266,5	7,3	118,2	5,9	94,5
1063	448,3	702	3781,9	605	360,7	230	201,0	82	103,7	55	104,3
47	234,0	12	191,3	8	154,3	8	135,5	6	79,5	45	86,0
83	287,1	87	244,0	106	179,2	126	191,4	36	82,5	1272	2864,4
156	293,4	176	301,5	122	384,2	41	226,3	14	95,1	20	101,9
18	254,1	958	378,1	237	212,0	130	492,3	70	95,4	57	145,2
50	237,6	38	181,4	15	147,8						

Um novo modelo de predição é calibrado e testado utilizando os dados transformados apresentados na Seção 4.4.

Segunda etapa: modelos gerados após o pré-processamento de dados

Esta seção mostra a análise baseada em redes neurais artificiais, realizada sobre o conjunto de dados COCOMO pré-processado. Vários experimentos foram conduzidos para predizer o esforço de software sobre os seis conjuntos de dados de treinamento derivados do conjunto de dados COCOMO.

O uso de redes neurais artificiais para estimar o esforço de desenvolvimento de software não requer um conhecimento e análise da distribuição dos dados. Os modelos baseados em redes neurais foram obtidos manipulando-se os dados através de experimentação, sendo que os dados são mapeados para um espaço unitário n-dimensional $[0,1]^n$, em que n é o número de variáveis de entrada. Esse processo é precedido por um passo de normalização cujo objetivo é minimizar o *bias* das variáveis de entrada que apresentam valores altos quando comparados a outras variáveis. A normalização é executada estabelecendo-se os valores máximos possíveis para cada variável de uma

forma realista. O processo de normalização pode também ser executado através do mapeamento dos dados para um hiper-espaço no intervalo de $[0,1]^n$. Uma análise visual do conjunto de dados COCOMO revelou que duas variáveis (tamanho e esforço) apresentam valores muito altos quando comparados com outras. Com isso, elas foram pré-processadas aplicando-se uma transformação logarítmica (conforme descrito na Seção 4.4) de forma a tornar sua faixa de valores compatível com as outras variáveis.

A arquitetura básica de redes neurais, desenvolvida neste trabalho, consiste de uma camada escondida com 23 neurônios, utilizando-se a função de ativação logística sigmoidal, e um neurônio de saída com uma função de ativação linear. A Tabela 4.1 mostra as variáveis de entrada submetidas ao processo de treinamento da rede neural para estimar o esforço de software dado pela variável Ln(MMACT). A fase de treinamento foi repetida cinco vezes, na tentativa de alcançar a melhor rede para resolver o problema. Além disso, diferentes arquiteturas de redes neurais foram tentadas. Mas, os resultados apresentados na Tabela 4.13 correspondem à rede neural com o melhor desempenho de generalização sobre o conjunto de dados de teste.

Tabela 4.13 - Estimativas de redes neurais após pré-processamento dos dados

Conjunto1		**Conjunto2**		**Conjunto3**		**Conjunto4**		**Conjunto5**		**Conjunto6**	
obs	**est**	**Obs**	**est**	**obs**	**est**	**obs**	**est**	**obs**	**Est**	**obs**	**est**
2040	1801,4	1600	2120,8	243	175,2	240	221,0	33	30,1	43	18,2
8	9,9	1075	610,3	423	571,2	321	192,8	218	333,0	201	218,0
79	77,27	73	20,1	61	93,0	40	27,9	9	6,8	11400	3427,2
6600	6854,5	6400	4169,4	2455	685,7	724	947,2	539	601,6	453	157,8
523	871,7	387	222,0	88	139,8	98	77,4	7,3	5,8	5,9	17,1
1063	1156,1	702	888,8	605	347,0	230	137,9	82	82,7	55	94,2
47	64,5	12	22,8	8	9,0	8	11,2	6	8,1	45	41,4
83	72,8	87	130,8	106	86,6	126	114,9	36	36,7	1272	848,1
156	146,9	176	266,5	122	168,4	41	52,4	14	36,7	20	19,1
18	8,6	958	739,5	237	198,7	130	124,0	70	34,9	57	104,2
50	51,9	38	21,0	15	19,9						

Os valores estimados usando os modelos de redes neurais artificiais obtidos nessa etapa, ou seja, depois do processo de reclassificação das variáveis são mais próximos aos valores reais, quando comparados com aqueles obtidos antes do processo de reclassificação, mostrados anteriormente.

4.6.2. Análise de precisão dos modelos de redes neurais

O MMRE é adotado como indicador de desempenho de predição permitindo a comparação dos resultados obtidos nesta tese com aqueles obtidos por outros pesquisadores. A Tabela 4.14 apresenta os valores de MMRE obtidos através das estimativas realizadas sobre os seis conjuntos de dados de teste, utilizando os modelos de redes neurais artificiais gerados.

Tabela 4.14 - MMRE dos modelos de redes neurais sem pré-processamento dos dados

Conjunto 1	Conjunto 2	Conjunto 3	Conjunto 4	Conjunto 5	Conjunto 6
506	278	353	384	559	278

Os resultados na Tabela 4.14 mostram que nenhum dos modelos executa particularmente bem para estimativas de esforço de desenvolvimento de software, quando se considera o MMRE, utilizando os dados sem pré-processamento.

A Tabela 4.15 apresenta os valores de MMRE resultantes da análise de precisão das estimativas utilizando os modelos de redes neurais artificiais gerados após o pré-processamento dos dados.

Tabela 4.15 - MMRE dos modelos de redes neurais após pré-processamento dos dados

Conjunto1	Conjunto2	Conjunto3	Conjunto4	Conjunto5	Conjunto6
36	47	38	25	40	63

Os resultados apresentados na Tabela 4.15 representam valores médios de 5 diferentes experimentos para cada um dos seis conjuntos de dados considerados. O desenvolvimento de cinco experimentos para cada conjunto de dados tenta minimizar o *bias* que pode ter favorecido o desempenho de alguns conjuntos de dados. Os resultados obtidos mostram que o pré-processamento dos dados teve um impacto positivo na precisão das estimativas e que estes modelos podem ser utilizados com segurança para realizar estimativas.

4.7 Estudo de caso utilizando análise de variância e análise de resíduos

O objetivo deste estudo é apresentar uma abordagem que combina os métodos de análise de regressão, análise de variância e análise de resíduos para gerar modelos de estimativa de esforço de desenvolvimento de software. Essa abordagem (adaptada de Kitchenham, 1998) foi aplicada sobre seis conjuntos de dados de aprendizagem derivados do conjunto de dados COCOMO, mas o conteúdo apresentado nessa seção se refere à análise realizada utilizando o segundo conjunto de dados. Esta seção explica a abordagem seguida na construção e aplicação dos modelos. A escolha de análise de variância é motivada principalmente pela necessidade de se ter um modelo capaz de lidar com conjuntos de dados que apresenta um grande número de variáveis categóricas.

4.7.1. Geração do modelo

O problema de estimativa utilizando um conjunto de dados com um grande número de variáveis categóricas (fatores) pode ser resolvido utilizando um método baseado em análise de variância e análise de resíduos (Kitchenham, 1998). A geração do modelo segue um conjunto de passos conforme descrito abaixo.

Passo 1: análise da normalidade da variável resposta

Uma vez que ANOVA é um modelo estatístico paramétrico e requer uma distribuição normal aproximada é necessário transformar a variável MMACT para melhorar a normalidade, particularmente a estabilidade da variância. Neste caso foi utilizado o conjunto de dados COCOMO com as modificações realizadas no pré-processamento dos dados, incluindo a transformação logarítmica, conforme mostrado na Seção 4.4.

Passo 2: análise de cada fator

Os dados dos tipos absoluto, escalar e intervalo podem ser facilmente incluídos no modelo. Para esses tipos de dados, o impacto da variável independente sobre a variável resposta é avaliado usando regressão linear e calculando-se o erro quadrático médio devido à equação de regressão. Além dos fatores de escala nominal e ordinal, existe um fator com valor absoluto: tamanho da aplicação em número de linhas de código. O impacto dessa variável sobre o esforço é avaliado aplicando-se uma análise de regressão simples (ao invés de ANOVA) sobre o seu valor transformado através de uma operação logarítmica: Ln(ADJKDSI). A linha de regressão que representa este relacionamento é:

$$Ln(MMACT)= 1,245387+1,124779*Ln(ADJKDSI)$$

O efeito de uma regressão linear pode ser representado em uma tabela ANOVA como mostrado na Tabela 4.16. Uma descrição detalhada sobre ANOVA se encontra no Capítulo 3.

Tabela 4.16 - Tabela ANOVA para a regressão

	Soma dos Quadrados	**Grau de liberdade**	**Quadrado médio**	**valor de F**
Regressão de Ln(ADJKDSI)	125,9457	1	125,9457	172,7451
Erro	36,4542	50	0,7291	
Total	162,3999	51		

A análise de variância utilizando um fator (nesse trabalho, as palavras fator e variável são utilizadas indistintamente) foi aplicada para cada variável multiplicadora de esforço sobre os valores da variável Ln(MMACT). A Tabela 4.17 mostra o resultado do primeiro ciclo de análise realizado sobre o segundo conjunto de dados de treinamento. As colunas um, dois, três, quatro, cinco e seis representam, respectivamente, o fator analisado; os níveis de cada fator; o esforço médio de cada nível; o quadrado médio e o grau-de-liberdade (gl) entre grupos; o quadrado médio (termo de erro) e o grau de liberdade dentro de grupos; a estatística F. O fator que apresenta um valor F significativo e o menor termo de erro (observa o valor da quinta coluna da tabela) é definido como o fator mais significativo.

Tabela 4.17 - Análise realizada sobre o segundo conjunto de dados

Variável	**Fator**	**Media**	**MS entre grupos gl**	**MS no grupo gl**	**F teste**
Type	1	4,655259	4,214	3,072	1,3716
	2	4,609130	(5)	(46)	Não
	3	5,819744			
	4	4,166839			
	5	4,177279			
	6	4,207869			
LANG	1	4,996202	3,2769	3,1720	1,0331
	2	5,686715	(6)	(45)	Não
	3	4,661173			
	4	4,175481			
	5	5,799383			
		5,269105			
	7	2,843488			
RELY	1	3,862304	7,9648	2,7775	2,8677
	2	4,071683	(4)	(47)	SIM
	3	4,205353			
	4	5,961153			
	5	4,924316			
DATA	1	4,137777	14,6127	2,2117	6,6070
	2	3,867774	(4)	(47)	SIM
	3	4,431306			
	4	6,033214			
	5	7,000206			

(Continua)

Tabela 4.17 – Continuação

CPLX	1	4,960466	0,4960	3,4765	0,1427
	2	5,041323	(5)	(46)	Não
	3	4,417213			
	4	4,431306			
	5	4,742191			
	6	4,568943			
TIME	1	3,847432	8,9091	2,6971	3,3032
	2	5,717464	(4)	(47)	SIM
	3	5,261823			
	4	5,206835			
	5	6,080538			
STOR	1	3,655818	8,8381	2,7031	3,2696
	2	4,925731	(4)	(47)	SIM
	3	7,805882			
	4	5,346117			
	5	5,069353			
VIRT	1	4,487551	1,1426	3,3119	0,3450
	2	4,609858	(3)	(48)	Não
	3	5,122347			
	4	4,053559			
TURN	1	3,970514	9,2907	2,6646	3,4867
	2	6,897705	(4)	(47)	SIM
	3	4,384828			
	4	5,739705			
	5	4,804021			
PLATF	1	4,750864	2,3355	3,2374	0,7214
	2	4,687060	(3)	(48)	Não
	3	4,693724			
	4	2,817395			
ACAP	1	4,261466	3,7064	3,1399	1,1804
	2	5,097286	(4)	(47)	Não
	3	3,964734			
	4	5,174820			
	5	3,761200			
AEXP	1	4,291339	5,5231	2,9853	1,8501
	2	6,457632	(4)	(47)	Não
	3	4,704904			
	4	5,258210			
	5	3,292906			
PCAP	1	3,482945	5,0950	3,0649	1,6624

(Continua)

Tabela 4.17 - Conclusão

	2	4,587189	(3)	(48)	Não
	3	5,283406			
	4	4,724007			
VEXP	1	4,610972	1,2987	3,3022	0,39328
	2	4,391983	(3)	(48)	Não
	3	5,062189			
	4	4,110874			
LEXP	1	4,212622	2,3962	3,2336	0,74105
	2	5,018607	(3)	(48)	Não
	3	4,631056			
	4	4,110874			
MODP	1	4,698884	1,2791	3,3465	0,3822
	2	4,534184	(4)	(47)	Não
	3	4,559642			
	4	4,256822			
	5	5,328092			
TOOL	1	6,115892	0,5972	3,4045	0,17542
	2	4,609223	(4)	(47)	Não
	3	4,675154			
	4	4,548189			
	5	4,406719			
SCHED	1	4,398547	2,1116	3,2514	0,6494
	2	5,036979	(3)	(48)	Não
	3	5,343950			
	4	4,522050			
RVOL	1	2,811739	7,1707	2,9352	2,4430
	2	4,638885	(3)	(48)	Não
	3	5,126080			
	4	5,186622			
MODE	1	3,672360	22,4544	2,3978	9,3647
	2	4,277047	(2)	(49)	SIM
	3	5,665642			
CONT	1	6,475791	10,4939	2,8860	3,6362
	2	4,767052	(2)	(49)	SIM
	3	3,997877			
Ln(ADJKDSI)			125,9457	0,729084	172,745116 SIM

Verificou-se através do valor F obtido da análise de regressão que a variável que representa o tamanho da aplicação é significativa e positivamente correlacionada com os valores transformados da variável esforço.

Passo 3: identificação do fator mais significante

Tomando-se o conjunto de fatores que tem um efeito significante estatisticamente sobre o esforço, escolhe-se aquele que contém o menor termo de erro (ou seja, o quadrado da média dentro do grupo). Se um fator do tipo absoluto é o fator mais significante (com menor termo de erro), remove-se seu efeito e continua a análise sobre seus resíduos.

A partir dos resultados mostrados na Tabela 4.17, identificou-se que a variável tamanho da aplicação, dada por Ln(ADJKDSI), é a mais significante, uma vez que ela resulta no menor termo de erro. Este resultado é consistente com a literatura e outros trabalhos relacionados (Barcelos Tronto et al., 2006b; Barcelos Tronto et al., 2007), os quais mostram que o tamanho da aplicação é a variável mais relacionada ao esforço.

Passo 4: cálculo dos resíduos

Como a variável mais significante é do tipo absoluto, então os resíduos são calculados subtraindo-se o valor do esforço estimado – resultante da regressão sobre a variável Ln(ADJKDSI) – do esforço real. Se a variável mais significante fosse do tipo nominal, os resíduos seriam calculados subtraindo-se o esforço médio do valor de esforço real do projeto para cada nível de fator considerado. Os resíduos são mostrados na Tabela 4.18.

Tabela 4.18 - Primeiro resíduo

Valor observado	Valor estimado	Resíduo
4,7273878	6,562652	1,058053
4,8828019	6,737458	-1,2444
3,8286414	5,551761	-0,07112
2,7725887	4,363935	-0,86743
1,3862944	2,804661	0,956539
1,9315214	3,417921	-1,33848
3,4011974	5,070981	0,976391
2,8903718	4,496415	1,275026
2,9957323	4,614923	0,769572
3,6109179	5,30687	-0,00357
3,1780538	4,819994	-0,45055
1,3609766	2,776184	1,33469
1,3083328	2,716972	0,971908
0,6418539	1,96733	0,229894
5,768321	7,733471	1,607897
6,8731638	8,976175	-0,18135
5,5294291	7,464771	0,341111
4,6913479	6,522115	0,062676
4,3174881	6,101605	0,18811
4,4998097	6,306677	-0,19078
3,6375862	5,336866	0,922715
2,2407097	3,765689	0,711648
2,5649494	4,130387	0,45458
0,7608058	2,101125	-0,11325
0,6830968	2,013719	-0,23877
3,912023	5,645547	1,323303
3,6888795	5,39456	1,010669
3,0910425	4,722125	0,715954
2,5649494	4,130387	0,276332
2,4849066	4,040357	-0,03302
3,5263605	5,211762	-1,36161
1,8245493	3,297601	-1,21816
0,9162907	2,276011	-0,19657
1,6677068	3,121188	-1,32943
2,9704145	4,586446	-0,77978
3,3322045	4,993379	-0,57454
3,4657359	5,143573	-0,48013
4,0430513	5,792925	-0,95664
3,1354942	4,772124	-1,1886
5,7397929	7,701384	-0,55304
4,5108595	6,319106	-1,26925
2,3025851	3,835285	0,968736
2,1041342	3,612072	0,1015
1,6677068	3,121188	-0,48213
1,4816045	2,911864	0,083868
1,8405496	3,315598	-0,42523

(Continua)

Tabela 4.18 - Conclusão

2,7080502	4,291344	1,176716
3,2188758	4,86591	0,001625
3,0445224	4,669801	-0,42131
1,9021075	3,384837	0,658215
3,3322045	4,993379	-1,08136
2,3025851	3,835285	-1,12724

A Figura 4.3 mostra o gráfico de probabilidade normal para o primeiro resíduo.

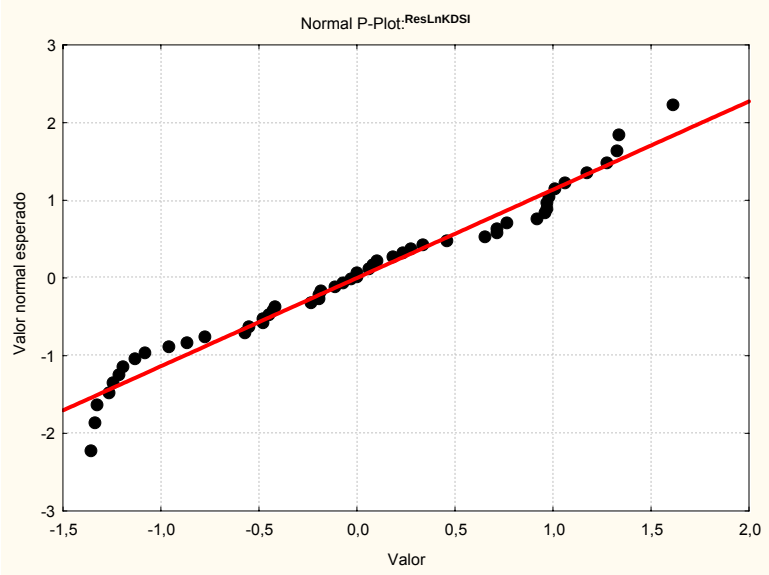

Figura 4. 3 - Representação gráfica da probabilidade normal do primeiro resíduo.

Passo 5: aplicação da ANOVA sobre os resíduos

Este é o segundo ciclo de análise onde se aplica ANOVA sobre os resíduos, para todos os fatores restantes. Nesta etapa deve-se reduzir o grau de liberdade aplicado para o termo de erro porque já foi utilizado k-1 grau de liberdade para remover o impacto do fator mais significante, onde k é o número de níveis do fator mais significante. Nessa análise deve-se subtrair um grau-de-liberdade, pois a variável mais significante encontrada na análise anterior tem domínio em valores absolutos. A Tabela 4.19 mostra o resultado da aplicação de ANOVA sobre os resíduos.

Tabela 4.19 - Análise realizada sobre o primeiro resíduo

Variável	Fator	Media	MS entre grupos gl	MS no grupo gl	F teste
Type	1	-0,251139	1,3778	0,6570	2,0971
	2	0,637921	(5)	(45)	NAO
	3	0,324053			
	4	-0,342303			
	5	-0,225077			
	6	-0,132224			
LANG	1	0,154297	0,9154	0,7037	1,3009
	2	0,017751	(6)	(44)	NAO
	3	-0,254415			
	4	0,451162			
	5	-0,071856			
	6	-0,116598			
	7	-0,660062			
RELY	1	-0,273067	3,4562	0,4919	7,0256
	2	-0,355185	(4)	(46)	SIM
	3	-0,502829			
	4	0,471894			
	5	0,864586			
DATA	1	-0,088436	1,2901	0,6803	1,8963
	2	-0,082569	(4)	(46)	NAO
	3	-0,697775			
	4	0,548356			
	5	0,435710			
CPLX	1	-0,199977	1,1980	0,6770	1,7696
	2	-0,318922	(5)	(45)	NAO
	3	-0,060505			
	4	-0,697775			
	5	-0,054757			
	6	0,440146			
TIME	1	-0,510744	3,8541	0,4573	8,4272
	2	0,151569	(4)	(46)	SIM
	3	0,413923			
	4	0,842158			
	5	0,778533			
STOR	1	-0,589480	3,4421	0,4932	6,9797

(Continua)

Tabela 4.19 – Continuação

	2	-0,014659	(4)	(46)	SIM
	3	0,341111			
	4	0,447610			
	5	0,930062			
VIRT	1	-0,704800	4,5434	0,4856	9,3560
	2	0,222529	(3)	(47)	SIM
	3	0,616884			
	4	0,347293			
TURN	1	-0,173130	0,7453	0,7277	1,0242
	2	-0,138360	(4)	(46)	NAO
	3	-0,050606			
	4	0,369733			
	5	0,968735			
PLATF	1	-0,231321	1,8621	0,6568	2,8352
	2	-0,317678	(3)	(47)	SIM
	3	0,430195			
	4	-0,199131			
ACAP	1	-0,739456	1,9774	0,6205	3,1866
	2	0,280857	(4)	(46)	SIM
	3	-0,196925			
	4	0,209392			
	5	0,956539			
AEXP	1	-0,175003	0,9699	0,7081	1,3696
	2	-0,132295	(4)	(46)	NAO
	3	0,048795			
	4	1,016986			
	5	-0,180934			
PCAP	1	-0,848070	1,9626	0,6504	3,0177
	2	0,104509	(3)	(47)	SIM
	3	0,130602			
	4	0,213044			
VEXP	1	-0,284171	1,3841	0,6873	2,0139
	2	0,038782	(3)	(47)	NAO
	3	0,240464			
	4	1,334690			
LEXP	1	-0,423946	2,3023	0,6287	3,6621
	2	0,154180	(3)	(47)	SIM

(Continua)

Tabela 4.19 - Conclusão

	3	0,377496			
	4	1,334690			
MODP	1	-0,354973	0,8203	0,7212	1,1374
	2	-0,130083	(4)	(46)	NAO
	3	0,072478			
	4	-0,010561			
	5	0,503813			
TOOL	1	-0,190785	1,0436	0,7017	1,4872
	2	-0,364105	(4)	(46)	NAO
	3	-0,044277			
	4	0,404727			
	5	0,276332			
SCHED	1	-0,135545	1,5179	0,6787	2,2363
	2	-0,261010	(3)	(47)	NAO
	3	0,569130			
	4	0,446590			
RVOL	1	-0,381009	4,0795	0,5152	7,9178
	2	-0,362145	(3)	(47)	SIM
	3	0,571968			
	4	0,807510			
MODE	1	-0,431760	6,8037	0,4760	14,2941
	2	-0,515580	(2)	(48)	SIM
	3	0,573548			
CONT	1	0,985365	2,9502	0,6365	4,6347
	2	0,056959	(2)	(48)	SIM
	3	-0,335716			

Passo 6: identificação dos fatores redundantes

Uma parte importante do procedimento de análise é identificar as variáveis cujo impacto sobre o esforço é confundido com a variável mais significante anterior. Nesse caso, algumas variáveis parecem estar relacionadas com a variável Ln(ADJKDSI), porque seu impacto sobre os resíduos não é significante (ver

estatística F) mas seu impacto sobre os dados originais é significante. Estas variáveis são:

- DATA (tamanho da base de dados).
- TURN (tempo de desempenho da máquina).

Não é possível identificar se uma destas variáveis ao invés de Ln(ADJKDSI), está causando o impacto sobre o esforço. No entanto, porque elas são, em efeito, equivalentes a Ln(ADJKDSI), elas devem ser removidas das análises subseqüentes. Como em qualquer análise estatística existe uma chance de que o efeito confuso seja espúrio, mas para simplificar, o procedimento de análise assume que os efeitos confusos são sempre corretamente identificados. Como eles são, em efeito, equivalentes a Ln(ADJKDSI), devem ser removidos da análise subseqüente.

Passo 7: identificação do fator mais significante sobre o resíduo

Somente os fatores que têm um impacto significante baseado na análise de variância sobre os resíduos devem ser considerados, e de novo, se existem vários fatores candidatos, escolhe-se aquele que minimiza o termo de erro. O fator que levou ao menor termo de erro, nesta análise, foi o fator TIME (restrição de tempo). Além disso, o efeito é consistente com a hipótese de que a restrição de tempo é um fator que aumenta o esforço. Assim, assume-se que TIME é o fator mais significante.

Passo 8: remoção do efeito e obtenção do segundo resíduo

O segundo resíduo é obtido removendo-se o efeito do fator TIME dos valores residuais originais (ou seja, para cada nível do fator TIME, subtrai-se o respectivo esforço médio dos valores de esforço). A Tabela 4.20 apresenta o segundo resíduo.

Tabela 4.20 - Segundo resíduo

Valor observado	Valor estimado (médias)	Resíduo (TIME)
1,05805	-0,510744	1,568797
-1,24440	-0,510744	-0,733653
-0,07112	-0,510744	0,439622
-0,86743	-0,510744	-0,356684
0,95654	-0,510744	1,467283
-1,33848	-0,510744	-0,827735
0,97639	0,842158	0,134234
1,27503	0,413923	0,861103
0,76957	0,413923	0,355650
-0,00357	0,413923	-0,417489
-0,45055	0,413923	-0,864469
1,33469	0,842158	0,492532
0,97191	0,842158	0,129750
0,22989	0,842158	-0,612263
1,60790	0,842158	0,765739
-0,18135	0,413923	-0,595274
0,34111	0,151569	0,189542
0,06268	0,842158	-0,779482
0,18811	0,151569	0,036541
-0,19078	0,151569	-0,342354
0,92272	0,413923	0,508792
0,71165	0,842158	-0,130510
0,45458	0,413923	0,040657
-0,11325	-0,510744	0,397494
-0,23877	-0,510744	0,271977
1,32330	0,778533	0,544771
1,01067	0,778533	0,232136
0,71595	0,151569	0,564384
0,27633	0,151569	0,124762
-0,03302	-0,510744	0,477721
-1,36161	-0,510744	-0,850870
-1,21816	-0,510744	-0,707415
-0,19657	-0,510744	0,314175
-1,32943	-0,510744	-0,818684
-0,77978	-0,510744	-0,269039
-0,57454	-0,510744	-0,063795
-0,48013	-0,510744	0,030611
-0,95664	-0,510744	-0,445899
-1,18860	-0,510744	-0,677861
-0,55304	-0,510744	-0,042293
-1,26925	-0,510744	-0,758505
0,96874	-0,510744	1,479480
0,10150	-0,510744	0,612244
-0,48213	0,413923	-0,896053
0,08387	-0,510744	0,594613
-0,42523	-0,510744	0,085518

(Continua)

Tabela 4.20 - Conclusão

1,17672	0,413923	0,762793
0,00163	0,778533	-0,776908
-0,42131	0,151569	-0,572875
0,65821	0,413923	0,244291
-1,08136	-0,510744	-0,570613
-1,12724	-0,510744	-0,616491

Passo 9: repete até que todos os efeitos significantes tenham sido removidos

O processo de análise é repetido até que todos os fatores significantes sejam removidos, ou que não exista graus-de-liberdade disponível para continuar. É preciso ter certeza de que pelo menos quatro graus-de-liberdade estão disponíveis para o termo erro e preferencialmente mais que 10.

Passo 9.1: terceiro ciclo de análise

A análise realizada sobre o segundo resíduo (resultante de ANOVA sobre TIME) é mostrada na Tabela 4.21. Os resultados indicam que os seguintes fatores são confundidos com o fator TIME e devem ser removidos da análise subseqüente:

- RELY (confiabilidade requerida do sistema);
- STOR (restrição de armazenamento);
- VIRT (volatilidade da máquina virtual);
- PLATFORM (plataforma de desenvolvimento);
- LEXP (experiência com linguagem de programação);
- MODE (modo de desenvolvimento).

Tabela 4.21 - Análise realizada sobre o segundo resíduo

Variável	Fator	Media	MS entre grupos gl	MS no grupo gl	F teste
Type	1	0,259605	0,2286	0,4852	0,4710
	2	0,009881	(5)	(42)	Nao
	3	0,033011			
	4	-0,110357			
	5	0,153572			

(Continua)

Tabela 4.21 - Continuação

	6	-0,254967			
LANG	1	0,665041	0,5515	0,4432	1,2444
	2	0,307724	(6)	(40)	Nao
	3	-0,131032			
	4	0,105401			
	5	-0,361671			
	6	-0,268167			
	7	-0,149318			
RELY	1	0,105214	0,1849	0,4833	0,3826
	2	0,056373	(4)	(42)	Nao
	3	-0,196566			
	4	-0,041049			
	5	0,153346			
CPLX	1	0,310767	0,1930	0,4896	0,3942
	2	0,097206	(4)	(42)	Nao
	3	0,042617			
	4	-0,187031			
	5	-0,197320			
	6	0,005568			
STOR	1	-0,115531	0,1832	0,4835	0,3789
	2	-0,028378	(4)	(42)	Nao
	3	0,189542			
	4	0,075721			
	5	0,251858			
VIRT	1	-0,287408	0,9091	0,4258	2,1349
	2	0,066909	(3)	(43)	Nao
	3	0,348028			
	4	-0,066630			
PLATF	1	-0,004095	0,0832	0,4834	0,1721
	2	-0,128262	(3)	(43)	Nao
	3	0,068292			
	4	-0,150720			
ACAP	1	-0,736622	2,4109	0,2713	8,8865
	2	0,027749	(4)	(42)	SIM
	3	-0,038097			
	4	0,720137			
	5	1,467283			
AEXP	1	-0,220367	0,6914	0,4351	1,5892
	2	-0,018296	(4)	(42)	Nao

(Continua)

Tabela 4.21 - Continuação

	3	0,058627			
	4	0,768540			
	5	0,109039			
PCAP	1	-0,785699	2,8954	0,2872	10,0797
	2	-0,060981	(4)	(42)	SIM
	3	0,079439			
	4	0,723788			
VEXP	1	-0,112538	0,1927	0,4758	0,4049
	2	0,035755	(3)	(43)	Nao
	3	0,071579			
	4	0,492532			
LEXP	1	-0,064584	0,1317	0,4801	0,2743
	2	-0,001271	(3)	(43)	Nao
	3	0,095635			
	4	0,492532			
MODP	1	-0,396307	1,6431	0,3444	4,7707
	2	-0,201202	(4)	(42)	SIM
	3	-0,036061			
	4	0,103897			
	5	0,762656			
TOOL	1	-0,342354	0,8364	0,4212	1,9856
	2	-0,145740	(4)	(42)	Nao
	3	-0,158157			
	4	0,410701			
	5	0,124762			
SCHED	1	-0,148909	0,5678	0,4496	1,2627
	2	0,249734	(3)	(43)	Nao
	3	0,012462			
	4	0,249463			
RVOL	1	-0,273307	2,4450	0,3187	7,6724
	2	-0,273937	(3)	(43)	SIM
	3	0,372822			
	4	0,763454			
MODE	1	-0,040618	0,3161	0,4638	0,6815
	2	-0,203208	(2)	(44)	Nao
	3	0,107767			

(Continua)

Tabela 4.21 - Conclusão

CONT	1	0,761138	1,3616	0,4162	3,2712
	2	-0,014128	(2)	(44)	SIM
	3	-0,153329			

Após a aplicação de ANOVA sobre o segundo resíduo identificou-se que o fator mais significante é ACAP (capacidade do analista).

O terceiro resíduo é obtido removendo-se o efeito do fator ACAP (ou seja, para cada nível do fator ACAP, subtrai-se o respectivo esforço médio dos valores de esforço). A Tabela 4.22 apresenta o terceiro resíduo.

Tabela 4.22 - Terceiro resíduo

Valor observado	Valor estimado (médias)	Resíduo (ACAP)
1,568797	0,720137	0,848661
-0,733653	0,027749	-0,761402
0,439622	0,720137	-0,280515
-0,356684	-0,038097	-0,318587
1,467283	1,467283	-0,000000
-0,827735	-0,038097	-0,789638
0,134234	0,027749	0,106485
0,861103	0,027749	0,833354
0,355650	0,027749	0,327900
-0,417489	0,027749	-0,445238
-0,864469	-0,736622	-0,127848
0,492532	0,027749	0,464783
0,129750	0,027749	0,102001
-0,612263	0,027749	-0,640012
0,765739	0,027749	0,737990
-0,595274	-0,736622	0,141348
0,189542	0,027749	0,161793
-0,779482	0,027749	-0,807231
0,036541	0,027749	0,008792
-0,342354	-0,038097	-0,304257
0,508792	0,027749	0,481043
-0,130510	-0,038097	-0,092413
0,040657	0,027749	0,012908
0,397494	0,720137	-0,322643
0,271977	-0,038097	0,310074
0,544771	0,027749	0,517022
0,232136	0,027749	0,204387
0,564384	-0,038097	0,602481
0,124762	-0,038097	0,162859
0,477721	0,720137	-0,242416

(Continua)

Tabela 4.22 - Conclusão

-0,850870	0,027749	-0,878619
-0,707415	-0,736622	0,029207
0,314175	0,027749	0,286426
-0,818684	-0,038097	-0,780587
-0,269039	0,027749	-0,296788
-0,063795	0,027749	-0,091544
0,030611	-0,038097	0,068708
-0,445899	-0,038097	-0,407802
-0,677861	-0,736622	0,058761
-0,042293	0,720137	-0,762430
-0,758505	-0,736622	-0,021883
1,479480	0,720137	0,759343
0,612244	-0,038097	0,650341
-0,896053	-0,736622	-0,159432
0,594613	-0,038097	0,632710
0,085518	0,027749	0,057769
0,762793	-0,038097	0,800890
-0,776908	-0,736622	-0,040286
-0,572875	-0,038097	-0,534778
0,244291	0,027749	0,216542
-0,570613	0,027749	-0,598362
-0,616491	-0,736622	0,120131

PASSO 9.2: O QUARTO CICLO DE ANÁLISE

A análise realizada sobre o terceiro resíduo (resultante de ANOVA sobre ACAP) é mostrada na Tabela 4.23. Os resultados indicam que os seguintes fatores são confundidos com o fator ACAP e devem ser removidos da análise subseqüente:

- PCAP (capacidade do programador),
- MODP (uso de práticas modernas de programação).

Tabela 4.23 - Análise realizada sobre o terceiro resíduo

Variável	**Fator**	**Media**	**MS entre grupos gl**	**MS no grupo gl**	**F teste**
Type	1	-0,216913	0,1147	0,2924	0,3923
	2	0,077678	(5)	(37)	NAO
	3	0,029764			
	4	-0,054850			
	5	0,165203			
	6	-0,005977			

(Continua)

Tabela 4.23 - Continuação

LANG	1	0,134463	0,2666	0,2721	0,9798
	2	-0,199870	(6)	(36)	NAO
	3	-0,090277			
	4	0,243695			
	5	-0,198327			
	6	-0,262993			
	7	-0,144144			
CPLX	1	-0,063176	0,1441	0,2885	0,4996
	2	-0,206007	(5)	(37)	NAO
	3	-0,003910			
	4	-0,181857			
	5	0,051670			
	6	0,125254			
AEXP	1	-0,070940	0,2669	0,2718	0,9821
	2	-0,184684	(4)	(38)	NAO
	3	0,033437			
	4	0,509995			
	5	-0,105609			
PCAP	1	-0,148867	0,2254	0,2748	0,8202
	2	-0,077011	(3)	(39)	NAO
	3	0,153321			
	4	0,083355			
VEXP	1	-0,231377	0,6701	0,2406	2,7851
	2	0,038294	(3)	(39)	SIM
	3	0,231578			
	4	0,464783			
MODP	1	-0,077656	0,3008	0,2682	1,1214
	2	-0,118073	(4)	(38)	NAO
	3	-0,046837			
	4	0,095838			
	5	0,320012			
TOOL	1	-0,304257	0,6800	0,2283	2,9790
	2	-0,142154	(4)	(38)	SIM
	3	-0,137824			
	4	0,368526			
	5	0,162859			
SCHED	1	-0,049987	0,1117	0,2836	0,3938
	2	-0,034602	(3)	(39)	NAO

(Continua)

Tabela 4.23 - Conclusão

	3	0,123082			
	4	0,151021			
RVOL	1	-0,413196	1,3172	0,1908	6,9021
	2	-0,143588	(3)	(39)	SIM
	3	0,216747			
	4	0,610396			
CONT	1	0,576754	0,8531	0,2422	3,5223
	2	0,006937	(2)	(40)	SIM
	3	-0,148356			

Após a aplicação de ANOVA sobre o terceiro resíduo identificou-se que o fator mais significante é RVOL (volatilidade dos requisitos).
O quarto resíduo é obtido removendo-se o efeito do fator RVOL (ou seja, para cada nível do fator RVOL, subtrai-se o respectivo esforço médio dos valores de esforço).

Passo 9.3: o quinto ciclo de análise

A análise realizada sobre o quarto resíduo (resultante de ANOVA sobre RVOL) é mostrada na Tabela 4.24. Os resultados indicam que os seguintes fatores são confundidos com o fator RVOL e devem ser removidos da análise subseqüente:

- CONT (continuidade do pessoal no projeto);
- VEXP (experiência com máquina virtual),
- TOOL (uso de ferramentas de software).

Tabela 4.24 - Análise realizada sobre o quarto resíduo

Variável	Fator	Media	MS entre grupos gl	MS no grupo gl	F teste
Type	1	-0,253493	0,1420	0,1980	0,7173
	2	-0,113697	(5)	(34)	NAO
	3	0,088312			
	4	0,056519			
	5	-0,009586			
	6	0,122490			

(Continua)

Tabela 4.24 - Conclusão

LANG	1	-0,090613	0,2094	0,1875	1,1170
	2	-0,176393	(6)	(33)	NAO
	3	-0,004073			
	4	0,158371			
	5	-0,423403			
	6	0,015399			
	7	0,134248			
CPLX	1	-0,032354	0,1537	0,1963	0,7832
	2	-0,273084	(6)	(33)	NAO
	3	0,061725			
	4	-0,038269			
	5	0,145140			
	6	0,009195			
AEXP	1	-0,021209	0,1518	0,1953	0,7771
	2	-0,221264	(4)	(35)	NAO
	3	-0,011087			
	4	0,293248			
	5	0,217718			
VEXP	1	-0,186121	0,3633	0,1765	2,0586
	2	0,074754			
	3	0,138763	(3)	(36)	NAO
	4	0,248036			
TOOL	1	-0,160669	0,2501	0,1841	1,3586
	2	-0,042805	(4)	(35)	NAO
	3	-0,102968			
	4	0,224234			
	5	-0,053888			
SCHED	1	-0,031059	0,2024	0,1899	1,0659
	2	-0,038480	(3)	(36)	NAO
	3	-0,099217			
	4	0,268687			
CONT	1	0,441762	0,4286	0,1780	2,4083
	2	-0,025286	(2)	(37)	NAO
	3	-0,057835			

Nenhuma das outras variáveis apresentou um nível de significância do F teste significativo. Conseqüentemente, o processo de análise é finalizado e o modelo final obtido é apresentado na próxima seção.

Passo 10: apresentação dos modelos gerados

O modelo derivado a partir do conjunto 2 de aprendizado é dado por:

$$Ln(X) = \alpha + \beta * Ln(ADJKDSI) + \mu_I (TIME / Ln(ADJKDSI)) + \mu_J (ACAP / Ln(ADJKDSI \quad e \quad TIME) + \mu_k (RVOL / Ln(ADJKDSI) e \quad TIME e \quad ACAP) + \varepsilon$$

Para realizar as estimativas utilizando este modelo, os parâmetros formais são substituídos pelos valores apresentados na Tabela 4.25.

Tabela 4.25 - Valores dos parâmetros formais

Parameters	Estimate
α	1,245387
β	1,124779
μ1(TIME/Ln(ADJKDSI))	-0,510744
μ2(TIME/Ln(ADJKDSI))	0,151569
μ3(TIME/Ln(ADJKDSI))	0,413923
μ4(TIME/Ln(ADJKDSI))	0,842158
μ5(TIME/Ln(ADJKDSI))	0,778533
μ1 (ACAP/Ln(ADJKDSI/TIME))	-0,736622
μ2 (ACAP/Ln(ADJKDSI/TIME))	0,027749
μ3 (ACAP/Ln(ADJKDSI/TIME))	-0,038097
μ4 (ACAP/Ln(ADJKDSI/TIME))	0,720137
μ5 (ACAP/Ln(ADJKDSI/TIME))	1,467283
μ1(RVOL/Ln(ADJKDSI/TIME/ACAP))	-0,413196
μ2(RVOL/Ln(ADJKDSI/TIME/ACAP))	-0,143588
μ3(RVOL/Ln(ADJKDSI/TIME/ACAP))	0,216747
μ4(RVOL/Ln(ADJKDSI/TIME/ACAP))	0,610396

O termo de erro da aplicação de ANOVA sobre RVOL, dado por ε é 0,1908. A Tabela 4.26 mostra o resultado das estimativas realizadas através do modelo mostrado anteriormente utilizando o conjunto de teste 2.

Tabela 4.26 - Estimativas obtidas a partir do conjunto dois de teste

alor Observado	Valor Estimado	Resíduo ABSOLUTO	MMRE
1600	1042,8	557,2	81
1075	261,2	813,8	
73	42,9	30,1	
6400	2199,3	4200,7	
387	486,1	99,1	

(Continua)

Tabela 4.26 - Conclusão

702	546,8	55,2	
12	42,7	30,7	
87	187,1	100,1	
176	444,7	268,7	
958	752,6	205,4	

A Tabela 4.27 mostra os modelos finais construídos a partir de cada um dos seis conjuntos de aprendizado, utilizando o mesmo processo de análise apresentado nesta Seção. Verifica-se que os parâmetros dos modelos não são muito estáveis, uma vez que as únicas variáveis que são comuns a todos os modelos são Ln(ADKKDSI), TIME e RVOL. A maior mudança está relacionada aos parâmetros MODP e SCHED, nos conjuntos 4 e 6, os quais não estão presentes em nenhum dos outros modelos.

Tabela 4.27 - Modelos de estimativa de esforço utilizando análise de resíduos

Dados	**Modelo Final**
Conjunto 1	$Ln(X) = \alpha+\beta*Ln(ADJKDSI)+\mu_i(RVOL/Ln(ADJKDSI))+\mu_j(TIME/Ln(ADJKDSI)e\ RVOL)+\mu_k(ACAP/Ln(ADJKDSI)e\ RVOL\ e\ TIME)+\varepsilon$
Conjunto 2	$Ln(X) = \alpha+\beta*Ln(ADJKDSI)+\mu_i(TIME/Ln(ADJKDSI))+ \mu_j(ACAP/Ln(ADJKDSI)\ e\ TIME) + \mu_k(RVOL/ Ln(ADJKDSI)\ e\ TIME\ e\ ACAP) + \varepsilon$
Conjunto 3	$Ln(X) = \alpha+\beta*Ln(ADJKDSI)+\mu_i(RVOL/Ln(ADJKDSI))+\mu_j(TIME/Ln(ADJKDSI)\ e\ RVOL) + \mu_k(PCAP/Ln(ADJKDSI)\ e\ RVOL\ e\ TIME) + \varepsilon$
Conjunto 4	$Ln(X) = \alpha+\beta*Ln(ADJKDSI)+\mu_i(TIME/Ln(ADJKDSI))+\mu_j(ACAP/Ln(ADJKDSI)\ e\ TIME))+\mu_k(RVOL/ Ln(ADJKDSI)\ e\ TIME\ e\ ACAP) + \mu_k(SCHED/ Ln(ADJKDSI)\ e\ TIME\ e\ ACAP\ E\ RVOL)+ \mu_k(MODP/ Ln(ADJKDSI)\ e\ TIME\ e\ ACAP\ E\ RVOL\ E\ SCHED)+ \varepsilon$
Conjunto 5	$Ln(X) = \alpha+\beta*Ln(ADJKDSI)+\mu_i(TIME/Ln(ADJKDSI))+\mu_j(MODP/Ln(ADJKDSI)\ e\ TIME) + \mu_k(RVOL/ Ln(ADJKDSI)\ e\ TIME\ e\ MODP) + \varepsilon$
Conjunto 6	$Ln(X) = \alpha+\beta*Ln(ADJKDSI)+\mu_i(TIME/Ln(ADJKDSI))+\mu_j(RVOL/ Ln(ADJKDSI)\ e\ TIME) + \mu_k(PCAP/ Ln(ADJKDSI)\ e\ TIME\ e\ RVOL) + \mu_k(SCHED/ Ln(ADJKDSI)\ e\ TIME\ e\ RVOL\ e\ PCAP)+ \varepsilon$

Os subconjuntos de teste foram utilizados para avaliar a precisão das estimativas realizadas através dos modelos mostrados na Tabela 4.27. Uma vez que os modelos são baseados nos dados transformados, os valores resultantes precisam ser transformados de volta para a escala dos dados brutos para evitar valores irrealistas. A Tabela 4.28 mostra os valores estimados para cada subconjunto de teste.

Tabela 4.28 - Estimativas usando análise de resíduos

Conjunto 1	Conjunto 2	Conjunto 3	Conjunto 4	Conjunto 5	Conjunto 6
1925,2	1042,8	444,5	866,2	139,8	25,8
15,1	261,2	635,0	196,1	344,8	390,5
93,2	42,9	63,5	45,0	23,7	7576,2
5314,1	2199,3	1360,8	2961,4	492,1	510,7
1091,7	486,1	165,5	84,3	4,6	4,8
626,6	546,8	505,2	282,0	296,8	58,2
126,9	42,7	11,1	5,1	12,1	45,2
82,3	187,1	86,8	145,6	98,1	2062,5
669,1	444,7	218,9	54,2	40,4	15,3
16,1	752,6	58,7	311,2	105,1	60,0
82,3	67,9	24,2			

4.7.2. Análise de precisão dos modelos de resíduos

Um dos objetivos desse trabalho é investigar a precisão das estimativas realizadas pelos modelos gerados utilizando análise de resíduos, ANOVA e regressão. O MMRE é adotado como o indicador da precisão das estimativas produzidas pelos modelos. A Tabela 4.29 apresenta os valores de MMRE obtidos para as estimativas realizadas sobre os seis conjuntos de teste (derivado do modelo COCOMO, conforme descrito na Seção 4.3).

Tabela 4.29 - MMRE dos modelos gerados a partir da análise de resíduos

Conjunto 1	Conjunto 2	Conjunto 3	Conjunto 4	Conjunto 5	Conjunto 6
78	81	51	88	137	30

Esses resultados indicam que as estimativas realizadas, utilizando os métodos estatísticos, têm um forte relacionamento com os valores de esforço reais para todos os projetos de teste.

4.8 Testes utilizando dados de empresas brasileiras

O conjunto de dados COCOMO foi separado em amostras utilizadas para treinar o sistema de aprendizagem e amostras para testar a precisão dos modelos treinados. A fim de avaliar a precisão das estimativas de uma maneira mais realista os modelos gerados também foram testados com um conjunto de dados de 6 projetos desenvolvidos pela Companhia de Desenvolvimento de Software do Estado do Paraná – CELEPAR e com dados de dois projetos desenvolvidos pelo grupo de desenvolvimento de software do DSS/ INPE. Os dados dos projetos destas empresas foram coletados a partir das respostas a um questionário, elaborado com base na descrição dos dados do conjunto de dados COCOMO disponível em Boehm (1981). Esse questionário é apresentado no Apêndice C.

Por questões de privacidade das informações das empresas, os dados referentes aos projetos não são mostrados aqui. As Tabelas 4.30 e 4.31 mostram os valores de MMRE referentes aos testes realizados com os seis modelos regressão múltipla, de redes neurais e de análise de resíduos utilizando os dados das empresas CELEPAR e INPE, respectivamente. As variáveis consideradas nesses conjuntos de dados são as mesmas daquelas do conjunto de dados COCOMO.

Tabela 4. 30 - Precisão das estimativas para os dados de teste da CELEPAR

	Análise de Resíduos	**Redes Neurais**	**Regressão Múltipla**
Modelo 1	73,49	79,72	60,05
Modelo 2	61,94	80,97	66,53
Modelo 3	62,58	83,64	66,30
Modelo 4	79,75	85,25	58,30
Modelo 5	59,73	85,06	57,34
Modelo 6	80,51	85,21	60,41

Tabela 4.31 - Precisão das estimativas para os dados de teste do INPE

	Análise de Resíduos	**Redes Neurais**	**Regressão Múltipla**
Modelo 1	73,92	82,93	88,31
Modelo 2	71,59	81,71	83,16
Modelo 3	74,38	82,71	89,73
Modelo 4	72,50	84,90	89,50
Modelo 5	69,79	84,12	94,89
Modelo 6	61,79	83,35	93,88

Comparando-se os valores de MMRE mostrados acima, com aqueles obtidos dos testes realizados a partir do conjunto de dados COCOMO, percebe-se que houve uma diferença considerável nos resultados para as três técnicas. Essa diferença pode ser atribuída ao uso de linguagens e técnicas modernas de programação, as quais não estavam presentes nos dados COCOMO usados na calibração dos modelos testados. Uma investigação mais detalhada deverá ser conduzida utilizando outros conjuntos de dados.

4.9 Discussão dos resultados

Para cada um dos experimentos, foram coletadas as medidas dos erros das estimativas realizadas através de cada modelo gerado, comparando-se os esforços estimados e os observados. Diferentes medidas de erro têm sido utilizadas por vários pesquisadores, mas nessa tese a principal medida para avaliar a precisão do modelo é a magnitude média do erro relativo – Mean Magnitude of Relative Error – MMRE dada pela equação 2 (capítulo 2).

A Tabela 4.32 apresenta os valores de MMRE obtidos através das estimativas realizadas sobre os seis conjuntos de dados de teste, utilizando os modelos de redes neurais artificiais, os modelos de regressão linear múltipla e os modelos gerados através de um procedimento estatístico que envolve análise de regressão, ANOVA e análise de resíduos. Esses valores correspondem aos modelos gerados utilizando os dados de treinamento pré-processados, uma

vez que para todos os modelos analisados verificou-se um impacto positivo do pré-processamento dos dados nos resultados.

Outros pesquisadores têm utilizado o R^2_ajustado ou o coeficiente de determinação para indicar a percentagem de variação na variável dependente, que é "explicada" em termos das variáveis independentes. Nessa pesquisa realizou-se uma análise de regressão linear para calibrar as predições realizadas utilizando redes neurais, regressão múltipla e o procedimento estatístico usando análise de resíduos. Para tanto se considerou M_{est} como a variável independente e M_{act} como a variável dependente. O valor de R^2 indica a quantidade de variação nos valores reais de esforço, considerada por causa de um relacionamento linear com os valores estimados. Valores de R^2 próximos de 1.0 sugerem um forte relacionamento linear e aqueles próximos de 0.0 sugerem que não há relacionamento. A Tabela 4.32 apresenta, além dos valores de MMRE, os valores de R^2 resultantes da regressão linear de M_{est} e M_{act} para os modelos gerados nessa pesquisa.

Tabela 4.32 - Uma comparação de abordagens de estimativa de esforço

	ANOVA e Resíduos		**Redes Neurais**		**Regressão Múltipla**	
	MMRE	R^2	MMRE	R^2	MMRE	R^2
Conjunto 1	78	0,97	36	0,99	102	0,92
Conjunto 2	81	0,89	47	0,92	51	0,98
Conjunto 3	51	0,86	38	0,94	62	0,83
Conjunto 4	88	0,84	25	0,91	47	0,91
Conjunto 5	137	0,78	40	0,96	48	0,80
Conjunto 6	30	0,97	63	0,98	37	0,98

A Tabela 4.33 mostra os resultados obtidos por Kemerer (1987) com os modelos COCOMO- básico, modelo baseado em Pontos por Função e SLIM utilizando o conjunto de dados COCOMO.

Tabela 4.33 - Resultados de outras abordagens sobre o conjunto de dados COCOMO

Modelos de Estimativa	**MMRE**	**R^2**
Modelo baseado em Pontos por Função	103	0.58
SLIM	772	0.89
COCOMO Básico	610	0.70

Esses valores indicam que as estimativas obtidas nesse trabalho têm um forte relacionamento linear com os valores de esforço de desenvolvimento observados para os projetos usados para teste. Sobre a dimensão R^2, na maioria dos casos, o desempenho do modelo de redes neurais é melhor que o desempenho de SLIM (0,89), modelo baseado em Pontos por Função (0.58) e do modelo COCOMO (0.70) nos experimentos de Kemerer, além dos modelos de regressão múltipla *stepwise* e ANOVA com análise de resíduos.

Em geral, um valor alto de R^2 sugere que ao calibrar um modelo de predição em um novo ambiente, o modelo de predição ajustado pode ser usado com segurança. Considerando-se a dimensão R^2 a abordagem de redes neurais fornece o ajuste mais significativo para os dados.

As estimativas realizadas utilizando análise de regressão e ANOVA com análise de resíduos também têm um forte relacionamento com os valores de esforço reais para os dados de teste. Em termos de MMRE, os resultados de ANOVA com análise de resíduos se mostraram competitivos em relação àqueles obtidos utilizando redes neurais artificiais e apresenta o melhor resultado para o conjunto 6. Porém, outros aspectos devem ser analisados, por exemplo, a complexidade envolvida na aplicação do procedimento de análise de variância e análise de resíduos. Apesar de fornecer resultados com nível de confiança bastante adequado (quando comparado com regressão e redes neurais), este é um método bastante trabalhoso para se aplicar, o que compromete sua utilização (gerentes não querem utilizar um método que tome muito tempo em sua aplicação).

O modelo de redes neurais executa notavelmente melhor que as demais abordagens. Como pode ser visto na Tabela 4.32, redes neurais executa bem e certamente melhor que a abordagem que combina regressão, ANOVA e análise de resíduos, sobre a maioria dos pontos, com exceção do conjunto 6, que tem um valor de MMRE maior que as demais abordagens. Sabe-se que a abordagem de redes neurais executará melhor se os conjuntos de dados de treinamento representarem o maior número possível de cenários de desenvolvimento de software. Com isso, é necessário obter mais dados para que se possa treinar o modelo sobre a maior quantidade possível de características dos dados. Embora o ajuste de redes neurais tenha se mostrado melhor que o ajuste das demais abordagens, estas estimativas estão sujeitas a erros e devem ser interpretadas com cautela.

4.10 Considerações finais

Nesse capítulo foi apresentada uma metodologia para a estimativa de esforço de software, através de três estudos de casos que utilizam redes neurais artificiais, análise de regressão múltipla e uma combinação de análise de variância, análise de regressão e análise de resíduos. Os resultados obtidos foram analisados e discutidos. Finalmente, no próximo capítulo são apresentadas as conclusões finais da tese e sugestões de trabalhos futuros.

5 Conclusões e trabalhos futuros

Nesse trabalho de doutorado foram calibrados e testados modelos de estimativa de esforço, a fim de apresentar abordagens que possam ser utilizadas por uma organização de desenvolvimento de software que deseje aplicar modelos de estimativa em seu processo de gerenciamento de projetos.

Conforme mencionado na revisão bibliográfica dessa tese, existem várias abordagens que são empregadas na estimativa de esforço de software, dentre elas RNA, análise de regressão e análise de variância. Como ainda não se tem um consenso sobre qual é a melhor técnica para uma organização específica, foram realizados vários estudos de casos, a fim de dar suporte a um gerente de projetos na decisão de quais técnicas utilizar e como realizar as estimativas através da técnica selecionada.

Inicialmente foram gerados modelos de estimativa de esforço através das técnicas de regressão simples, regressão múltipla e redes neurais artificiais, utilizando o conjunto de dados de projetos COCOMO, publicado em Boehm (1981). Posteriormente, foram construídos modelos, através de regressão múltipla e redes neurais artificiais, porém utilizando o conjunto de dados COCOMO com algumas transformações realizadas sobre as variáveis. Através desse pré-processamento dos dados foi possível melhorar os resultados das estimativas para todos os modelos. Também, foi gerado um modelo de estimativa de esforço através da combinação de análise de regressão, análise de variância (de um único fator) e análise de resíduos.

Apesar de ter utilizado dados antigos de projetos, ou seja, dados de projetos desenvolvidos utilizando técnicas e linguagens antigas de programação, foi possível construir modelos de estimativa de esforço, cuja eficiência foi avaliada através de testes realizados sobre o conjunto de dados do COCOMO, dos dados de dois projetos desenvolvidos no INPE e de seis projetos desenvolvidos pela CELEPAR, os quais utilizam técnicas modernas de programação.

Além disso, através dos modelos gerados é possível que o gerente identifique um conjunto de atributos que se mostrou influenciar no esforço de desenvolvimento de software. Este conjunto de atributos pode ser utilizado tanto para calibrar um novo modelo para a organização quanto para implementar um programa de medição para apoiar estimativas de projetos futuros baseado nos dados históricos da organização. Um programa de medição contribuir para a melhoria do processo de software.

O trabalho desenvolvido oferece uma contribuição relevante, não só do ponto de vista científico, como também por possibilitar às organizações (sobretudo aquelas que possuem poucas medidas coletadas e recursos humanos com baixo nível de experiência em estimativa de esforço) acesso a um modelo que possa ser útil na realização de estimativas e assim minimizar os problemas causados por estimativas realizadas de forma ad-hoc.

Os resultados obtidos dos experimentos, que utilizam uma combinação de análise de regressão, análise de variância e análise de resíduos, para gerar modelos de estimativa mostram que um modelo linear pode ser utilizado com sucesso, seja como um precursor para uma análise mais detalhada, envolvendo um número restrito de fatores, ou em uma atividade mais focada na coleta de dados. Os modelos gerados incluem os fatores que influenciam a variável resposta. A vantagem da combinação dessas técnicas é evitar contar o mesmo efeito ou enfatizar demais a importância de um fator particular. Este procedimento de análise pode ser bastante útil quando se tem um conjunto de dados com muitos fatores e poucos pontos de dados. Os resultados obtidos, em termos de MMRE, são comparáveis àqueles obtidos utilizando regressão múltipla. Um problema enfrentado por essa abordagem é que a análise ignora qualquer interação entre as variáveis independentes. Além disso, ao treinar um modelo, se o conjunto de dados não tiver informação registrada para todos os níveis de uma variável, o valor da média para aquele nível faltante não será calculado. Assim, se o projeto para o qual a estimativa está sendo realizada tiver o dado registrado para aquele nível e sua média não foi calculada no

modelo, a estimativa é inviabilizada. Isso não acontece, por exemplo com análise de regressão e RNA, que sempre permitirá obter uma resposta, ou seja, realizar uma estimativa.

Considerando-se o conjunto de dados COCOMO, existem dois projetos com valores de esforço muito altos, os quais estão fora de todas as proporções de seus tamanhos, bem como um projeto com valor de esforço muito pequeno para seu tamanho. Nesses casos uma função linear não teria muito sucesso para realizar predições, uma vez que a variável que determina o tamanho de software (ADJKDSI) apresenta a maior significância nas estimativas de esforço. Por outro lado, uma tentativa de resolver estes *outliers* poderia influenciar negativamente na precisão da regressão para realizar estimativas para outras observações.

A abordagem de redes neurais se mostrou bastante adequada para modelar as complexidades envolvidas no domínio de estimativa de esforço de software. A principal vantagem de uma abordagem de RNA é que ela é adaptável e não-paramétrica; com isto, os modelos de estimativa podem ser ajustados aos dados para um domínio particular. Uma vez que RNA não é limitada a uma função linear, ela pode lidar eficientemente com observações que estão fora da linha de ajuste. Os modelos baseados em redes neurais são capazes de capturar os parâmetros que influenciam no esforço de desenvolvimento. Sobre um conjunto de dados mais homogêneo, sem outliers, o método de regressão poderia apresentar um desempenho melhor. Mas, a abordagem de redes neurais também apresenta um melhor desempenho sobre tal conjunto de dados, dependendo, é claro, da fase de treinamento. As estimativas de RNA apresentaram melhores índices de precisão que aqueles obtidos com regressão múltipla (Barcelos Tronto et al., 2006b; Barcelos Tronto et al., 2007b) e regressão simples (Barcelos Tronto et al., 2007b). Os resultados também se mostraram competitivos quando comparados com SLIM, COCOMO e modelo baseado em Pontos por Função.

O grau de precisão das estimativas realizadas sobre o conjunto de dados COCOMO depois do pré-processamento (usando análise de variância para redefinir as variáveis categóricas e uma transformação logarítmica das variáveis numéricas) é melhor que aquele obtido antes de se realizar essas operações. Com isso, conclui-se que o pré-processamento dos dados tem um impacto bastante significativo na precisão dos modelos de estimativa de esforço de software para redes neurais e para a abordagem de análise de regressão, quando comparado a métodos de estimativa de esforço existentes na literatura.

Os resultados de vários experimentos revelaram que pode existir uma considerável variação no desempenho desses sistemas quando a natureza dos dados históricos muda. Por exemplo, para um conjunto de dados, a regressão teve um desempenho melhor que o modelo de redes neurais, o que conduz à continuidade desse estudo. Assim, novos experimentos poderão ser conduzidos a fim de combinar técnicas de regressão múltipla, redes neurais e a abordagem combinada de análise de variância e análise de resíduos para calibrar e testar modelos de estimativa sobre outros conjuntos de dados, tais como, o conjunto de dados ISBSG (International Software Benchmarking Standard Group). Ele contém informações sobre projetos de software desenvolvidos com técnicas modernas de desenvolvimento de software. O objetivo é melhorar o desempenho dos modelos obtidos nesse trabalho e obter um modelo que possa ser utilizado com segurança em desenvolvimento de software.

Outro aspecto importante a se considerar é o fato de que as estimativas devem contemplar as várias fases do ciclo de desenvolvimento de software e o nível de precisão das estimativas deve ser melhorado à medida que se prossegue no ciclo. Dentro dessa linha de pesquisa, se insere uma vertente que a autora pretende explorar em trabalhos futuros: estabelecer uma metodologia que permita realizar estimativas em cada fase do processo de desenvolvimento, constituída de um procedimento para realizar estimativas em fases anteriores à

fase de projeto (em que não se sabe o tamanho da aplicação) e outro procedimento para realizar estimativas em fases posteriores (em que já é possível saber o tamanho da aplicação).

O primeiro procedimento deverá ser aplicado quando ainda não se tem uma especificação de requisitos funcionais muito segura para se estimar o tamanho da aplicação (que é a variável que tem maior influência sobre o esforço). Nesse caso, a idéia é realmente definir um conjunto de medidas que influenciam o esforço de um modo geral, e oferecer este *framework* a uma empresa, juntamente com um procedimento simples que permita realizar estimativas baseado em casos. De posse desse *framework* a empresa poderá selecionar as variáveis sobre as quais ela tem registrado medidas de projetos e utilizar o procedimento para estimar usando aquelas medidas. O conjunto de variáveis desse *framework* será definido aplicando-se a abordagem descrita em Barcelos Tronto et al. (2006a), porém utilizando a base de dados fornecida pelo ISBSG. O objetivo é evitar que informações irrelevantes sejam consideradas nas estimativas pelas empresas.

O segundo procedimento deverá ser aplicado quando se encontra na fase de projeto ou em fases posteriores ao projeto, com condições para realizar estimativas seguras do tamanho da aplicação. Nesse caso, a idéia é aplicar modelos algorítmicos, seguindo a abordagem de modelo composição de especialistas. Nessa abordagem devem ser consideradas as técnicas experimentadas nos estudos de casos realizados como parte desse trabalho: Redes Neurais Artificias, Análise de Regressão, ANOVA e análise de resíduos, raciocínio baseado em casos.

É importante notar que uma única técnica pode não ser capaz de fornecer resposta para todos os aspectos relacionados ao esforço de software. É preciso ter um conjunto compreensivo de técnicas, em que cada técnica tem suas próprias características e capacidades. Nesse sentido propõe-se que seja projetado e implementado um ambiente global integrado para dar suporte aos

procedimentos mencionados e apoiar gerentes de projetos no processo de estimativa de esforço de software.

Resultados preliminares de testes utilizando dados de empresas brasileiras (o grupo de desenvolvimento de software do DSS/ INPE e a Companhia de Desenvolvimento de Software do Estado do Paraná - CELEPAR) mostram a necessidade de ajustar os modelos com os dados históricos da organização. Assim, recomenda-se fortemente a utilização da abordagem de redes neurais, que tem um procedimento simples de calibração e fornece boas estimativas. Por outro lado, uma empresa que ainda não possui uma base de dados de histórico de projetos poderá utilizar os modelos gerados e apresentados nessa tese para realizar estimativas, porém, combinando-se vários modelos (por exemplo, através do cálculo da média das estimativas obtidas dos vários modelos) e ajustando as estimativas através da opinião de especialista.

Nesse trabalho foram identificadas duas grandes contribuições. Primeiro, foram identificados fatores-chave de projeto que determinam o esforço de desenvolvimento de software, através das variáveis que aparecem em cada um dos modelos obtidos. Segundo, são fornecidas três abordagens para estimativa de esforço de software, que podem ser utilizadas de forma simples para obter resultados consideravelmente precisos, quando comparados com métodos da literatura. Essas abordagens podem ser combinadas, realizando-se estimativas e calculando-se a média das estimativas ou escolhendo-se, através da técnica do bom senso, aquela que julgar mais próxima da realidade.

REFERÊNCIAS BIBLIOGRÁFICAS

ABDEL-HAMID, T. Adapting, correcting, and perfecting software estimates: a maintenance metaphor. **IEEE Computer**, v. 26, n. 3, p. 20-29, Mar. 1993.

ADDISON, T.; S. VALLABH. Controlling software project risks - an empirical study of methods used by experienced project managers. In: ANNUAL RESEARCH CONFERENCE OF THE SOUTH AFRICAN INSTITUTE OF COMPUTER SCIENTISTS AND INFORMATION TECHNOLOGISTS ON ENABLEMENT THROUGH TECHNOLOGY**,** 2002, Port Elizabeth, South Africa. **Proceedings…** Port Elizabeth: ACM, 2002. p. 128-140. (ISBN:1-58113-596-3)

AGARVAL, R. Estimating software projects. **Software Engineering Notes**, v.26, p. 60-57, 2001.

AGRAWAL, M.; CHARI. K. Software effort, quality, and cycle time: a study of CMM level 5 projects. **IEEE Transaction on Software Engineering**, v. 33, n.3, p. 145-156, Mar. 2007.

ALBRECHT, A. Measuring application development productivity. In: JOINT SHARE/GUIDE/IBM APPLICATION DEVELOPMENT SYMPOSIUM, 1979, Monterey. **Proceedings…** Monterey: IBM, 1979. p. 83-92.

ALBRECHT, A.J.; GAFFNEY, J.R. Software function, source lines of code and development effort prediction: a software science validation. **IEEE Transactions on Software Engineering**, v. 9, n. 6, p. 639-648,1983.

AMESCUA, J.G.; VELASCO, M.; MARTINEZ, P.; RUIZ, B.; GARCIA, L.; CALVO-MANZANO, A.; FELIU, T.S. A Software project management framework. **Information System Management**, Spring, v.21, n.2, p.78-85, 2004.

ANDA, B.; ANGELVIK, E.; RIBU, K. Improving estimation practices by applying use case models. In: INTERNATIONAL CONFERENCE ON PRODUCT FOCUSED SOFTWARE PROCESS IMPROVEMENT**,** 4, 2002. Rovaniemi, Finland. **Proceedings...** Rovaniemi, Finland: Springer-Verlag, 2002. p. 383-397. Lecture Notes in Computer Science.

ANGELIS, L.; STAMELOS, I. A simulation tool for efficient analogy based cost estimation. **Empirical Software Engineering**, n.5, p.35-68, 2000.

ANGELIS, L.; STAMELOS, I.; MORISIO, M. Building a Software Cost Estimation Model Based on Categorical Data. In**:** IEEE INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, 7., 2001, London:. **Proceedings…** London: IEEE, 2001. p. 4-15.

ATKINSON, A. Plots. **Transformations and regression:** an Introduction to graphical methods of diagnostic regression analysis. Oxford: Clarendon Press, 1983.

BARCELOS TRONTO, I.F.; SIMÕES DA SILVA, J. D.; SANT'ANNA, N. Melhorando as Estimativas de Esforço de Software Através de um Modelo Baseado em Analogia. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 26. SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE – SEMISH, 33., 2002, Campos grande. **Anais...** Campo Grande: [s.n], 2006[a]. p. 217-230.

BARCELOS TRONTO, I. F.; SIMÕES DA SILVA, J.D.; SANT'ANNA, N. Uma investigação de modelos de estimativas de esforço em gerenciamento de projeto de software. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE – SBES, 2006, Florianópolis, Brasil. **Anais...** Florianópolis [s.n], 2006b. p. 224-240.

BARCELOS TRONTO, I. F.; SIMÕES DA SILVA, J.D.; SANT'ANNA, N. A Comparative assessment of predictive methods in software development. In: IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, 2007, Orlando, EUA. **Proceedings…** Orlando: IEEE, v.1, p. 771-776, 2007a.

BARCELOS TRONTO, I. F.; SIMÕES DA SILVA, J.D.; SANT'ANNA, N. An investigation of artificial neural networks based prediction systems in software project management. **Journal of System and Software**, Elsevier, June 2007b (doi:10.1016/j.jss.2007.05.011)

BASILI, V.R.; ROMBACH, H.D. The TAME project: towards improvement-oriented software environments. **IEEE Transactions on Software Engineering**, v.14, n.6, p.758-771, 1988.

BASILI, V.R.; CALDIERA, G.; ROMBACH, H.D. Experience factory. In J.J.Marciniak (ed.), **Encyclopedia of Software Engineering**, v.1, New York: John Wiley & Sons, 1994a.

BASILI, V.R.; CALDIERA, G.; ROMBACH, H.D. Goal question metric paradigm. In J.J.Marciniak (ed.). **Encyclopedia of Software Engineering**, v.1, New York: John Wiley & Sons, 1994b.

BEALE, R.; JACKSON, T. **Neural computing.** Bristol – UK: Institute of Physics Publishing, 1992. 240 p.

BERGERON, F.; ST-ARNAUD, J.-Y. Estimation of Information Systems Development Efforts: A Pilot Study. **Information & Management**, v. 22, p. 239-254, 1992.

BISIO, R.; MALABOCCHIA, F. Cost estimation of software projects through case base reasoning.In: CASE-BASED REASONING RESEARCH & DEVELOPMENT: NTERNATIONAL CONFERENCE ON CASE-BASED REASONING, 1., 1995, Berlin. **Proceedings…** Berlin: Springer-Verlag, 1995, p.11-22.

BOEHM, B. **Software Engineering Economics**. Upper Saddle River, NJ: Prentice Hall, 1981.

BOEHM, W.; HOROWITZ, E.; MADACHY, R.; REIFER, D.; CLARK, B.K.; STEECE, B.; BROWN, A.D.; ABTS, C. **Software Cost Estimation with COCOMOII.** Upper Saddle River, NJ: Prentice-Hall, 2000.

BRIAND, L.; BASILI V.; THOMAS, W. A Pattern recognition rpproach for software engineering data analysis. **IEEE Transactions on Software Engineering**, v.18, n. 11, Nov. 1992.

BRIAND, L.C.; LANGLEY, T.; WIECZOREK, I. A replicated assessment and comparison of common software cost modeling techniques. In: INTERNATIONAL CONFERENCE ON ON SOFTWARE ENGINEERING, 22., 2000, Limerick**. Proceedings…** Limerick, Ireland: ACM, 2000. p. 377-386.

BRIAND, L.C. AND WIECZOREK, I. Software resource estimation, **Encyclopedia of Software engineering**, n.2, p. 1160-1196, 2002.

BRIAND, L.C.; MORASCA, S.; BASILI, V.R. An operational process for goal-driven definition of measures. **IEEE Transactions on Software Engineering**, v. 28, n.12, p. 1106-1125, Dec. 2002.

CLARK, L.J. **The Essentials of Business statistics II**. New Jersey: Research & Education Association, 2001.

CMMI Product Team. **CMMI® for systems engineering/software engineering/Integrated product and process development/supplier sourcing**, Version 1.1, Continuous Representation. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. (CMMI-SE/SW/IPPD/SS, V1.1, Continuous) (CMU/SEI-2002-TR-011, ADA339818) Disponível em: http://www.sei.cmu.edu/publications/documents/02.reports/02tr011.html. Acesso em: 28 nov. 2007.

DRAPER, N.R., SMITH, H. **Applied regression analysis**. New York: Jonh Willey & Sons, 1966. 407p

FENTON, N. E.;PFLEEGER, S. L. **Software metrics:** a rigorous and practical approach. Cambrige: Cambridge University Press, 1997.

FINNIE, G.R.; WITTING, G.E.; DESHARNAIS, J-M. A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models, **Journal of Systems and Software**, v. 39, p. 281-289, 1997.

FREEMAN, J.; SKAPURA, D.M. **Neural network –** algorithms and programming techniques. Reading: Addison Wesley,1991.

GENERAL ACCOUNTING OFFICE. **Summary of federal agencies' information resources management problems.** Washington, DC: GAO/IMTEC, 1992. (GAO/IMTEC-92-13FS).

GENERAL ACCOUNTING OFFICE. **Improving mission performance through strategic information management and technology**. Washington DC: GAO/AIMD, 1994. (GAO/AIMD-94-115)

GIBBS, W. W. Software's chronic crisis. **Scientific American**, v. 271, n. 3, p. 86-95, 1994.

GOODMAN, P. A. Application of cost-estimation techniques: Industrial perspective. **Information and Software Technology**, v. 34, n.6, p. 379-382, 1992.

GRAY, A.R.; MACDONELL, S.G. A comparison of model building techniques to develop predictive equations for software metrics. **Information and Software Technology**, v. 39, p. 425-437, 1997.

GRAY, A.R.; MACDONELL, S.G.; SHEPPERD, M. Factors systematically associated with errors in subjective estimates of software development effort: the stability of expert judgment. In: IEEE INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, 6., 1999, Boca Raton. **Proceedings…** Boca Raton: IEEE, 1999.

GRAY, A.R.; MACDONELL, S.G. Software metrics data analysis – exploring the relative performance of some commonly used modeling techniques. **Empirical Software Engineering**, v.4, p.297-316, 1999.

HASTING, T.E.; SAJEEV, A.S.M. A vector based approach to software size measurement and effort estimation. **IEEE Transactions on Soft. Engineering**, v. 27, n.4, 2001.

HAYKIN, S. **Redes neurais:** princípios e práticas. Porto Alegre: Bookman, 2001.

HECHT-NIELSEN, R. **Neurocomputers.** Reading, MA: Addison Wesley, 1990.

HEEMSTRA, F. J.; KUSTERS, R.J. Function point analysis: Evaluation of a software cost estimation model. **European Journal of Information Systems**, v.1, n. 4, p. 223-237, 1991.

HEEMSTRA, F. J. Software cost estimation. **Information and Software Technology, v.** 34, n.10, p. 627-639, 1992.

HOAGLIN, D.; MOSTELLER, F.; TUKEY, J. **Understanding robust and exploratory data analysis**. New York: John Wiley & Sons, 1983.

HOPFIELD, J.J. Neural networks and physical systems with emergent collective computational abilities. **Proceedings of the National Academy of Science**, USA, v. 79, p. 2554-2558, 1962.

INGRAM, T. Client/server and imaging: on time, on budget, as promised. **PM Network**, v. 9, p. 13-20, 1995.

INTERNATIONAL STANDARD ORGANIZATION (ISO). **SPICE software process assessment** – Parte1: Concepts and Introductory Guide, v.1, s.1. ISO/IEC, 1995.

THE INTERNATIONAL ORGANIZATION FOR STANDARDIZATION AND THE INTERNATIONAL ELECTROTECHNICAL COMMISSION. **ISO/IEC TR 15504 -** information technology - software process assessment, document set with nine parts: ISO/IEC TR 15504-1 to ISO/IEC TR 15504-9, 1998.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION AND THE INTERNATIONAL ELECTROTECHNICAL COMMISSION. **ISO/IEC 12207-** tecnologia da informação – processos de ciclo de vida de software. Rio de Janeiro, 1997.

JEFFERY, R.;RUHE, M.; WIECZOREK, I. Using public domain metrics to estimate software development effort. In: IEEE INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, 7., 2001, London. **Proceedings…** London: IEEE, 2001, p.16-27.

JENKINS, A.M.; NAUMANN, J.D.; WETHERBE, J.C. Empirical Investigation of Systems Development Practices and Results. **Information & Management**, v.7, p. 73-82, 1984.

JONES, CAPERS. **Programmer productivity**. New York: McGraw-Hill Publishing Co,1986.

JØRGENSEN, M. Estimation of software development work effort: evidence on expert judgment and formal models. **International Journal of Forecasting**, v. 33, n.1, p. 33-53 Jan. 2007.

JORGENSEN,M.; SHEPPERD, M. A Systematic Review of Software Development Cost Estimation Studies. **IEEE Transactions on Software Engineering,** v.33, n.1, Jan 2007.

KARNER, G. **Metrics for objectory**. Diploma thesis, University of Linkoping, Dec. 1993.

KARUNANITTHI, N.; WHITLEY, D.; MALAIYA, Y.K. Using neural networks in reliability prediction**. IEEE Software**, v. 9, n.4, p.53-59, 1992.

KEMERER, C.F. An empirical validation of software cost estimation models, **Comunication of ACM**, v.30, p.416-429, 1987.

KHOSHGOFTAAR T.; PANDYA A.; LANNING, D. Application of Neural Networks for predicting program faults. **Annals of Software Engineering**, v. 1, p.141-154,1995.

KITCHENHAM, B. A.; TAYLOR, N. R. Software project development cost estimation. **The Journal of Systems and Software**, v. 5, p. 267-278, 1985.

KITCHENHAM, B.; LINKMAN, L.; LAW, D. DESMET: A methodology for evaluating software engineering methods and tools. **Computing and Control Engineering Journal**, v. 8, n. 3, p. 120-126, 1997.

KITCHENHAM, B. A procedure for analyzing unbalanced datasets. **IEEE Transactions on Software Engineering**, v. 24, n. 4, p. 278-301, 1998.

KOHONEN, T. **Content-addressable memories.** Berlin: Springer-Verlag, 1987.

KOSKO, B. **Neural networks and fuzzy systems.** New Jersey: Prentice-Hall, 1992.

KUMAR, S.; KRISHNA, B.A.; SATSANGI, P.S. Fuzzy systems and neural networks in software engineering project management. **Journal of Applied Intelligence**, v.4: p. 31-52, 1994.

LAI, R.; HUANG, S. A model for estimating the size of a formal communication protocol specification and its implementation. **IEEE Transaction on Software Engineering**, v.29, n. 1, p. 46-62, 2003.

LIPPMANN, R.P. An Introduction to computing with neural nets**. IEEE-ASSP Magazine**, v. 4, n.2, Apr. 1987.

MACHADO, C.A.F. NBR ISO /IEC 12207. Processos de ciclo de vida de software. In: Rocha, A.R.C.; Maldonado, J.C.; Weber, K.C. **Qualidade de software:** teoria e prática. São Paulo: Prentice Hall, 2001.

MATSON, J.E.; BARRETT, B.E.; MELLICHAMP, J.M. Software Development Cost Estimation Using Function Points. **IEEE Transactions on Software Engineering**, v. 20, p. 275-287, 1994.

MAXWELL , K. D. **Applied statistics for software managers**. 1. ed. New Jersey: Prentice Hall, 2002,

McCULLOCK, W. S.; PITTS, W. H. A logical calculus of ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 115-133, 1943.

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA (MCT). **Qualidade e produtividade no setor de software**. Disponível em: http://www.mct.gov.br/Temas/info/Dsi/Quali2001/2001Tab40.htm, Tabela 40 – Práticas de Engenharia de Software no Desenvolvimento e Manutenção de Software.

MEDEIROS, J.S. **Banco de dados geográficos e redes neurais artificiais:** tecnologias de apoio à gestão do território. Tese de Doutorado, Faculdade de Filosofia, Letras e Ciências Humanas, USP, Brasil, 1999.

MOLØKKEN-ØSTVOLD, K.; JØRGENSEN, M.; TANILKAN, S. S.; GALLIS, H.; LIEN, A. C.; HOVE1, S. E. A Survey on Software Estimation in the Norwegian Industry. In: INTERNATIONAL SYMPOSIUM ON SOFTWARE METRICS, 10., Chicago**,** 2004. **Proceedings…**Chicago: IEEE, 2004.

MOORES, T.T.; EDWARDS,J.S. Could LargeUK Corporations and Computing Companies Use Software Cost Estimating Tools? – A Survey. **European Journal of Information Systems**, v.1, n. 5, p. 311-319, 1992.

MINSKY, M.L.; PAPERT, S.A. **Perceptrons:** an introduction to computational geometry**.** Massachussets: M.I.T. Press, 1969.

MUKHOPADHYAY, T.; SUNDER, K. Software effort models for early estimation of process control applications. **IEEE Transactions on Software Engineering,** v. 18, p. 915-924, 1992.

MYRTVEIT, I.; STENSRUD, E.; SHEPPERD, M. Reliability and validity in comparative studies of software prediction models, **IEEE Transaction on Software Engineering**, v.31, n. 5, p. 46-62, 2005.

NICK, M.; ALTHOFF, K.D.; TAUZ, C. Facilitating the paractical evaluation of organizational memories using the goal-question-metric technique. In: WORKSHOP ON KNOWLEDGE ACQUISITION, MODELING AND MANAGEMENT, 12., 1999, Banff, Alberta, Canada, **Proceedings…** Bannff, Alberta: [s.n], 1999.

PHAN, D. **Information systems project management:** an integrated resource planning perspective model**.** P&h Thesis - Department of Management and information systems**,** Arizona: Tucson, 1990.

PROJECT MANAGEMENT INSTITUTE (PMI). **A guide to the project management body of knowledge:** PMBOK guide 2000 edition. Pensylvania: Project Management Institute, 2000.

PUTNAM, L. H. A general empirical solution to the macro software sizing and estimation problem. **IEEE Transactions on Software Engineering,** n.4, p.345-361, 1978.

RICH, E.; KNIGHT, K. **Inteligencia artificial.** São Paulo: Makron Books, 1993.

ROSENBLATT, F. **Principles of neurodynamics:** perceptrons and theory of brain mechanisms. Washington, DC: Spartan Books, 1962.

RUMELHART, D.E.; HILTON, G.E.; WILLIANS, R.J. **Learning internal representations by error propagation, parallel distributed processing:** explorations in the microstructure of cognition 1**.** Cambridge, MA: M.I.T. Press, 1986. v. 1, p. 318-362.

SAMSON, B.; ELLISON, D.; DUGARD,P. Software Cost Estimation Using an Albus Perceptron (CMAC). **Information and Software Technology**, v. 39, n. 1, p. 55-60, 1997.

SANT'ANNA, N. **Um ambiente integrado para o apoio ao desenvolvimento e gestao de projetos de software para sistemas de controle de satélite**. 2000-09. 225 p. (INPE-8306-TDI/765). Tese (Doutorado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, Sao Jose dos Campos. 2000. Disponível em: <http://urlib.net/dpi.inpe.br/lise/2002/03.28.20.03>. Acesso em: 27 nov. 2007.

SCHNEIDER, G.; WINTERS, J. **Applying use cases –** a practical guide. Reading, MA: Addison-Wesley, 1998.

SELBY, R.W.; PORTER, A.A. Learning from examples: generation and evaluation of decision trees for software resource analysis. **IEEE Transactions on Software Engineering,** v. 14, p. 1743-1757, 1998.

SENTAS, P.; ANGELIS, L.; STAMELOS, I.; BLERIS, G. Software productivity and effort prediction with ordinal regression. **Journal Information and Software Technology**, n. 47, p.17-29, 2005.

SHAPIRO, S.; WILK, M. An Analysis of variance test for non-normality (complete samples). **Biometrics**, v. 52., p.591-611, 1965.

SHEPEERD, M.; SCHOFIELD, M.; KITCHENHAM, B. Effort Estimation Using Analogy. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE18), 18., 1996, Berlin. **Proceedings…**, Berlin: IEEE, 1996.

SHEPPERD, M.; SCHOFIELD, M. Estimating software project effort using analogies. **IEEE Transactions on Software Engineering**, v. 23, n. 12, 1997.

SRINIVAZAN, K.; FISHER, D. Machine learning approaches to estimating software development effort. **IEEE Transactions on Software Engineering**, v.21, n.2, p.126-137, 1995.

SUBRAMANIAN, G. H.; PENDHARKAR, P.C.; WALLACE, M. An empirical study of the effect of complexity, platform, and program type on software development effort of business applications. **Empir. Software Engineering**, v.11, p. 541-553, 2006.

SUBRAMANIAN, G. H.; BRESLAWSKI, S. An empirical analysis of software effort estimate alterations. **The Journal of Systems and Software**, v. 31, p. 135-141,1995.

VAZQUEZ, C. E.; SIMÕES, G.S.; ALBERT, R.M. **Análise de pontos de função:** medição, estimativas e gerenciamento de projetos de software. 5. ed. São Paulo: Érica, 2006.

VICINANZA, S.; PRIETULA, M.J.; MUKHOPADHYAY, T. Case-based reasoning in software effort estimation. In: INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS, COPENHAGEN, DENMARK INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS, 11., Copenhagen, Denmark. **Proceedings…** Copenhagen: [s.n], 1990. p.149-158.

ZHONG, S.; KHOSHGOFTAAR, T.M.; SELIYA, N. Analyzing software measurement data with clustering techniques. **IEEE Intelligent Systems**, v. 19, n. 2, p.20-27, 2004.

WITTING, G.; FINNIE, G. Estimating software development effort with connectionist models. **Information and Software Technology**, v. 39, n. 7, p. 369-476, 1997.

APÊNDICE A - CONJUNTO DE DADOS COCOMO

TYPE	LANG	RELY	DATA	CPLX	TIME	STOR	VIRT	TURN	PLATFORM	ACAP	AEXP
BUS	COB	0.88	1.16	0.70	1.0	1.06	1.15	1.07	MAX	1.19	1.13
BUS	COB	0.88	1.16	0.85	1.0	1.06	1.0	1.07	MAX	1.0	0.91
BUS	PLI	1.0	1.16	0.85	1.0	1.0	0.87	0.94	MID	0.86	0.82
BUS	COB	0.75	1.16	0.70	1.0	1.0	0.87	1.0	MID	1.19	0.91
BUS	FTN	0.88	0.94	1.0	1.0	1.0	0.87	1.0	MAX	1.0	1.0
BUS	PLI	0.75	1.0	0.85	1.0	1.21	1.0	1.0	MID	1.46	1.0
BUS	COB	0.75	1.0	1.0	1.0	1.0	0.87	0.87	MAX	1.0	1.0
CTL	MOL	1.15	0.94	1.30	1.66	1.56	1.30	1.0	MIN	0.71	0.91
CTL	FTN	1.15	0.94	1.30	1.30	1.21	1.15	1.0	MIN	0.86	1.0
CTL	MOL	1.40	0.94	1.30	1.11	1.56	1.0	1.07	MIN	0.86	0.82
CTL	MOL	1.40	0.94	1.30	1.11	1.56	1.0	1.07	MIN	0.86	0.82
CTL	JOV	1.15	0.94	1.30	1.11	1.06	1.0	1.0	MIN	0.86	0.82
CTL	FTN	1.15	0.94	1.30	1.11	1.06	1.15	1.0	MID	0.71	1.0
CTL	MOL	1.15	0.94	1.65	1.30	1.56	1.15	1.0	MIC	0.86	1.0
CTL	MOL	1.40	0.94	1.30	1.30	1.06	1.15	0.87	MIN	0.86	1.13
CTL	MOL	1.40	1.0	1.30	1.30	1.56	1.0	0.87	MIN	0.86	1.0
CTL	MOL	1.40	1.0	1.30	1.30	1.56	1.0	0.87	MIN	0.86	0.82
HMI	FTN	1.15	1.16	1.15	1.30	1.21	1.0	1.07	MAX	0.86	1.0
HMI	FTN	1.15	1.08	1.0	1.11	1.21	0.87	0.94	MAX	0.71	0.91
HMI	JOV	1.40	1.08	1.30	1.11	1.21	1.15	1.07	MAX	0.71	0.82
HMI	PLI	1.0	1.16	1.15	1.06	1.14	0.87	0.87	MAX	0.86	1.0
HMI	JOV	1.15	1.0	1.0	1.27	1.06	1.0	1.0	MAX	0.86	0.82
HMI	HOL	1.15	1.0	1.0	1.08	1.06	1.0	1.0	MAX	0.86	0.82
HMI	FTN	0.88	1.0	0.85	1.06	1.06	1.0	0.87	MIN	1.0	1.29
HMI	JOV	1.15	1.16	1.30	1.15	1.06	1.0	0.87	MAX	0.86	1.0
HMI	HOL	0.94	1.0	0.85	1.07	1.06	1.15	1.07	MAX	0.86	1.0
HMI	FTN	1.15	0.94	1.15	1.35	1.21	1.0	0.87	MIN	1.0	1.0
HMI	MOL	1.15	1.08	1.30	1.11	1.21	1.15	1.07	MIN	0.86	1.0
HMI	FTN	0.88	1.0	1.0	1.0	1.0	1.0	1.0	MAX	1.10	1.29
HMI	PSC	0.88	1.0	1.0	1.0	1.0	1.0	1.0	MAX	1.0	1.29
SCI	MOL	1.40	1.08	1.0	1.48	1.56	1.15	1.07	MIN	0.86	0.82
SCI	FTN	0.88	1.08	0.85	1.0	1.0	1.0	1.0	MAX	0.71	0.82
SCI	FTN	1.40	1.08	1.30	1.48	1.56	1.15	0.94	MIN	0.86	0.82
SCI	FTN	1.15	1.08	1.0	1.06	1.0	1.0	0.87	MIN	1.0	1.0
SCI	MOL	0.75	0.94	1.30	1.06	1.06	1.15	1.0	MID	1.0	0.91
SCI	FTN	0.88	1.08	0.85	1.0	1.0	0.87	0.87	MAX	1.19	1.0
SCI	FTN	0.88	0.94	0.70	1.0	1.06	1.0	1.0	MIN	0.86	0.82
SCI	PLI	1.0	1.0	1.15	1.0	1.0	0.87	0.87	MAX	0.71	0.91
SCI	FTN	1.0	1.0	1.15	1.0	1.0	0.87	1.0	MAX	0.71	0.82
SCI	MOL	1.0	0.94	1.30	1.0	1.0	1.0	0.87	MIN	0.86	0.82
SCI	FTN	0.88	0.94	1.0	1.0	1.0	0.87	0.87	MAX	1.0	0.82
SCI	FTN	0.88	1.04	1.07	1.0	1.06	0.87	1.07	MAX	0.86	1.0
SCI	FTN	1.0	1.04	1.07	1.0	1.21	0.87	1.07	MAX	0.86	1.0
SCI	FTN	0.88	1.04	1.07	1.06	1.21	0.87	1.07	MAX	1.0	1.0
SCI	FTN	0.88	1.04	1.07	1.0	1.06	0.87	1.07	MAX	1.0	1.0
SCI	FTN	0.88	1.04	1.07	1.0	1.06	0.87	1.07	MAX	1.0	1.0
SCI	FTN	0.75	0.94	1.30	1.0	1.0	0.87	0.87	MAX	0.71	0.82
SUP	FTN	0.88	0.94	0.85	1.0	1.0	0.87	1.0	MAX	1.19	0.91
SUP	JOV	1.0	1.0	0.85	1.0	1.0	1.0	0.87	MID	0.71	1.0
SUP	MOL	1.15	1.0	1.0	1.30	1.21	1.0	0.87	MIN	0.86	1.0
SUP	COB	0.88	1.0	1.0	1.0	1.0	1.0	1.15	MAX	1.19	1.0
SUP	MOL	0.88	0.94	0.85	1.0	1.06	1.15	1.0	MIN	1.0	1.0
SUP	MOL	0.88	0.94	1.15	1.11	1.21	1.30	1.0	MIC	0.71	1.0
SUP	FTN	1.0	0.94	1.0	1.0	1.06	1.15	0.87	MIC	1.0	0.82
SUP	FTN	0.88	0.94	0.70	1.0	1.0	0.87	0.87	MAX	0.86	0.82
SYS	MOL	1.15	0.94	1.30	1.30	1.21	1.0	1.0	MAX	0.86	0.91
SYS	MOL	1.0	0.94	1.15	1.11	1.21	1.30	1.0	MIN	1.0	1.0
SYS	MOL	1.40	0.94	1.30	1.66	1.21	1.0	1.0	MIN	0.71	0.82
SYS	HOL	1.0	0.94	1.15	1.06	1.06	1.0	0.87	MID	1.0	1.0
SYS	MOL	1.15	0.94	1.30	1.11	1.06	1.0	1.0	MAX	0.86	1.13
SYS	PSC	1.0	0.94	1.15	1.0	1.0	0.87	0.87	MAX	0.86	1.0
SYS	MOL	0.88	0.94	1.30	1.11	1.21	1.15	1.0	MIC	0.78	0.82
SYS	MOL	1.0	0.94	1.15	1.0	1.0	1.0	0.87	MIN	0.71	0.82

CONTINUAÇÃO: CONJUNTO DE DADOS COCOMO

PCAP	VEXP	LEXP	CONT	MODP	TOOL	SCHED	RVOL	MODE	ADJKDSI	MMACT
1.17	1.10	1.0	NOM	1.24	1.10	1.04	1.19	E	113	2040
1.0	0.90	0.95	NOM	1.10	1.0	1.0	1.0	E	249	1600
0.86	0.90	0.95	NOM	0.91	0.91	1.0	1.0	SD	132	243
1.42	1.0	0.95	NOM	1.24	1.0	1.04	1.19	ORG	46	240
0.86	0.90	0.95	NOM	1.24	1.0	1.0	1.0	ORG	16	33
1.42	0.90	0.95	HI	1.24	1.10	1.0	1.19	ORG	4.0	43
1.0	0.90	0.95	HI	0.91	0.91	1.0	1.0	ORG	6.9	8
1.0	1.21	1.14	NOM	1.10	1.10	1.08	1.38	E	22	1075
0.86	1.10	1.07	NOM	0.91	1.0	1.0	1.19	E	30	423
0.86	0.90	1.0	NOM	1.0	1.0	1.0	1.38	E	18	321
0.86	0.90	1.0	NOM	1.0	1.0	1.0	1.38	E	20	218
0.86	1.0	0.95	NOM	0.91	1.0	1.08	1.19	E	37	201
0.70	1.10	1.0	HI	0.82	1.0	1.0	1.0	E	24	79
0.70	1.10	1.07	NOM	1.10	1.24	1.23	1.19	SD	3.0	73
0.86	1.21	1.14	NOM	0.91	1.0	1.23	1.19	E	3.9	61
0.86	1.0	1.0	NOM	1.0	1.0	1.0	1.19	E	3.7	40
0.86	1.0	1.0	NOM	1.0	1.0	1.0	0.91	E	1.9	9
1.0	1.0	1.0	LO	1.24	1.10	1.08	1.19	E	320	11400
1.0	1.0	1.0	NOM	0.91	0.91	1.0	1.0	E	966	6600
1.08	1.10	1.07	NOM	1.24	1.0	1.08	1.19	SD	287	6400
1.0	1.0	1.0	HI	0.91	0.91	1.0	1.0	E	252	2455
0.86	0.90	1.0	HI	0.91	1.0	1.23	1.0	E	109	724
0.86	0.90	1.0	LO	1.0	1.0	1.23	1.0	E	75	539
1.0	1.10	0.95	NOM	0.82	0.83	1.0	1.0	SD	90	453
0.86	1.10	1.0	NOM	0.82	0.91	1.08	1.62	E	38	523
0.86	1.10	1.0	NOM	0.91	1.10	1.08	1.19	E	48	387
1.0	1.0	1.0	HI	0.82	1.10	1.08	1.19	E	9.4	88
0.86	1.10	1.07	NOM	1.10	1.10	1.0	1.0	ORG	13	98
0.86	1.0	1.0	HI	0.91	0.91	1.23	0.91	SD	2.14	7.3
0.86	1.0	1.0	HI	0.91	0.91	1.23	0.91	SD	1.98	5.9
0.86	1.10	1.07	NOM	1.0	1.0	1.0	1.0	E	50	1063
1.0	1.0	1.0	NOM	1.10	1.10	1.0	1.0	SD	261	702
0.86	0.90	1.0	HI	0.91	0.91	1.0	1.0	E	40	605
1.0	1.0	1.0	NOM	0.91	1.10	1.23	1.19	E	22	230
1.0	1.10	1.0	NOM	1.24	1.24	1.0	1.19	E	13	82
1.17	0.90	0.95	NOM	1.0	0.91	1.04	1.0	SD	12	55
0.86	1.0	1.0	NOM	1.0	1.0	1.0	0.91	ORG	34	47
1.0	0.90	0.95	NOM	0.82	0.91	1.0	1.0	ORG	15	12
0.70	1.0	0.95	HI	0.91	1.10	1.0	1.0	ORG	6.2	8
1.17	1.0	1.0	NOM	1.10	1.0	1.0	1.0	ORG	2.5	8
0.70	0.90	0.95	HI	0.91	0.91	1.0	1.0	ORG	5.3	6
0.93	0.90	0.95	HI	0.95	0.95	1.04	1.0	ORG	19.5	45
1.0	0.90	0.95	HI	1.0	1.0	1.04	1.0	ORG	28	83
1.0	0.90	0.95	HI	1.10	1.0	1.04	1.0	ORG	30	87
1.0	0.90	0.95	HI	1.0	0.95	1.04	1.0	ORG	32	106
0.86	0.90	0.95	NOM	1.0	1.0	1.04	1.0	ORG	57	126
0.70	1.10	1.07	NOM	1.10	1.0	1.04	1.0	ORG	23	36
1.17	0.90	0.95	NOM	1.10	1.0	1.04	1.09	SD	311	1272
0.70	1.10	1.0	HI	0.82	0.91	1.0	1.19	SD	91	156
0.86	1.10	1.0	NOM	1.0	1.0	1.0	1.19	E	24	176
1.42	1.0	0.95	LO	1.24	1.10	1.04	1.38	ORG	10	122
1.0	1.10	1.07	NOM	1.24	1.10	1.0	1.38	ORG	8.2	41
0.70	1.10	1.07	NOM	1.0	1.10	1.08	1.0	SD	5.3	14
1.0	1.0	0.95	HI	0.91	1.10	1.0	1.19	ORG	4.4	20
1.17	0.90	0.95	NOM	1.10	1.0	1.0	1.0	ORG	6.3	18
1.0	1.10	1.07	NOM	1.10	1.10	1.08	1.38	E	27	958
1.0	1.10	1.07	LO	1.10	1.10	1.23	1.0	E	15	237
0.70	0.90	0.95	NOM	0.91	1.0	1.0	1.0	E	25	130
1.0	1.0	1.0	NOM	0.91	1.0	1.0	0.91	ORG	21	70
0.86	1.10	1.07	NOM	1.10	1.10	1.08	1.19	ORG	6.7	57
0.86	0.90	1.0	HI	0.82	1.0	1.0	1.0	ORG	28	50
0.70	1.21	1.14	NOM	0.91	1.24	1.0	1.0	SD	9.1	38
0.86	1.0	1.0	NOM	0.82	1.0	1.0	1.0	E	10	15

APÊNDICE B - FATORES DE AJUSTE E NÍVEIS DAS VARIÁVEIS

Variable	Adjustment factor	Level assigned
TYPE	(BUS)	1
	(CTL)	2
	(HMI)	3
	(SCI)	4
	(SUP)	5
	(SYS)	6
LANG	COB	1
	PLI	2
	FTN	3
	MOL	4
	JOV	5
	HOL	6
	PSC	7
RELY	Very low = 0.75	1
	low = 0.88;0.94	2
	nominal = 1	3
	high = 1.15	4
	Very high = 1.4	5
DATA	low = 0.94	1
	nominal = 1	2
	high = 1.04	3
	Very high = 1.08	4
	Extra high = 1.16	5
CPLX	Very low = 0.7	1
	low = 0.85	2
	nominal = 1	3
	high = 1.07	4
	Very high = 1.15	5
	Extra high = 1.30; 165	6
STOR	nominal = 1	1
	high = 1.06	2
	Very high = 1.14	3
	Extra high = 1.21	4
	Extra-extra high = 1.56	5
VIRT	Low = 0.87	1
	Nominal = 1	2
	High = 1.15	3
	Very high = 1.30	4

TURN	Very low = 0.87	1
	low = 0.94	2
	nominal = 1	3
	high = 1.07	4
	Very high = 1.15	5
PLATF	MAX	1
	MID	2
	MIN	3
	MIC	4
ACAP	Extra high = 0.71	1
	Very high= 0.78;0.86	2
	High = 1	3
	Nominal = 1.10;1.19	4
	Low = 1.46	5
AEXP	Very high = 0.82	1
	High = 0.91	2
	Nominal = 1	3
	Low = 1.13	4
	Very Low = 1.29	5
PCAP	Very high = 0.7	1
	high = 0.86 e0.93	2
	nominal =1	3
	Below average = 0.8; 1.17; 1.42	4
VEXP	high = 0.9	1
	nominal =1	2
	low = 1.1	3
	Very low = 1.21	4
LEXP	high = 0.95	1
	nominal = 1	2
	low = 1.07	3
	Very low =	4
MODP	Very high = 0.82	1
	high = 0.91 e 0.95	2
	nominal = 1	3
	low = 1.1	4
	Very low = 1.24	5
TOOL	Very high = 0.83	1
	high = 0.91; 0.95	2
	nominal = 1	3
	low = 1.1	4
	Very low = 1.24	5
SCHED	nominal= 1	1
	High = 1.04	2

	Very high = 1.08	3
	Extra high = 1.23	4
RVOL	Low = 0.91	1
	nominal = 1; 1.09	2
	High = 1.19	3
	Very high = 1.38;1.62	4
MODE	Organic	1
	Semidetashed	2
	Embedded	3
CONT	LO	1
	NOM	2
	HI	3

CONJUNTO DE DADOS APÓS O PRÉ-PROCESSAMENTO

TYPE	LANG	RELY	DATA	CPLX	TIME	STOR	VIRT	TURN	PLATFORI	ACAP	AEXP
1	1	2	5	1	1	2	3	4	1	4	4
1	1	2	5	2	1	2	2	4	1	4	2
1	2	4	5	2	1	1	1	2	2	3	1
1	1	1	5	1	1	1	1	3	2	3	2
1	3	2	1	3	1	1	1	3	1	2	3
1	2	1	2	2	1	4	2	3	2	4	3
1	1	1	2	3	1	1	1	1	1	3	3
2	4	5	1	6	5	5	4	3	3	1	2
2	3	5	1	6	4	4	3	3	3	2	3
2	4	6	1	6	3	5	2	4	3	2	1
2	4	6	1	6	3	5	2	4	3	2	1
2	5	5	1	6	3	2	2	3	3	2	1
2	3	5	1	6	3	2	3	3	2	1	3
2	4	5	1	7	4	5	3	3	4	2	3
2	4	6	1	6	4	2	3	1	3	2	4
2	4	6	2	6	4	5	2	1	3	2	3
2	4	6	2	6	4	5	2	1	3	2	1
3	3	5	5	5	4	4	2	4	1	2	3
3	3	5	4	3	3	4	1	2	1	1	2
3	5	6	4	6	3	4	3	4	1	1	1
3	2	4	5	5	2	3	1	1	1	2	3
3	5	5	2	3	4	2	2	3	1	2	1
3	6	5	2	3	2	2	2	3	1	2	1
3	3	2	2	2	2	2	2	1	3	3	5
3	5	5	5	6	3	2	2	1	1	2	3
3	6	3	2	2	2	2	3	4	1	2	3
3	3	5	1	5	4	4	2	1	3	3	3
3	4	5	4	6	3	4	3	4	3	2	3
3	3	2	2	3	1	1	2	3	1	4	5
3	7	2	2	3	1	1	2	3	1	3	5
4	4	6	4	3	5	5	3	4	3	2	1
4	3	2	4	2	1	1	2	3	1	1	1
4	3	6	4	6	5	5	3	2	3	2	1
4	3	5	4	3	2	1	2	1	3	3	3
4	4	1	1	6	2	4	3	3	2	3	2
4	3	2	4	2	1	1	1	1	1	4	3
4	3	2	1	1	1	2	2	3	3	2	1
4	2	4	2	5	1	1	1	1	1	1	2
4	3	4	2	5	1	1	1	3	1	1	1
4	4	4	1	6	1	1	2	1	3	2	1
4	3	2	1	3	1	1	1	1	1	3	1
4	3	2	3	4	1	2	1	4	1	2	3
4	3	4	3	4	1	4	1	4	1	2	3
4	3	2	3	4	2	4	1	4	1	3	3
4	3	2	3	4	1	2	1	4	1	3	3
4	3	2	3	4	1	2	1	4	1	3	3
4	3	1	1	6	1	1	1	1	1	1	1
5	3	2	1	2	1	1	1	3	1	4	2
5	5	4	2	2	1	1	2	1	2	1	3
5	4	5	2	3	4	4	2	1	3	2	3
5	1	2	2	3	1	1	2	5	1	4	3
5	4	2	1	2	1	2	3	3	3	3	3
5	4	2	1	5	3	4	4	3	4	1	3
5	3	4	1	3	1	2	3	1	4	3	1
5	3	2	1	1	1	1	1	1	1	2	1
6	4	5	1	6	4	4	2	3	1	2	2
6	4	4	1	5	3	4	4	3	3	3	3
6	4	6	1	6	5	4	2	3	3	1	1
6	6	4	1	5	2	2	2	1	2	3	3
6	4	5	1	6	3	2	2	3	1	2	4
6	7	4	1	5	1	1	1	1	1	2	3
6	4	2	1	6	3	4	3	3	4	2	1
6	4	4	1	5	1	1	2	1	3	1	1

CONTINUAÇÃO: CONJUNTO DE DADOS APÓS O PRÉ-PROCESSAMENTO

PCAP	VEXP	LEXP	CONT	MODP	TOOL	SCHED	RVOL	MODE	ADJKDSI	LnADJKDS	MMACT	LnMMACT
4	3	2	2	5	4	2	3	3	113	4,7274	2040	7,6207
3	1	1	2	4	3	1	2	3	249	5,5175	1600	7,3778
2	1	1	2	2	2	1	2	2	132	4,8828	243	5,4931
4	2	1	2	5	3	2	3	1	46	3,8286	240	5,4806
2	1	1	2	5	3	1	2	1	16	2,7726	33	3,4965
4	1	1	3	5	4	1	3	1	4	1,3863	43	3,7612
3	1	1	3	2	2	1	2	1	6,9	1,9315	8	2,0794
3	4	4	2	4	4	3	4	3	22	3,0910	1075	6,9801
2	3	3	2	2	3	1	3	3	30	3,4012	423	6,0474
2	1	2	2	3	3	1	4	3	18	2,8904	321	5,7714
2	1	2	2	3	3	1	4	3	20	2,9957	218	5,3845
2	2	1	2	2	3	3	3	3	37	3,6109	201	5,3033
1	3	2	3	1	3	1	2	3	24	3,1781	79	4,3694
1	3	3	2	4	5	4	3	2	3	1,0986	73	4,2905
2	4	4	2	2	3	4	3	3	3,9	1,3610	61	4,1109
2	2	2	2	3	3	1	3	3	3,7	1,3083	40	3,6889
2	2	2	2	3	3	1	1	3	1,9	0,6419	9	2,1972
3	2	2	1	5	4	3	3	3	320	5,7683	11400	9,3414
3	2	2	2	2	2	1	2	3	966	6,8732	6600	8,7948
4	3	3	2	5	3	3	3	2	287	5,6595	6400	8,7641
3	2	2	3	2	2	1	2	3	252	5,5294	2455	7,8059
2	1	2	3	2	3	4	2	3	109	4,6913	724	6,5848
2	1	2	1	3	3	4	2	3	75	4,3175	539	6,2897
3	3	1	2	1	1	1	2	2	90	4,4998	453	6,1159
2	3	2	2	1	2	3	4	3	38	3,6376	523	6,2596
2	3	2	2	2	4	3	3	3	48	3,8712	387	5,9584
3	2	2	3	1	4	3	3	3	9,4	2,2407	88	4,4773
2	3	3	2	4	4	1	2	1	13	2,5649	98	4,5850
2	2	2	3	2	2	4	1	2	2,14	0,7608	7,3	1,9879
2	2	2	3	2	2	4	1	2	1,98	0,6831	5,9	1,7750
2	3	3	2	3	3	1	2	3	50	3,9120	1063	6,9689
3	2	2	2	4	4	1	2	2	261	5,5645	702	6,5539
2	1	2	3	2	2	1	2	3	40	3,6889	605	6,4052
3	2	2	2	2	4	4	3	3	22	3,0910	230	5,4381
3	3	2	2	5	5	1	3	3	13	2,5649	82	4,4067
4	1	1	2	3	2	2	2	2	12	2,4849	55	4,0073
2	2	2	2	3	3	1	1	1	34	3,5264	47	3,8501
3	1	1	2	1	2	1	2	1	15	2,7081	12	2,4849
1	2	1	3	2	4	1	2	1	6,2	1,8245	8	2,0794
4	2	2	2	4	3	1	2	1	2,5	0,9163	8	2,0794
1	1	1	3	2	2	1	2	1	5,3	1,6677	6	1,7918
2	1	1	3	2	2	2	2	1	19,5	2,9704	45	3,8067
3	1	1	3	3	3	2	2	1	28	3,3322	83	4,4188
3	1	1	3	4	3	2	2	1	30	3,4012	87	4,4659
3	1	1	3	3	2	2	2	1	32	3,4657	106	4,6634
2	1	1	2	3	3	2	2	1	57	4,0431	126	4,8363
1	3	3	2	4	3	2	2	1	23	3,1355	36	3,5835
4	1	1	2	4	3	2	2	2	311	5,7398	1272	7,1483
1	3	2	3	1	2	1	3	2	91	4,5109	156	5,0499
2	3	2	2	3	3	1	3	3	24	3,1781	176	5,1705
4	2	1	1	5	4	2	4	1	10	2,3026	122	4,8040
3	3	3	2	5	4	1	4	1	8,2	2,1041	41	3,7136
1	3	3	2	3	4	3	2	2	5,3	1,6677	14	2,6391
3	2	1	3	2	4	1	3	1	4,4	1,4816	20	2,9957
4	1	1	2	4	3	1	2	1	6,3	1,8405	18	2,8904
3	3	3	2	4	4	3	4	3	27	3,2958	958	6,8648
3	3	3	1	4	4	4	2	3	15	2,7081	237	5,4681
1	1	1	2	2	3	1	2	3	25	3,2189	130	4,8675
3	2	2	2	2	3	1	1	1	21	3,0445	70	4,2485
2	3	3	2	4	4	3	3	1	6,7	1,9021	57	4,0431
2	1	2	3	1	3	1	2	1	28	3,3322	50	3,9120
1	4	4	2	2	5	1	2	2	9,1	2,2083	38	3,6376
2	2	2	2	1	4	1	2	3	10	2,3026	15	2,7081

APÊNDICE C – QUESTIONÁRIO DE COLETA DE DADOS

Os dados dos projetos fornecidos pela CELEPAR e pelo grupo de desenvolvimento de software do departamento de sistemas de solos do INPE foram coletados com base nas respostas a um questionário. Este questionário foi elaborado baseando-se nas descrições do conjunto de dados COCOMO, fornecidas por Boehm (1981). O objetivo foi coletar informações de projetos para testar os modelos gerados a partir deste mesmo conjunto de dados. O questionário é constituído do nome da variável, de uma questão e da medida (que é a resposta à questão relacionada), conforme apresentado a seguir.

Nome da variável: **ID**
Questão: Qual o nome que identifica o projeto a ser desenvolvido?
Medida: O nome do projeto.

Nome da variável: **MMACT**
Questão: Qual o esforço real do projeto, ou seja, qual a quantidade de trabalho realizado pelo pessoal de desenvolvimento de software desde a especificação até a entrega do produto, medido em horas?
Medida: A quantidade de horas/ homem utilizadas para desenvolver o projeto.

Nome da variável: **ADJKDSI**
Questão: Qual o tamanho da aplicação, dado em número de linhas de código, pontos por função ou pontos por caso de uso?
Medida: A quantidade de linhas de código ou pontos por função ou pontos por caso de uso.

Nome da variável: **TYPE**

Questão: Qual o tipo de aplicação que será desenvolvida?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Serviço de cliente

( ) 2. Sistema de informação gerencial

( ) 3. Processamento de transação

( ) 4. Controle de produção, logística, processamento de pedido

( ) 5. Serviço on-line/ Informação

( ) 6. Outros

Nome da variável: **LANG**

Questão: Qual a linguagem de programação utilizada para o desenvolvimento da aplicação?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. COB	( ) 4. MOL	( ) 7. PSC
( ) 2. PLI	( ) 5. JOV	
( ) 3. FTN	( ) 6. HOL	

Nome da variável: **RELY**

Questão: Qual o nível de confiabilidade requerida do software?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Muito baixa: O efeito de uma falha é simplesmente um inconveniente no que diz respeito a incumbência do pessoal de desenvolvimento para resolver a falha.

( ) 2. Baixa: o efeito de uma falha de software é um nível baixo, com perdas facilmente superadas pelos usuários.

( ) 3. Media: o efeito de uma falha de software é uma perda moderada para os usuários, mas a situação a partir da qual uma pessoa pode superar sem penalidades extremas. Exemplos típicos são sistemas de informação gerenciais.

( ) 4. Alta: o efeito de uma falha de software pode ser uma perda financeira maior ou um inconveniente para a humanidade. Exemplos são sistemas bancários e sistemas de distribuição de força elétrica.
( ) 5. Muito alta: o efeito de uma falha de software pode ser a perda de vida humana. Exemplos são sistemas de controle de aeronaves ou sistemas de controle de reatores nucleares.

Nome da variável: **DATA**
Questão: Qual o tamanho da base de dados do projeto?
Medida: Valor inteiro correspondente a uma das seguintes opções:
( ) 1. Baixa
( ) 2. Media
( ) 3. Alta
( ) 4. Muito alta
( ) 5. Extra Alta

Nome da variável: **CPLX**
Questão: Qual o nível de complexidade do software?
Medida: Valor inteiro correspondente a uma das seguintes opções:
( ) 1. Muito baixa: somente rotinas; nenhuma necessidade de interface de usuário, base de dados simples.
( ) 2. Baixo: funcionalidade clara, solução de base de dados clara.
( ) 3. Média: funcionalidade típica, normal base de dados padrão.
( ) 4. Alta: demanda maior processamento; base de dados grande e complexa; novos requisitos para interfaces de usuário.
( ) 5. Muito alta: solução difícil funcional e tecnicamente; interface do usuário muito complexa; bases de dados distribuídas.
( ) 6. Extra alta

Nome da variável: **TIME**

Questão: Qual o nível de restrição de tempo de execução do sistema?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Muito baixo

( ) 2. Baixo

( ) 3. Médio

( ) 4. Alto

( ) 5. Muito alto

Nome da variável: **STOR**

Questão: Qual a restrição de armazenamento principal?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Baixa

( ) 2. Media

( ) 3. Alta

( ) 4. Muito alta

( ) 5. Extra alta

Nome da variável: **VIRT**

Questão: Qual a volatilidade da máquina virtual (o complexo de hardware e software)?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Baixa

( ) 2. Media

( ) 3. Alta

( ) 4. Muito alta

Nome da variável: **TURN**

Questão: Qual o tempo de resposta do computador utilizado no desenvolvimento do sistema?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ). 1. Muito baixa

( ) 2. Baixa

( ) 3. Media

( ) 4. Alta

( ) 5. Muito alta

Nome da variável: **PLATFORM**

Questão: Qual a plataforma de hardware utilizada no desenvolvimento?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Rede

( ) 2. Mainfraime

( ) 3. Computador pessoal

( ) 4. Mini- computador

( ) 5. Multi-plataforma

Nome da variável: **ACAP**

Questão: Qual o nível de experiência da equipe de análise do sistema?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Muito baixa: nenhuma experiência em análise de requisitos ou projetos similares

( ) 2. Baixa: menos de 30% do pessoal de projeto com experiência em análise e projeto em projetos similares

( ) 3. Media: de 30 a 70% do pessoal de projeto com experiência em análise; um número experiente. Um membro experiente

( ) 4. Alta: a maioria dos membros de pessoal de projeto com experiência em especificações e análise; profissionais de análise em mudança

( ) 5. Muito alta: pessoal de projeto composto de profissionais de primeira classe; membros têm forte visão e experiência com analise de requisitos

Nome da variável: **AEXP**

Questão: Qual o nível de experiência da equipe com o tipo de aplicação relacionada ao projeto?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Muito baixa

( ) 2. Baixa

( ) 3. Media

( ) 4. Alta

( ) 5. Muito alta

Nome da variável: **PCAP**

Questão: Qual a capacidade do programador?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Muito baixa: equipe não tem experiência com as técnicas de programação necessárias; a experiência da equipe é em média menor que 6 meses

( ) 2. Baixa: experiência com as técnicas de programação é menor que a média; alguns membros têm experiência com algumas ferramentas,. De 6 a 12 meses, em média

( ) 3. Media:mais ou menos a metade de equipe tem boa experiência com as técnicas de programação; alguns membros conhecem bem as ferramentas de documentação e desenvolvimento; em média de 1 a 3 anos

( ) 4. Alta: a maioria dos membros da equipe conhece bem as técnicas de programação; alguns podem ajudar os outros; de 3 a 6 anos em media.

( ) 5. Muito alta: a equipe conhece todas as técnicas de programação bem; suporte disponível para necessidades especificas de projeto; experiência media maior que 6 anos.

Nome da variável: **VEXP**

Questão: Qual o nível de experiência da equipe com máquina virtual?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Muito baixa

( ) 2. Baixa

( ) 3. Media

( ) 4. Alta

Nome da variável: **LEXP**

Questão: Qual o nível de experiência da equipe com a linguagem de programação utilizada no projeto?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Muito baixa

( ) 2. Baixa

( ) 3. Media

( ) 4. Alta

Nome da variável: **MODP**

Questão: Em que proporções são utilizadas práticas modernas de programação?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Muito baixa

( ) 2. Baixa

( ) 3. Media

( ) 4. Alta

( ) 5. Muito alta

Nome da variável: **TOOL**

Questão: Em que proporções são utilizadas ferramentas modernas no desenvolvimento de software?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Muito baixa

( ) 2. Baixa

( ) 3. Media

( ) 4. Alta

( ) 5. Muito alta

Nome da variável: **SCHED**

Questão: Qual o nível de restrição no cronograma, que é imposto sobre a equipe de projeto que desenvolve o sistema?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Muito baixo:cronograma bastante flexível

( ) 2. Baixo: cronograma flexível

( ) 3. Médio: cronograma de flexibilidade média

( ) 4. Alto: cronograma com exigências (alta probabilidade) alta de se cumprir ou até mesmo diminuir o tempo de desenvolvimento

( ) 5.Muito alto: cronograma com exigências (muito alta probabilidade) muito alta de se cumprir ou até mesmo diminuir o tempo de desenvolvimento

Nome da variável: **RVOL**

Questão: Qual o nível de volatilidade dos requisitos do usuário durante o projeto?

Medida: Valor inteiro correspondente a uma das seguintes opções:

( ) Baixa: algumas mudanças nas especificações; algumas funções adaptadas ou novas; algumas pequenas mudanças em conteúdos de dados

( ) Media: algumas mudanças nas especificações mas os membros do projeto podem lidar bem com elas; impacto menor que (funções novas ou modificadas são menores que 15%)

( ) Alta: algumas mudanças maiores afetam a arquitetura geral e requer um re-trabalho; tem 15 a 30% das funções novas ou modificadas

( ) Muito alta: novos requisitos adicionados continuamente; muito re-trabalho; mais que 30% das funções são novas ou modificadas comparada aos requisitos originais.

Nome da variável: **MODE**

Questão: Qual o modo de desenvolvimento do software?

Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Orgânico (significa que o software foi produzido por uma equipe relativamente pequena, in-house, em ambiente familiar

( ) 2. Semi-destacado (significa que o software foi produzido em um modo intermediário entre o orgânico e o embutido

( ) 3. Embutido (significa que o software foi produzido dentro de um ambiente extremamente complexo de software, hardware, procedimentos operacionais e regulamentos. Em geral, sistemas de controle de tráfego aéreo.

Nome da variável: **CONT**

Questão: Qual o nível de rotatividade da equipe no projeto?

Valor inteiro correspondente a uma das seguintes opções:

( ) 1. Menos que 10% de mudança (rotatividade)

( ) 2. De 10% a 30% de mudança (rotatividade)

( ) 3. Mais que 30% de mudança (rotatividade)