PONTÍFICIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Silvia Cristina Nunes das Dores

Um Modelo para Estimativas de Esforço em Projetos de Reengenharia de Sistemas Legados

Orientador: Prof. Dr. Duncan Dubugras Alcoba Ruiz

PLANO DE ESTUDO E PESQUISA

Porto Alegre
2013

LISTA DE ABREVIATURAS E SIGLAS

COCOMO – COstructive COst Model

SLIM – Software Life-Cycle Model - SLIM

RBC – Raciocínio Baseado em Casos

RNA – Redes Neurais Artificiais

ML – *Machine Learning*

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Reengenharia de Sistemas é também conhecida como a renovação ou reconstrução de software, e envolve a análise e mudança de um sistema de software para reconstituí-lo em uma nova forma [CC90] [Sne95]. Partindo-se do sistema legado existente (via código-fonte, interface ou ambiente), são abstraídas as suas funcionalidades e são construídos o modelo de análise e o projeto do software.

Um Sistema Legado é qualquer sistema de informação que resiste significativamente a modificações e alterações. Geralmente um sistema legado tem milhões de linhas de código e mais de 10 anos de idade, embora isto nem sempre seja verdadeiro, já que um sistema recentemente desenvolvido, mas que não pode ser prontamente modificado para se adaptar a constantes mudanças nos requisitos de negócio, também pode ser considerado legado [BS95].

Realizar a reengenharia de sistemas legados não é um processo trivial e empresas enfrentam uma série de problemas a fim de garantir a sustentabilidade através da reestruturação e migração de sistemas legados para plataformas mais modernas, uma vez que, por vezes, estas medidas são complexas e têm um alto risco e custo associados. Portanto, é importante entender e planejar estrategicamente a modernização do sistema com foco na reengenharia desses sistemas legados [Sne05].

Apesar da importância da reengenharia, um estudo realizado por [BSTWW05] concluiu que mais de 50 % dos projetos de reengenharia falham. As razões para o fracasso em projetos de reengenharia incluem o uso de uma estratégia inadequada, requisitos mal coletados, planejamento inadequado e fracasso para dar conta da complexidade do sistema a ser reconstruído.

Estimar adequadamente o esforço para a realização da reengenharia é uma atividade crucial durante o planejamento da mesma, pois conecta medições para custo e cronograma do projeto. A determinação do esforço necessário para a realização do projeto permite planejar adequadamente quaisquer futuras atividades e é realizada por uma variedade de razões como a seleção de projetos, planejamento de recursos,

programação, monitoramento de status, avaliação de desempenho da equipe, dentre outras [Som07] [Pre11].

De acordo com a revisão sistemática realizada nesta pesquisa (Capítulo 6), apesar de existirem diversas soluções bem consolidas para apoio a realização de estimativas de esforço em projetos de desenvolvimento de software, poucas dessas soluções são designadas especificamente para o contexto de reengenharia de sistemas.

Neste contexto, esta pesquisa objetiva propor um modelo de estimativas de esforço atenda diretamente as características e etapas de um projeto de reengenharia de sistemas, e cuja formulação esteja alinhada com as práticas atualmente aplicadas na indústria e na literatura vigente sobre o tema.

1.1 ORGANIZAÇÃO DO PLANO

Este plano está organizado em 7 capítulos, da seguinte forma:

- CAPÍTULO 1 – apresenta introdução do trabalho;
- CAPÍTULO 2 – apresenta a justificativa para a realização do mesmo;
- CAPÍTULO 3 – apresenta os objetivos geral e específicos do trabalho;
- CAPÍTULO 4 – apresenta a fundamentação teórica do trabalho, no que diz respeito a estimativas de esforço e projetos de reengenharia de sistemas;
- CAPITULO 5 – apresenta a metodologia a ser utilizada;
- CAPITULO 6 – apresenta o andamento da pesquisa;
- CAPÍTULO 7 – apresenta o cronograma de atividades para o segundo ano.

2 JUSTIFICATIVA

Em um estudo realizado pelo *The Standish Group* em 1995, denominado de *Chaos Report*, cujo foco foi a indústria de software comercial, foi detectado que 31,1% dos projetos de software foram cancelados, antes mesmo de serem concluídos e 52,7% custaram 189% a mais do que o previsto inicialmente, enquanto que apenas 16,2% dos projetos são completados no tempo e custo estimados. Embora muitos destes projetos entregues não reflitam o que foi especificado, uma vez que apenas 42% dos requisitos originais são desenvolvidos.

Embora essa pesquisa tenha sido realizada na década de 90, ainda hoje estes números são reais na indústria de software. O *Chaos Report* de 2012 identifica que apesar de 39% dos projetos serem entregues no prazo, orçamentos e terem recursos necessários para as funções, 43% dos projetos enfrentam situações opostas a esta, pois estão atrasados, acima do orçamento e/ou sem os recursos necessários para sua conclusão e 18% foram cancelados antes do término ou entregues e nunca utilizados [SG12].

Dentre os fatores identificados pelo *The Standish Group* [SG12] como sendo determinantes do insucesso do projeto está predominantemente o estouro do tempo e/ou do orçamento. Os dados mostram que em 2012 cerca de 74% dos projetos ultrapassaram o tempo previsto, enquanto 59% ultrapassaram o orçamento.

Estes dados de projetos de software reais evidenciam o que a literatura acerca de Gerenciamento de Projetos já afirma amplamente: que a estimativa de custos de desenvolvimento de software representa uma importante área de pesquisa e é uma das tarefas mais relevantes no planejamento e gerenciamento de processo de desenvolvimento de software [Som07] [Pre11] [Pfl04].

Dada a relevância da realização de estimativas de esforço muitas soluções foram propostas no intuito de auxiliar este processo, estas soluções envolvem modelos algorítmicos, como, por exemplo, COCOMO [Boe81] [Boe00] e SLIM [Put78], Julgamento de Especialistas [TMKK12] [Jor07] [Jor05], Regressão Estatística [NSB08] [SYB08] [RTR08], Modelos Dinâmicos e soluções baseadas em aprendizagem de máquina, que incluem, Árvores de Regressão e Classificação [BBS13] [BBR12], Lógica

Fuzzy [KJHK11] [ZKKN13] [SMAKT11] [ANC09] [BMDP12] [SS12], Redes Neurais Artificiais [AO12] [KTB08] [SM08], Algoritmos Genéticos [BDSM12] [SMLE02] [LC11], Estimativa por Analogia [SSK96] [MC00] [WLT09], dentre outras.

Com o avanço da tecnologia, muitas mudanças estão acontecendo na indústria de software e, portanto, as aplicações de software têm que estar em sintonia com as mais recentes tecnologias e tendências para permanecer competitivas no mercado. A exigência de manter o ritmo do mercado e cumprir requisitos adicionais demonstra a necessidade de reengenharia de sistemas, onde o software é transformado de um estado para outro estado.

Um processo de reengenharia tem características que implicam em novas exigências do modelo de estimativa, como componentes de software mais detalhados (ou seja, código) do que aqueles normalmente utilizados nos processos de desenvolvimento (requisitos de sistema e documentos de análise) [Pre11]. Além disso, inclui atividades que dependem de parâmetros de qualidade do sistema legado [Vis01]. Nas comunidades científica e industrial os processos de reengenharia têm uma história mais recente em comparação ao desenvolvimento e manutenção ordinária. Por estas razões, os projetos de reengenharia têm maiores riscos em termos de pessoas, programação e custos [BCV03]. Assim, o monitoramento intenso durante a execução do projeto deve ser realizado a fim de gerenciar os riscos de subestimação e superestimação.

No contexto de apoio a realização de estimativa de esforço em projetos de reengenharia, nota-se poucos trabalhos que relacionem propostas de apoio a realização de estimativas de esforço em projetos de reengenharia de software concebidas com base na realização desta tarefa na prática, de maneira aproximar a solução proposta da realidade organizacional.

Assim, este trabalho pretende atuar junto a esta lacuna, propondo um modelo para apoiar a realização de estimativas de esforço em projetos de reengenharia de sistemas, baseado no estado da arte das técnicas de estimativa de esforço existentes e em boas práticas de estimativa aplicadas em projetos reais.

Para nortear o estudo foi definida a seguinte questão: “**Como realizar estimativas de esforço em projetos de reengenharia de sistemas, de forma a alcançar boas aproximações em relação ao esforço real?**”, que serve não apenas para

identificar o principal objetivo do trabalho, que é propor uma solução para este problema, como para limitar o escopo de aplicação do mesmo.

3 OBJETIVOS

Esta seção tem como objetivo apresentar os objetivos geral e específicos relacionados a este trabalho.

3.1 OBJETIVO GERAL

O objetivo geral deste trabalho é propor um modelo para apoiar a realização de estimativas de esforço em projetos de reengenharia de sistemas.

3.2 OBJETIVOS ESPECÍFICOS

Como objetivos específicos relacionados ao geral tem-se:

- Estudar as soluções propostas para realizar estimativa de esforço em projetos de desenvolvimento de software, especialmente em projetos de reengenharia de sistemas, existentes na literatura;
- Identificar práticas para realização de estimativa de esforço em projetos de reengenharia de sistemas aplicadas em organizações de desenvolvimento de software;
- Propor um modelo para realização de estimativa de esforço em projetos de reengenharia, a partir das soluções dispostas na literatura e nas práticas identificadas nas organizações estudadas;
- Validar o modelo proposto aplicando-o sobre dados reais de projetos de reengenharia, a fim de comprovar sua aplicabilidade e eficiência em relação a outras soluções.

4 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a base teórica referente aos principais conceitos que envolvem a proposta deste plano de pesquisa. Na seção 4.1, serão abordados os conceitos referentes ao processo de Reengenharia de Sistemas, conceitos, abordagens, modelo de processo. Na seção 4.2 estimativa de esforço é apresentada com as principais técnicas adotadas. Por fim, são apresentados os trabalhos relacionados.

4.1 REENGENHARIA DE SISTEMAS

Software evolui independente do domínio da aplicação, tamanho ou complexidade. As mudanças dirigem esse processo. No âmbito do software, ocorrem alterações quando são corrigidos erros, quando há adaptação a um novo ambiente, quando o cliente solicita novas características ou funções e quando a aplicação passa por um processo de reengenharia para proporcionar beneficio em um contexto moderno [Pre11].

A manutenção se torna difícil a medida em que os anos passam. Chega então um momento que o software precisa de reforma. Precisa ser transformado em um produto com mais funcionalidades, melhor desempenho e confiabilidade e de manutenção mais fácil, este processo é chamado de reengenharia.

A reengenharia de software está relacionada à reimplementação de sistemas legados para torna-los mais fácil de manter. A reengenharia pode envolver uma nova documentação, organização e reestruturação do sistema, conversão do sistema em uma linguagem mais moderna, modificação e atualização da estrutura e dos valores dos dados de sistema. A funcionalidade do software não é alterada e, normalmente, a arquitetura do sistema também permanece a mesma [Som07].

Rosenberg e Hyatt (1996) define reengenharia como sendo o exame, análise e alteração de um sistema de software existente para reconstituí-lo em uma nova forma, e a posterior implementação da nova forma. O processo geralmente envolve uma combinação de outros processos, como a engenharia reversa, redocumentação, reestruturação, tradução e engenharia direta. O objetivo é entender o software existente (especificação, projeto, implementação) e reimplementá-lo para melhorar a funcionalidade, o desempenho ou a implementação do sistema [RH96].

A distinção principal entre reengenharia e desenvolvimento de um novo software é o ponto de partida para o desenvolvimento. Em vez de começar com uma especificação escrita, o sistema antigo funciona como uma especificação para o sistema novo [Pfl04].

4.1.1 Objetivos da Reengenharia

O desafio da reengenharia de software é, a partir de sistemas existentes, incutir bons métodos de desenvolvimento de software e novas propriedades, gerando um novo sistema de destino que mantém as funcionalidades necessárias com a aplicação de novas tecnologias. Embora os objetivos específicos de um processo de reengenharia sejam determinados pelos objetivos das empresas, há quatro objetivos gerais principais [Sne95]:

• Preparação para o aprimoramento funcional
• Melhorar a capacidade de manutenção
• Migração
• Melhorar a confiabilidade

4.1.2 Abordagens de Reengenharia de Sistemas

A) Big Bang

Apresentada por [Big89] a abordagem “big bang”, substitui todo o sistema de uma só vez. Esta abordagem é frequentemente utilizada por projetos que precisam resolver um problema imediatamente, como a migração para uma arquitetura de sistema diferente.

A vantagem desta abordagem é que o sistema é levado a um novo ambiente de uma só vez. Nenhuma interface entre os componentes antigos e novos deve ser desenvolvida. A desvantagem desta abordagem é que o resultado tende a ser projetos monolíticos que podem não ser sempre adequados. Para grandes sistemas, esta abordagem pode consumir muitos recursos ou exigir grandes quantidades de tempo até que o sistema alvo seja produzido. O risco com esta abordagem é alto, o sistema “novo” tem que ter seu funcionalmente intacto e funcionar em paralelo com o sistema antigo para garantir a funcionalidade. Esta operação em paralelo pode ser difícil e cara de se fazer. Uma das maiores dificuldades é o controle de mudança, entre o tempo que o novo sistema é iniciado e terminado, muitas mudanças poderão ser feitas no sistema antigo, e isto tem que ser refletido no novo sistema [Big89] [BS95].

B) Abordagem incremental

Na abordagem incremental [SB95] [BCMV03], ou iterativa, os módulos do sistema passam pela reengenharia e são adicionados gradualmente à medida que novas versões do sistema são geradas. O projeto é dividido em módulos de reengenharia com base em seções do sistema existente.

As vantagens desta abordagem são que os componentes do sistema são produzidos mais rapidamente e é mais fácil de detectar erros uma vez que os novos componentes estão claramente identificados. Como as versões intermediárias são liberadas, o cliente pode ver o progresso e identificar rapidamente uma funcionalidade perdida. Outra vantagem é que a mudança para o sistema antigo pode ser mais fácil tratar, uma vez que alterações nos componentes que não estão sob a reengenharia não têm impacto sobre o componente atual. Uma desvantagem para a abordagem incremental é que o sistema demora mais tempo para ser concluído com a versões provisórias múltiplas que requerem controle de configuração cuidadoso. Outra desvantagem é que a estrutura inteira do sistema não pode ser alterada, apenas a estrutura dentro dos módulos específicos. Isso requer cuidadosa identificação de componentes no sistema existente e extenso planejamento da estrutura do sistema de destino. Esta abordagem tem um risco menor do que o Big Bang , porque como cada componente passa pela reengenharia, os riscos na parte de código podem ser identificados e monitorados.

C) Abordagem Evolucionária

Na abordagem “evolucionária”, como na abordagem incremental, os módulos do sistema original são substituídos por módulos do sistema que passam por reengenharia. Nesta abordagem, no entanto, os módulos são escolhidos com base na sua funcionalidade, não sobre a estrutura do sistema existente. O sistema de destino é construído usando módulos funcionalmente coesos, conforme necessário. A abordagem evolutiva permite aos desenvolvedores concentrar os esforços de reengenharia na identificação de objetos funcionais, independentemente de onde as tarefas residirem no sistema atual.

4.1.3 Processo de Reengenharia de Sistemas

Diversos autores apresentam seus modelos de processo de reengenharia de sistemas [Pre11] [Som07] [Pfl04] [Ros96] [Sne95]. Tais processos se diferenciam por

denominações das etapas principais e/ou supressão e contração de etapas. Porém, o cerne o processo é o mesmo e pode ser sintetizado nas seguintes etapas:

A) Planejamento: é a etapa que mais impacta no sucesso do projeto de reengenharia [Sne95], devido às decisões cruciais que são tomadas nesta fase. O planejamento inclui análise de viabilidade do projeto (determinar o retorno do investimento, avaliar as necessidades e objetivos que o sistema existente atende, dentre outras), análise do portfólio (priorizar as aplicações a serem submetidas ao processo de reengenharia de acordo com a qualidade técnica e o valor que agregam ao negócio), estimativa de custo/esforço (estimar o esforço e o orçamento que serão empregados para a realização da reengenharia do sistema), análise de custo benefício (comparação das estimativas com o custo esperado) e contratação (estabelecer tarefas e distribuir o esforço de desenvolvimento).

B) Reestruturação dos Documentos: documentação pobre é a marca registrada de muitos sistemas legados [Pre11]. Neste caso, pode aplicar uma das seguintes alternativas, dependendo do contexto da organização:

- Criar documentação consome muito tempo: se o sistema funciona, pode-se optar por conviver com o que se tem. Em alguns casos, essa atitude é correta. Não é possível recriar documentação para centenas de programas de computador. Se um programa for relativamente estático, está chegando ao fim de sua vida útil, e provavelmente não terá uma modificação significativa.
- A documentação tem que ser atualizada, mas a organização apresenta recursos limitados: deve-se empregar uma abordagem "documentar quando usar". Pode não ser necessário redocumentar totalmente o aplicativo. Em vez disso, partes do sistema que estão passando por alteração são completamente documentadas.
- O sistema é critico para o negócio e deve ser totalmente documentado: mesmo neste caso, uma abordagem inteligente é limitar a documentação a um mínimo essencial.

C) Engenharia Reversa: é o processo de análise de um sistema para identificação dos componentes deste sistema e suas inter-relações, com objetivo de criar representações do sistema em outra forma ou em um nível mais alto de abstração. Na engenharia reversa, os requisitos, projeto essencial, a estrutura e o conteúdo do sistema legado devem ser recapturados. Além de capturar relações técnicas e interações, informações sobre a aplicação de regras de negócio e

processos que provaram ser úteis na gestão do negócio também deve ser recuperados. Os principais objetivos da engenharia reversa são: gerar visões alternativas, recuperar informações perdidas, detectar efeitos colaterais e facilitar a reutilização. A eficácia deste processo irá afetar o sucesso do projeto de reengenharia. A engenharia reversa não envolve mudanças no sistema ou a criação de um novo sistema, que é o processo de exame, sem alterar a sua funcionalidade geral.

D) **Reestruturação do Código:** tipo mais comum de reengenharia. Alguns sistemas legados tem uma arquitetura de programa razoavelmente sólida, mas os módulos individuais foram codificados de um modo que se torna difícil de entendê-los, testá-los e mantê-los. Nesses casos, o código dentro dos módulos suspeitos pode ser reestruturado.

   Para executar essa atividade, o código fonte é analisado por meio de uma ferramenta de reestruturação. As violações das construções de programação estruturada são registradas, e o código é então reestruturado ou mesmo reescrito em uma linguagem de programação mais moderna. O código reestruturado resultante é revisado e testado para garantir que não tenham sido introduzidas anomalias. A documentação interna do código é atualizada.

E) **Reestruturação de Dados**: diferentemente da reestruturação de código, que ocorre em um nível relativamente baixo de abstração, a reestruturação de dados é uma atividade de reengenharia em escala completa. Em muitos casos, a reestruturação de dados começa com atividade de engenharia reversa. A arquitetura de dados atual é dissecada, e os modelos de dados necessários são definidos. Identificam-se os objetos de dados e atributos, e as estruturas de dados existentes são revisas quanto à qualidade.

F) **Engenharia Direta:** após o desejado nível de abstração ser atingido (durante a engenharia reversa), a engenharia direta pode começar. Engenharia direta corresponde exatamente ao processo normal de desenvolvimento de software, a partir do nível de abstração alcançado no processo de engenharia reversa. Ou seja, se o software está a sendo redesenhado para atender a uma nova arquitetura global do sistema, o processo de engenharia reversa deve extrair os requisitos de software, e o processo de engenharia direta iria começar com o desenvolvimento de um novo design. Na engenharia direta, qualquer mudança ou aumento de funcionalidade deve ser evitada, uma vez que complica o processo de validação.

Durante esta fase, garantia de qualidade e disciplinas de gerenciamento de configuração devem ser aplicadas. Técnicas de medição também devem ser aplicadas, para avaliar a melhoria do software e para identificar potenciais áreas de risco.

4.1.4 Riscos da Reengenharia de Sistemas

Apesar de reengenharia ser muitas vezes usada como um meio de reduzir os riscos e reduzir os custos de operação e manutenção do software legado, reengenharia não é um processo sem riscos [Ros96]. O planejamento auxilia os gerentes de projeto na identificação precoce de riscos e na preparação para a estimativa e avaliação de riscos de reengenharia, além de fornecer um quadro realista de expectativas. A identificação dos riscos é essencial para a avaliação de risco, análise de risco eficaz e gestão de riscos.

4.2 ESTIMATIVA DE ESFORÇO

A gestão bem sucedida de projetos de software começa com uma estimativa precisa do esforço de desenvolvimento [Pre11] [Som08]. Imprecisão nas estimativas continua a ser um dos fatores-chave que contribuem para falhas de projeto de software [SG12]. Subestimar o esforço a ser gasto gera pressões que comprometem a qualidade do desenvolvimento do projeto, enquanto que, superestimar pode elevar o custo.

Técnicas de estimativa de projeto de software são projetadas para converter uma estimativa de tamanho (como um número estimado de linhas de código-fonte) para uma estimativa de esforço (pessoas por hora, dia, mês ou ano).

A previsão exata tem sido difícil, já que muitos desses relacionamentos não são bem compreendidos. Melhorar as técnicas de estimação disponíveis para gerentes de projeto facilitaria o controle mais eficaz do tempo e os orçamentos no desenvolvimento de software [CW97].

4.2.1 Técnicas de Estimativa de Esforço

Esta pesquisa adota a taxonomia sugerida por [ABC00] por entender que esta, além de ser a primeira proposta de taxonomia na área, é a mais adequada ao estado da arte das estimativas classificando, se não todas, grande parte das técnicas existentes na literatura. Assim, as técnicas de estimativas são classificadas como se segue:

- Baseadas em Modelo;
- Dinâmicas;
- Baseadas em Expertise;
- Baseadas em Regressão;
- Orientadas a Aprendizagem de Máquina;
- Compostas;

4.2.1.1 Baseadas em Modelo

O final dos anos 70 produziu uma grande variedade de modelos robustos de estimativa como o SLIM [PM92], Checkpoint [Jon97], o PRICE-S [Par88], SEER [Jen83] e COCOMO [Boe81]. Muitos deles são modelos proprietários e, portanto, não podem ser comparados e contrastados, em termos de estrutura do modelo.

Teoria ou experimentação determinam a forma funcional destes modelos. Esta seção discute SLIM e COCOMO81, dois dos mais populares modelos conhecidos.

- SLIM

Larry Putnam desenvolveu o modelo de ciclo de vida de software (Software Life-Cycle Model - SLIM) no final de 1970 [PM92]. O SLIM é baseado na análise do ciclo de vida de Putnam em termos de uma chamada distribuição Rayleigh da equipe do projeto *versus* tempo. Ele suporta a maioria dos métodos de estimativa de tamanho populares, incluindo técnicas de estimativa, instruções de código, pontos por função, etc. Ele faz uso de uma curva de Rayleigh para estimar esforço do projeto, cronograma e taxa de defeito. SLIM pode gravar e analisar dados de projetos previamente preenchidos que são então utilizados para calibrar o modelo, ou se os dados não estão disponíveis, um conjunto de perguntas podem ser respondidas para obter estes valores a partir do banco de dados existente.

- COCOMO

O modelo de estimativa de custo e cronograma (COnstructive COst Model - COCOMO) foi originalmente publicado em 1981[Boe81]. Tornou-se um dos mais populares modelos paramétricos de estimativa de custo da década de 1980. Porém, devido à evolução nos processo de desenvolvimento de software, o COCOMO '81 passou a ter dificuldades em estimar os custos de software desenvolvidos para novos processos de ciclo de vida e capacidades. O esforço de pesquisa para o COCOMO II foi

iniciado em 1994 na Universidade do Sul da Califórnia para abordar as questões sobre modelos não-sequenciais e processos rápidos de desenvolvimento, reengenharia, reutilização, abordagens orientadas a objeto, etc. COCOMO II foi publicado em 1995 [Boe95] e tem três submodelos: Composição de Aplicação – mais conveniente para a fase de prototipação em um ciclo de vida espiral - , Design precoce – quando os requisitos são bem conhecidos e a arquitetura do software foi bem explorada - e pós-arquitetura de modelos – onde se trabalha com o real desenvolvimento e manutenção do produto de software.

4.2.1.2 Dinâmicas

Técnicas dinâmicas reconhecem explicitamente que esforço e fatores de custo mudam ao longo do desenvolvimento do sistema, isto é, eles são dinâmicos (não-estáticos) ao longo do tempo. Esta é uma diferença significativa em relação às outras técnicas, que tendem a confiar em modelos estáticos e previsões com base em "quadros" de uma situação de desenvolvimento em um determinado momento no tempo. No entanto, fatores como prazos, níveis de pessoal, requisitos de design, necessidades de treinamento, orçamento, dentre outros, flutuam ao longo do desenvolvimento e causam flutuações correspondentes na produtividade da equipe do projeto. Isto, por sua vez, tem consequências no prazo e orçamento do projeto. As técnicas dinâmicas mais importantes são baseadas na abordagem dinâmica de sistemas cujo modelo foi proposto por Jay Forrester em 1961 [For61].

4.2.1.3 Baseadas em Expertise

Técnicas baseadas *expertise* são úteis na ausência de dados empíricos e quantificados. Elas capturam o conhecimento e a experiência de profissionais dentro de um domínio de interesse, fornecendo estimativas com base em uma síntese dos resultados conhecidos de todos os últimos projetos para que o especialista esta a par ou em que ele ou ela participou. A desvantagem óbvia deste método é que uma estimativa é tão boa quanto o parecer do perito, e não há nenhuma maneira geral para testar essa opinião até que seja tarde demais para corrigir o dano se que a opinião se provar errada. Anos de experiência não necessariamente se traduzem em elevados níveis de competência. Além disso, mesmo a mais competente de pessoas, às vezes, simplesmente estima errado. Duas técnicas foram desenvolvidas que captura parecer dos peritos, mas que também tomar

medidas para mitigar a possibilidade de que o julgamento de qualquer um especialista será desligado, estas técnicas são *Delphi* e *Work Breakdown Structure*.

4.2.1.4 Baseadas em Regressão

Modelos de regressão estatística estimam o esforço de desenvolvimento de software como a variável dependente. Tamanho de software (em métricas como linhas de código ou pontos de função) é usado como uma variável independente. Em alguns modelos, outros parâmetros, como a linguagem de programação de desenvolvimento ou sistema operacional pode ser usado como variáveis independentes adicionais para um modelo de regressão múltipla. Os modelos de regressão tem a vantagem de possuir uma base matemática sólida, bem como medidas de qualidade de ajuste (*goodness fit*), ou seja, quão bem a curva corresponde ao conjunto de dados especificado. Porém, modelos de regressão estatística são muito suscetíveis ao efeito de *outliers* (itens de dados que podem ser completamente fora de sintonia com o resto do conjunto de dados). Além disso, um modelo de regressão também precisa de um conjunto de dados relativamente grande que pode ser um problema no campo de estimativa de software campo [FWD97].

4.2.1.5 Orientadas a Aprendizagem de Máquina

As técnicas para as estimativas de software baseadas em Aprendizado de Máquina (ML–Machine Learning) representam uma alternativa às técnicas tradicionais, as quais são baseadas em regressão, técnicas estatísticas, ou ainda, na expertise humana [Nun10].

O método para a criação de modelos de ML é a exploração dos dados históricos do domínio, feita por algoritmos específicos na tentativa de formular ou inferir um conjunto de regras que permitam a dedução de valores futuros [SF95].

Dentre as abordagens de ML para o desenvolvimento de modelos preditivos de software destacam-se: Redes Neurais Artificiais; Sistemas Fuzzy, Raciocínio Baseado em Casos; Árvores de Classificação e Regressão e Algoritmos Genéticos.

- *Redes Neurais Artificiais*

De acordo com Gray e McDonell [GM97] redes neurais artificiais é a técnica de construção de modelos de estimativa mais comumente utilizada como uma alternativa à regressão de mínimos quadrados. As redes neurais artificiais (RNAs) adotam uma abordagem de aprendizagem derivadas de um modelo preditivo. A rede é projetada para um conjunto específico de entrada, por exemplo, pontos por função, linguagem de programação, etc, bem como a saída (s), por exemplo, o esforço de desenvolvimento. A rede recebe um conjunto de processos conhecidos (o conjunto de treino) que é utilizado para "treinar" a rede, ou seja, determinar os pesos associados a cada entrada na rede. Uma vez que a rede está treinada e estável, o esforço de desenvolvimento de um novo caso pode ser previsto, substituindo os valores de entrada relevantes para o caso específico. RNAs são reconhecidas por sua capacidade de fornecer bons resultados quando se trata de problemas onde existem relações complexas entre entradas e saídas, e onde os dados de entrada são distorcidos por altos níveis de ruído [TB91].

Apesar da robustez da técnica, algumas desvantagens são observadas como a falta de imunidade a problemas comuns em técnicas estatísticas, como existência de *outliers*, valores incompletos ou perdidos. Além disso, tem-se o fato de a rede apresentar um funcionamento dito “caixa-preta”, ou seja, não detalham as informações de processamento até chegar ao resultado final, o que pode ser critico no momento de se realizar uma análise causal, por exemplo [Nun10].

- *Sistemas Fuzzy*

Um sistema *fuzzy* é um mapeamento de valores em termos linguísticos, por exemplo, "muito baixo", "baixo", "alto" e "muito alto" para um conjunto de valores de variáveis correspondentes. Tanto a entrada quanto a saída do sistema fuzzy pode ser numérica ou linguística [GM97].

A principal vantagem da utilização de sistemas *fuzzy* para estimativas de software é a sua fácil compreensão, devido ao uso de termos linguísticos, fazendo com que o mesmo possa ser analisado e criticado por pessoas sem conhecimento ou treinamento prévio. Como desvantagem, a dificuldade de especificação de um sistema que permita uma alta precisão de resultados mantendo uma interface interpretável, onde geralmente, sistemas mais complexos precisam de mais regras, levando a um aumento de complexidade e decréscimo de poder de interpretação [GM97] [KKS94].

- *Raciocínio Baseado em Casos (Estimativa por Analogia)*

O raciocínio baseado em casos (RBC), também conhecido como Estimativa por Analogia (no contexto de estimativa de esforço), [WM94] é uma técnica de resolução de problemas que resolve novos problemas adaptando soluções que foram usadas para resolver problemas antigos. RBC recupera um ou mais casos semelhantes ao problema atual e tenta modificar estes casos para ajustar aos parâmetros do problema atual. Na estimativa de esforço de desenvolvimento de software, cada caso pode ser um desenvolvimento de software anterior, enquanto o problema atual é extrair uma estimativa adequada para o projeto atual. Como resultado, o desenvolvimento deve ser mais rápido e a estimativa de tempo ajustada em conformidade.

A vantagem da utilização dessa abordagem é que pode-se justificar decisões com base em casos anteriores utilizadas na resolução de um problema. Além disso, a abordagem RBC é intuitivamente semelhante ao julgamento de especialistas adotado em muitas organizações que dependem de uso de desenvolvedores experientes para estimar esforço do projeto. Estes indivíduos estimar pela adaptação julgamento de desenvolvimentos de sistemas anteriores.

O problema dessa técnica é que os projetos têm que ter a mesma característica para que as estimativas tenham uma boa proximidade do real, o que nem sempre é possível, principalmente no contexto de reengenharia de sistemas.

- *Árvores de Decisão e Regressão*

Árvores de decisão e de regressão são conceitos similares, mas diferem na representação do atributo que se deseja prever. Ambas se utilizam de um conjunto de dados previamente conhecidos, induzindo as regras necessárias para classificarem esses dados. O método mais comumente utilizado para construção da árvore é o *top-down*. Essa estratégia consiste da análise de qual atributo dos registros de dados melhor divide o conjunto de dados em subpopulações disjuntas. Dentre as abordagens utilizadas para a realização dessa divisão estão: o cálculo do erro médio quadrado, o cálculo de entropia, dentre outros [SF95] [GM97].

As árvores de regressão são mais utilizadas em métricas de software do que as de decisão. Isso é dado pela própria necessidade que o domínio impõe de obtenção de resultados numéricos. Além de serem utilizadas para estimar esforço, as árvores de regressão também são utilizadas para estimarem defeitos [JKA00] [GM97].

A vantagem em se utilizar as árvores de regressão é que elas são simples de serem aplicadas sobre um conjunto de dados, já que existem várias ferramentas que as implementam, como o WEKA por exemplo [UW13]. Como desvantagens tem-se a dificuldade de compreensão, pois quanto mais níveis e nodos a árvore tiver, menos compreensível ela se torna. Além disso, por depender de dados do projeto, a árvore é sensível à *outliers*, isso quer dizer que a qualidade dos dados é importante para que se obtenham bons resultados com essa técnica.

- *Algoritmos Genéticos*

Algoritmo Genético (GA) é um algoritmo usado para pesquisar a melhor solução de forma aleatória. Ela imita os genes biológicos. Após a avaliação, ele terá a melhor condição. GA mostram os genes por estado de cromossomo. Após a competição, os mais aptos são as melhores soluções. Os cromossomos são gerados por números aleatórios. Seu valor de fitness é dado pela função alvo. De acordo com o valor de fitness, o cromossomo será cruzado (crossover), sofrerá mutação, e será selecionado. O crossover vai escolher aleatoriamente dois cromossomos, chamados de pais, e trocar genes entre si. Em seguida, ele irá gerar dois novos cromossomos, chamados filhos. Mutação significa escolher um cromossomo aleatoriamente, e torná-lo um pouco diferente. Por cruzamento e mutação, são produzidos mais cromossomos, e pode-se obter o valor de aptidão de cada cromossomo. Então é aplicado um método de seleção. Os cromossomos selecionados serão usados na próxima avaliação.

Imitando seleção e reprodução biológica, GA pode eficientemente pesquisar através do espaço de soluções de problemas complexos e oferece oportunidade para escapar do ótimo local. Com isso, GA tornou-se um dos algoritmos mais populares para os problemas de otimização [LCH11].

4.2.1.6 Compostas

Por fim, [ABC00] afirma que nenhuma técnica em especial é adequada para todas as situações, portanto, na maioria das vezes faz-se uso de uma combinação de técnicas para se obter melhor acurácia nos resultados. Com base nisso, diversas soluções emergiram já propondo composição de técnicas de maneira a tornar a solução proposta mais generalizável aos diversos contextos de desenvolvimento. Assim, técnicas

compostas nada mais são do que modelos, metodologias, métodos, dentre outros que combinam duas ou mais técnicas para realização de estimativas de esforço.

4.3 ESTIMATIVA DE ESFORÇO EM REENGENHARIA DE SISTEMAS

Estimativa precisa dos custos do projeto é um pré-requisito essencial para fazer um projeto de reengenharia [Sne95]. Os sistemas existentes geralmente passam por reengenharia por ser mais barato do que a manutenção ou substituição dos mesmos [Pre11] [Som07]. No entanto, para tomar esta decisão, a organização deve saber o quanto a reengenharia vai custar e quando esse investimento será visível.

No inicio da década de 90 surgiram os primeiros estudos sobre economia de projetos de reengenharia de software [Sne90], desde então, um grande número de pesquisas tem sido feito e uma vasta gama de experiências práticas recolhidas. Antigamente, o esforço em projetos de reengenharia era calculado apenas com base no tamanho do sistema, sem levar em conta a complexidade e qualidade. Com o crescimento e amadurecimento da área tornou-se importante levar em conta os fatores de complexidade e qualidade nos custos de reengenharia de software. Reengenharia é uma alternativa para manutenção, compra de um pacote padrão ou a não fazer nada. Portanto, é importante para a gestão da organização saber qual o retorno sobre o investimento para cada alternativa.

No contexto de soluções para apoio da realização de estimativas de esforço em projetos de reengenharia, Sneed [Sne05] propõe uma ferramenta para o cálculo do tempo e os custos necessários para reestruturar um sistema existente. O processo é derivado da experiência do autor de 20 anos em estimar projetos de reengenharia e foi validado por vários experimentos de campo em que foi refinado e calibrado.

Além disso, [CLV01] [BCV03] e [BBCV06] apresentam um método para a estimativa de esforço dinâmico, a aplicação desse método em um projeto de software real e uma ferramenta de apoio, respectivamente. A ideia principal é permitir a mudança do modelo de estimativa em tempo de execução, de acordo com as melhorias de processos operados. O nível de granularidade proposto permite que os gerentes de projeto realizem a variação do processo e sua tendência a maturidade. O estudo empírico em um projeto real de reengenharia foi aplicado com objetivo de verificar se o nível de granularidade e calibração do modelo contínuo são eficazes para a precisão da estimativa.

5 METODOLOGIA

Com base no que foi exposto nas seções anteriores, tem-se que a finalidade deste trabalho é aplicada ou tecnológica, pois se preocupa em gerar um modelo para apoio a realização de estimativas de esforço em projetos de reengenharia de sistemas.

Quanto aos objetivos desse trabalho, ele foi exploratório e documental nas fases inicias, considerando que visou o levantamento e a compreensão do estado da arte de estimativa de esforço em projetos de software com ênfase em projetos de reengenharia, a partir da realização de uma revisão sistemática da literatura. Além disso, também buscou-se levantar os principais conceitos relacionados ao processo de reengenharia em si, a partir da exploração e leitura de livros e artigos relacionados ao tema.

Posteriormente, o objetivo tornara-se descritivo, pois baseado no referencial teórico e nas práticas de estimativa de esforço levantadas na indústria, será proposto um modelo para realização de estimativas de esforço em projetos de reengenharia de sistemas. Por fim, uma etapa experimental será inserida, onde o modelo proposto será validado em relação a outros modelos de estimativa de esforço, utilizando-se para isto uma base real de projetos.

Em relação ao procedimento, a pesquisa deu-se a partir da realização de uma revisão sistemática e uma revisão bibliográfica para coletar o estado da arte de estimativas de esforço e de reengenharia de sistemas, respectivamente.

Em seguida, será realizado um estudo empírico no qual serão aplicadas entrevistas semiestruturadas, junto a gerentes de projetos de desenvolvimento de software atuantes na indústria, com objetivo de coletar as práticas utilizadas para realização de estimativa de esforço em projetos de reengenharia de sistemas. Os resultados obtidos a partir da análise dos dados coletados no estudo e na pesquisa bibliográfica servirão como base para formulação da proposta do modelo de estimativas. Por fim, será realizado um experimento com dados reais de projetos de reengenharia para verificação e análise do modelo proposto.

A Figura 1 apresenta o desenho da pesquisa.

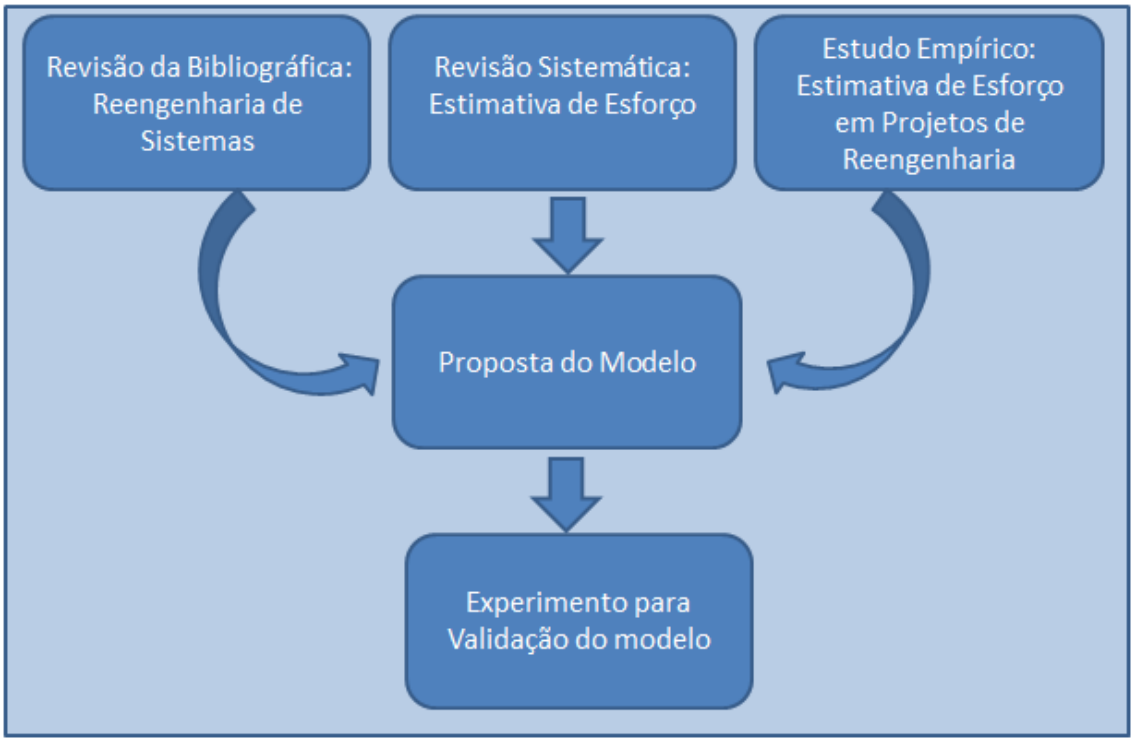

Figura 1 - Desenho da Pesquisa

6 ANDAMENTO DA PESQUISA

A pesquisa proposta neste trabalho vem sendo desenvolvida desde Junho de 2013, e começou com a revisão da literatura sobre Reengenharia de Sistemas e sobre Estimativa de Esforço. No que diz respeito à reengenharia de sistemas, foram consultados livros, periódicos, anais de conferências e sites de empresas multinacionais que realizam este processo, para ampliação do conhecimento sobre o tema.

Em relação à Estimativa de Esforço, inicialmente foi realizada uma revisão da literatura *ad-hoc*, porém, percebeu-se que este é um tema amplamente discutido desde a década de 80 e possuía uma gama de soluções propostas. Logo, optou-se por realizar uma Revisão Sistemática da Literatura, de acordo com o processo proposto por [Kit04] para se obter um entendimento mais aprofundado sobre o tema e o estado da arte em soluções de estimativa de esforço. Esta revisão serviu também para confirmar a escassez de trabalhos sobre estimativa de esforço direcionados para o contexto de reengenharia de sistemas.

O objetivo principal da revisão sistemática era, como mencionado acima, realizar um levantamento sobre o estado da arte de soluções de estimativa de esforço em projetos de reengenharia de sistemas. Porém, dado os poucos resultados retornados, optou-se por classificar todas as soluções de estimativa de esforço em projetos de desenvolvimento de software em geral, pois sabendo-se o estado da arte no geral, pode-se aproveitar este conhecimento na elaboração do modelo proposto nesta pesquisa, conforme foi indicado no Capítulo 5, sobre a Metodologia.

A revisão está atualmente em execução, na fase de extração de dados, onde os artigos selecionados pela varredura de título e abstract estão sendo lidos na integra e os dados de interesse estão sendo extraídos.

Além disso, está em andamento o planejamento do estudo empírico, mais precisamente a elaboração do guia a ser utilizado nas entrevistas.

A previsão para finalização da revisão e da elaboração do guia é dezembro de 2013.

7 ATIVIDADES E CRONOGRAMA

7.1 ATIVIDADES

As atividades propostas para o segundo ano do mestrado são:

1. Planejamento do Estudo de Campo: validação e avaliação do guia de entrevistas, seleção das organizações e dos profissionais, agendamento das reuniões.
2. Aplicação do Estudo de Campo: aplicação de entrevistas semiestruturadas junto a gerentes e demais pessoas envolvidas no projeto de reengenharia, responsáveis por estimar esforço.
3. Análise dos Dados do Estudo de Campo: transcrição e análise dos dados obtidos a partir das entrevistas.
4. Proposta do Modelo de Estimativa de Esforço para Projetos de Reengenharia de Sistemas: com base na revisão da literatura sobre reengenharia de projetos, na revisão sistemática sobre estimativas de esforço e no estudo empírico aplicado nas empresas, propor o modelo para estimar esforço.
5. Validação e Análise do Modelo: realização de experimentos utilizando dados reais de projetos de reengenharia para comprovar a acurácia do modelo em relação a outras propostas apresentadas na literatura e utilizadas nas organizações visitadas.
6. Escrita de Artigo sobre a Revisão Sistemática.
7. Escrita da Monografia para submissão como Produção Textual junto ao PPGCC
8. Seminário de Andamento.
9. Escrita e Entrega do Volume da Dissertação.
10. Defesa da Dissertação.

7.2 CRONOGRAMA

O apresenta Quadro 1 o cronograma de atividades proposto para a pesquisa:

	Mês (2014-2015)													
Atividade	**Jan**	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Out**	**Nov**	**Dez**	**Jan**	**Fev**
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														

Quadro 1 - Cronograma para o Período de Janeiro de 2014 à Fevereiro de 2015

8 REFERENCIAS BIBLIOGRÁFICAS

[ABC00] Chris Abts, Barry Boehm e Sunita Chulani, *"Software Development Cost Estimation Approaches – A Survey"*, *Annuals Software Engineering*, vol.10 (1-4), October 2000, p. 177–205.

[ANC04] Mohammad Azzeh, Daniel Neagu e Peter Cowling, *"Software Effort Estimation Based on Weighted Fuzzy Grey Relational Analysis"*. In: Proceedings of the 5th International Conference on Predictor Models in Software Engineering. ACM, 2009. p. 8.

[AO12] Iman Attarzadeh e Siew Hock Ow. *"Proposing a Novel Artificial Neural Network Prediction Model to Improve the Precision".* In: BIONETICS 2010, LNICST 87, pp. 334–342, 2012.

[BBCV06] Maria Teresa Baldassarre *et al. "Speed: Software project effort evaluator based on dynamic-calibration"*. In: Software Maintenance, 2006. ICSM'06. 22nd IEEE International Conference on. IEEE, 2006. p. 272-273

[BBR12] Márcio Basgalupp *et al. "Predicting Software Maintenance Effort through Evolutionary-based Decision Trees".* In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. ACM, 2012. p. 1209-1214.

[BBS13] Márcio Basgalupp *et al. "Software effort prediction: a hyper-heuristic decision-tree based approach".* In: Proceedings of the 28th Annual ACM Symposium on Applied Computing. ACM, 2013. p. 1109-1116.

[BCMV03] Alessandro Bianchi *et al. "Iterative reengineering of legacy systems*". IEEE Trans. Software Eng., 29(3):225{241, 2003.

[BCV03] Maria Teresa Baldassarre *et al*. *"Software renewal projects estimation using dynamic calibration"*. In: Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on. IEEE, 2003. p. 105-115.

[BDSM12] Tirimula Benala. *"Genetic Algorithm for Optimizing Functional Link Artificial Neural Network Based Software Cost".* In: Proceedings of the InConINDIA 2012, AISC 132, pp. 75–82.

[Big89] Ted Biggerstaff. *"Design recovery for maintenance and reuse".* Computer, 22(7):36–49, 1989.

[BMDP12] Tirimula Benala *et al. "Software Effort Prediction Using Fuzzy Clustering and Functional Link Artificial Neural Networks".* In: Swarm, Evolutionary, and Memetic Computing. Springer Berlin Heidelberg, 2012. p. 124-132.

[Boe81] Barry Boehm "*Software Engineering Economics"*, Prentice Hall, 1981.

[Boe95] Barry Boehm *et al. "Cost Models for Future Software Life-cycle Processes: COCOMO 2.0,"*.In: Annals of Software Engineering Special Volume on Software

Process and Product Measurement, J.D. Arthur and S.M. Henry (Eds.), J.C. Baltzer AG, Science Publishers, Amsterdam, The Netherlands, Vol 1, 1995, pp. 45 - 60.

[BSTWW05] John Bergey *et al.* *"Why Reengineering Projects Fail*". Carnegie-Mellon univ pittsburgh pa software engineering inst, 2005

[BS95] Michael Brodie e Michael Stonebraker. "*Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach"*. Morgan Kaufmann Pub; 1995, 1 edition.

[CC90] Elliot Chikofsky and James Cross. *"Reverse engineering and design recovery: A taxonomy"*, Software, IEEE 7.1, 1990, 13-17.

[CLV01] Danilo Caivano *et al.* *"Software renewal process comprehension using dynamic effort estimation"*. In: Software Maintenance, 2001. Proceedings. IEEE International Conference on. IEEE, 2001. p. 209-218.

[For61] Jay Forrester "*Industrial Dynamics",* Forrester, J., MIT Press, Cambridge, MA, 1961.

[FWD97] Gavin Finnie *et al.* *"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, n. 3, p. 281-289, 1997.

[GM97] Andrew Gray e Stephen MacDonell. *"A comparison of techniques for developing predictive models of software metrics", Information and Software Technology*, vol. 39-6, November 1996, 425–437.

[Jen83] Jensen R. *"An Improved Macrolevel Software Development Resource Estimation Model"*. In: Proceedings 5th ISPA Conference, April 1983, pp. 88-92.

[JKA00] Wendell Jones, Taghi Khoshgoftaar, Edward Allen, e John Hudepohl. *"Classification tree models of software-quality over multiple releases"*, IEEE Transactions on Reliabiilty, vol. 49-1, March 2000, 4–11.

[Jor05] Magne Jørgensen *"Evidence-based guidelines for assessment of software development cost uncertainty"*. In: IEEE Transactions, v. 31, n. 11, 2005, p. 942-954.

[Jor07] Magne Jørgensen, *"Forecasting of software development work effort: Evidence on expert judgement and formal models"*. In: International Journal of Forecasting, vol. 23, 2007, pp. 449-462.

[Jon97] James Jones. "*Applied Software Measurement*", 1997, McGraw Hill.

[Kit04] Barbara Kitchenham, *"Procedures for performing systematic reviews".*Keele, UK, Keele University, v. 33, 2004.

[KJHK11] Vahid Khatibi *et al.* *"A New Fuzzy Clustering Based Method to Increase the Accuracy of Software Development Effort Estimation Department of Software Engineering , Faculty of Computer Science and Information System"*. In: World Applied Sciences Journal 14 (9): 1265-1275, 2011

[KKS94] Ananda Krishna, Satish Kumar, and Prem Satsangi. *"Fuzzy systems and neural networks in software engineering project management", Journal of Applied Intelligence*, vol. 4-1, May 1993, 31–52.

[KTB08] Yigit Kultur. *"ENNA: Software Effort Estimation Using Ensemble of Neural Networks with Associative Memory".* In: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering. ACM, 2008. p. 330-338.

[LCH11] Jin-Cherng Lin *et al. "Research on Software Effort Estimation Combined with Genetic Algorithm and Support Vector Regression".* In: Computer Science and Society (ISCCS), 2011 International Symposium on. IEEE, 2011. p. 349-352.

[LC11] Jin-Cherng Lin e Chu-Ting Chang. *"Genetic Algorithm and Support Vector Regression for Software Effort Estimation"*. In: Advanced Materials Research, v. 282, p. 748-752, 2011.

[MC00] Emilia Mendes *et al. "Web development effort estimation using analogy".* In: Software Engineering Conference, 2000. Proceedings. 2000 Australian. IEEE, 2000. p. 203-212.

[NSB08] Vu Nguyen, Bert Steece, e Barry Boehm. *"A constrained regression technique for cocomo calibration"*. In: *ESEM '08: Second ACM-IEEE international symposium on Empirical Software Engineering and Measurement*, 2008, 213–222.

[Nun10] Nelson Nunes *"Modelo-E10: um modelo para estimativas de esforço em manutenção de software"*. Pontifícia Universidade Católica do Rio Grande do Sul, PPGCC, Tese de Doutorado, 2010.

[Par88] Park *"The Central Equations of the PRICE Software Cost Model"*, In: 4th COCOMO Users' Group Meeting, November 1988.

[Pfl11] Shari Pfleeger. *"Engenharia de Software Teoria e Prática".* São Paulo: Prentice Hall, 2004, 2ª edição, 535p.

[PM92] Lawrence Putnam e Ware Myers *"Measures for excellence: reliable software on time, within budget"*. Prentice Hall Professional Technical Reference, 1991.

[Pre11] Roger Pressman. *"Software Engineering: A Practitioner's Approach".* Boston: McGraw-Hill, 2011, 7th edition, 889p.

[RH96] Linda Rosenberg e Lawrence Hyatt. *"Software re-engineering".*Software Assurance Technology Center, p. 2-3, 1996.

[RTR08] Marcelo Blois Ribeiro, Nelson N. Tenorio Jr., and Duncan D. Ruiz. *"A quasi-experiment for Effort and Defect Estimation using Least Square Linear Regression and Function Points"*. In: *Software Engineering Workshop, Annual IEEE/NASA Goddard, 2008, 143–151.*

[SG12] The Standish Group. "*Chaos Report*". Boston. 2012.

[SMAKT11] Mohd Sadiq *et al.* "*Prediction of Software Project Effort Using Fuzzy Logic".* In: Electronics Computer Technology (ICECT), 2011 3rd International Conference on. Vol. 4. IEEE, 2011.

[SF95] Krishnamoorthy Srinivasan and Douglas Fisher. *"Machine learning approaches to estimating software development effort"*, IEEE Transaction Software Engineering, vol.21-2, February 1995, 126–137.

[SMLE02] Y Shan. *"Software Project Effort Estimation Using Genetic Programming".* In: Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference on. IEEE, 2002. p. 1108-1112.

[SM08] Ruchi Shukla. *"Estimating Software Maintenance Effort - A Neural Network Approach".* In:Proceedings of the 1st India software engineering conference. ACM, 2008. p. 107-112.

[Sne05] Harry Sneed. *"Estimating the costs of a reengineering Project*". In WCRE, pages 111{119. IEEE Computer Society, 2005.

[Sne90] Harry Sneed, *"Economics of software re-engineering",* Journal of Software Maintenance: Research and Practice, v. 3, n. 3, 1991, p. 163-182.

[Sne95] Harry Sneed. *"Planning the reengineering of legacy systems".* Software, IEEE, v. 12, n. 1,1995, p. 24-34.

[Som07] Ian Sommerville. *"Engenharia de Software".* Pearson Addison-Wesley, 2007, 8ª edição.

[SSK96] Martin Shepperd *et al*. *"Effort Estimation Using Analogy".* In: Proceedings of the 18th international conference on Software engineering. IEEE Computer Society, 1996. p. 170-178.

[SS12] Urvashi Saxena e S. Singh *"Software Effort Estimation Using Neuro-Fuzzy Approach".* In: Software Engineering (CONSEG), 2012 CSI Sixth International Conference on. IEEE, 2012. p. 1-6.

[SYB08] Yeong-Seok Seo, Kyung-A Yoon, and Doo-Hwan Bae. "An empirical analysis of software effort estimation with outlier elimination". In: *PROMISE '08: 4th International Workshop on Predictor Models in Software Engineering*, 2008, 25–32.

[TB91] Duarte Tregueiros e Robert Berry. *"The application of neural network based methods to the extraction of knowledge from accounting reports".* In: System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on. IEEE, 1991. p. 136-146.

[TMKK12] Masateru Tsunoda *et al. "Incorporating Expert Judgment into Regression Models of Software Effort Estimation".* In: 19th Asia-Pacific Software Engineering Conference, 2008, 374-379.

[UW01] UW The University of Waikato. Weka Machine Learning Project. http://www.cs.waikato.ac.nz/ ml/weka/, 2013. Último acesso: 11/12/2013.

[Vis01] Giuseppe Visaggio. *"Ageing of a data-intensive legacy system: symptoms and remedies".* Journal of Software Maintenance and Evolution: Research and Practice, v. 13, n. 5, p. 281-308, 2001.

[WLT09] Jianfeng Wen *et al. "Improve Analogy-Based Software Effort Estimation Using Principal Components Analysis and Correlation Weighting".* In: Software Engineering Conference, 2009. APSEC'09. Asia-Pacific. IEEE, 2009. p. 179-186.

[WM94] Ian Watson e Farhi Marir. *"Case-based reasoning: A review",* In: Knowledge Engineering Review, v. 9, n. 4, p. 327-354, 1994.

[ZKKN13] Ziauddin *et al. "A Fuzzy Logic Based Software Cost Estimation Model".* In: International Journal of Software Engineering and Its Applications Vol. 7, No. 2, March, 2013