Resumo

As atividades de Teste de Software são reconhecidas por sua importância e elevado custo. Entre elas, a geração manual de casos de teste a partir de artefatos do produto, como casos de uso e protótipos, para a realização de testes de sistema é destacada neste trabalho. Através da aplicação de Redes Neurais Artificiais, é possível estimar a quantidade de casos de teste que serão gerados para determinado caso de uso com base nos atributos do próprio caso de uso, do protótipo e do histórico de projetos de teste. Esta informação pode ajudar no planejamento e na precificação das atividades de teste, e no gerenciamento da equipe de teste. Um estudo de caso foi realizado com o auxílio da ferramenta WEKA e seus resultados mostraram que esta abordagem quantitativa tem potencial para auxiliar nos pontos propostos, embora tenha sido identificada a necessidade de ampliar o conjunto de treinamento para aumentar a precisão da resposta da rede.

Abstract

The Software Testing activities are recognized for their importance and high cost. Among them, the manual generation of test cases from artifacts of the product, like use cases and prototypes, for system testing is highlighted in this work. Through the application of Artificial Neural Networks, it is possible to estimate the amount of test cases that will be generated for a particular use case based on the attributes of the use case, the prototype and past test projects. This information can help in planning and pricing the test activities and managing the test team. A case study was conducted with the aid of the WEKA tool and its results showed that this quantitative approach has potential to assist in the points proposed, although it was identified the need to extend the training set to increase the precision of the response of the network.

Sumário

Tabela de Símbolos e Siglas

RNA – Rede Neural Artificial

CT – Caso de Teste

UC – *Use Case*

MCP – McCulloch e Pitts

RBFN – *Radial Basis Function Network*

MLP – *Multilayer Perceptron*

MMRE – *Mean Magnitude of Relative Error*

COCOMO – *COnstructive COst MOdel*

XP – *eXtreme Programming*

DDP – *Defect Detection Percentange*

CRUD – *Create, Read, Update and Delete*

HTML – *HyperText Markup Language*

WEKA – *Waikato Environment for Knowlegde Analysis*

GPL – *General Public License*

ARFF – *Attribute-Relation File Format*

CSV – *Comma-Separated Value*

Agradecimentos

Minha gratidão a Deus, meu suporte inabalável, e à minha família que sempre torce por mim e se orgulha de minhas conquistas. Em especial, aos meus pais, Antonio Reginaldo Farias de Melo e Maria de Fátima Lima Lira Farias de Melo, pelo amor e carinho que sempre me deram e pelos esforços que me permitiram chegar até aqui.

Muito obrigada às minhas avós, Regina Farias de Melo e Maria José de Lima Lira, por suas orações e desejos de sucesso. Ao meu noivo José Gilberto de França Matos que acompanhou todo meu trabalho de perto, sempre oferecendo ajuda, alegria, amor e paciência incondicionais.

Meus agradecimentos também aos docentes do Departamento de Sistemas e Computação, pelos quais tenho muita admiração, em especial ao meu orientador Prof. Dr. Adriano Lorena Inácio de Oliveira. Aos meus colegas de graduação e aos meus colegas de trabalho da Pitang Consultoria e Sistemas S/A, com os quais sempre pude contar.

É uma honra tê-los ao meu lado e um prazer dividir esta vitória com todos vocês.

Capítulo 1

Introdução

Teste de Software tem diferentes propósitos, entre eles: encontrar falhas, medir qualidade, promover confiança (se nenhuma ou pouca falha é encontrada, a confiança no produto aumenta) e analisar um programa e sua documentação para prevenir falhas [20].

A ausência de falhas não pode ser provada através dos testes, pois seria necessário executar o programa de maneira exaustiva, cobrindo todos os possíveis cenários e todas as possíveis combinações de valores [20]. De fato, o volume de testes pode tornar-se um problema [1], pois a atividade geralmente possui restrições de tempo e orçamento.

Na literatura, há trabalhos que se concentram no planejamento geral das atividades de teste [4][28][33] e outros na estimativa do esforço para realizá-las [2][11], alguns usando Redes Neurais Artificiais (RNAs) [1][19][34]. Além do esforço, outros pontos também são estudados, como:

- Análise da cobertura proporcionada por um conjunto de casos de teste (CTs);
- Geração automática de CTs;
- Geração automática de bases ou oráculos;
- Diminuição do conjunto de CTs através da eliminação daqueles que mais provavelmente não revelariam falhas, etc.

No contexto ilustrado na Figura 1, testes de sistema serão projetados manualmente através de artefatos como casos de uso (UCs, do inglês *Use Cases*) e protótipos. Esses testes buscam falhas no sistema como um todo, num ambiente o mais próximo possível daquele no qual ele será realmente operado. Este trabalho propõe, dentro deste contexto, prever a quantidade de CTs que serão projetados pela equipe de teste usando RNAs, de acordo com o esquema da Figura 2.

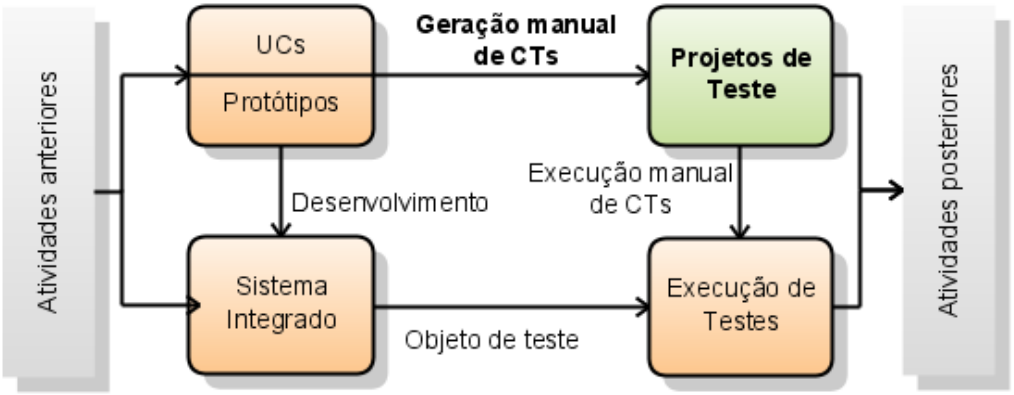

Figura 1. Contexto do estudo: Para cada UC do Sistema Integrado, CTs serão gerados manualmente para compor o Projetos de Teste.

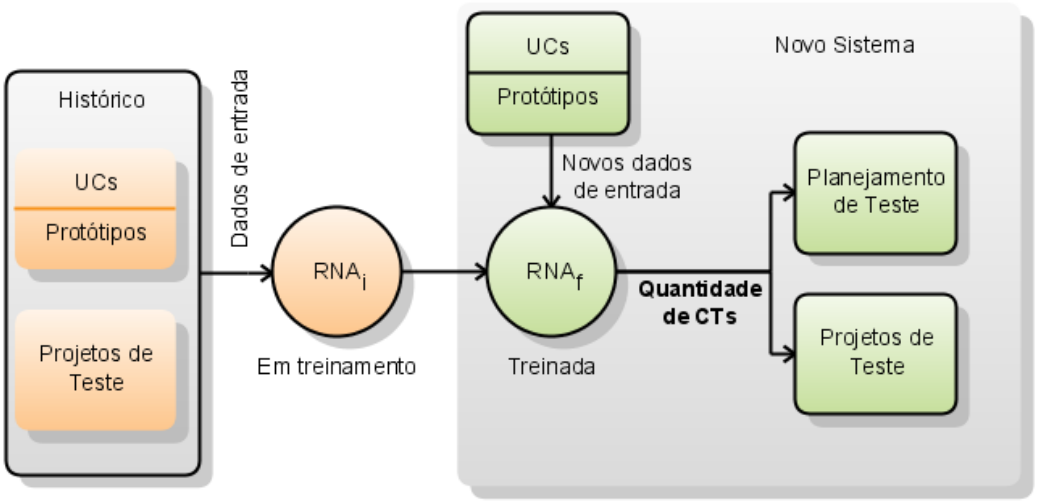

Figura 2. Proposta do estudo: A RNA_i será treinada com base no Histórico de sistemas desenvolvidos e será obtida a RNA_f capaz de prever a Quantidade de CTs de um Novo Sistema.

Na Figura 2, a RNA_i (inicial) representa uma rede em treinamento que recebe dados de entrada do histórico de artefatos como UCs, protótipos e projetos de teste. Ao final do treinamento, temos a RNA_f (final). Quando um novo sistema está em desenvolvimento, já é possível prever a quantidade de CTs que serão projetados usando a RNA_f (Vide Seção 2.1).

1.1 Motivação

As principais motivações para estimar a quantidade de CTs antes do início das atividades de teste são:

- Conhecer a quantidade de CTs de antemão pode ajudar no planejamento;
- Indiretamente, o esforço pode ser medido se o número de CTs a serem projetados é conhecido, através da Equação 1, onde $Esforço$ é a estimativa de esforço, CTs é a quantidade de CTs estimados pela RNA e t_m é o tempo médio para projetar testes por CT:

$$Esforço = CTs \times t_m \quad (1);$$

- Se o custo for calculado com base nesse valor, ele também pode ser indiretamente estimado, através da Equação 2, onde $Custo$ é a estimativa de custo, CTs é a quantidade de CTs estimados pela RNA e $\mathrm{Pr}eço$ é o valor a ser cobrado por CT:

$$Custo = CTs \times \mathrm{Pr}eço \quad (2);$$

- Além disso, como os testes são projetados manualmente, é possível conhecer o perfil ou as características dos projetos de teste de uma equipe (ou equipes), pois geralmente, há um padrão na maneira como os testes são projetados, em especial, quanto a sua granularidade (Vide Subseção 4.1.3). Se um novo integrante é adicionado à equipe, a estimativa poderá servir como uma referência sobre quantos CTs um integrante veterano provavelmente projetaria.

1.2 Objetivo

O objetivo principal deste trabalho é estimar quantos CTs são gerados a partir de artefatos como UCs e protótipos, usando RNAs.

Outros objetivos são:

- Determinar quais benefícios a informação quantitativa sobre os CTs pode proporcionar, respondendo a perguntas como:
  - Qual o impacto dessa informação para o gerente de testes?
  - É possível determinar a granularidade dos testes com base nesse valor?
- Relacionar esta abordagem quantitativa com as estimativas de esforço e cobertura propostas na literatura, no sentido de identificar se podem contribuir entre si.

1.3 Descrição dos Capítulos

A seguir, uma breve descrição dos capítulos que compõem este trabalho:

- Capítulo 1
  Introdução: Apresenta o contexto no qual a proposta deste trabalho foi desenvolvida, suas motivações e seus objetivos.
- Capítulo 2
  Redes Neurais Artificiais Aplicadas a Estimativas: Explica o embasamento teórico do trabalho com foco em Redes Neurais Artificiais.
- Capítulo 3
  Teste de Software: Explica o embasamento teórico do trabalho com foco em Teste de Software.
- Capítulo 4
  Estudo de Caso: Descreve o estudo prático desenvolvido e analisa seus resultados.
- Capítulo 5
  Conclusões: Conclui o trabalho com as considerações finais, as contribuições e os trabalhos futuros.

Capítulo 2

Redes Neurais Artificiais Aplicadas a Estimativas

Neste capítulo, há uma breve explicação sobre os conceitos básicos de Redes Neurais Artificiais e sua aplicação em estimativas no contexto de Engenharia de Software e Engenharia de Teste.

2.1 Redes Neurais Artificiais

Redes Neurais Artificiais (RNAs) inspiram-se na estrutura e funcionamento do cérebro [15]. Elas são compostas por unidades de processamento simples (neurônios artificiais) dispostas em uma ou mais camadas e interligadas por conexões geralmente associadas a pesos que armazenam o conhecimento adquirido e ponderam as entradas recebidas pelos neurônios [6].

O primeiro modelo artificial de um neurônio biológico foi proposto por McCulloch e Pitts [21], por isso, ele é denominado neurônio MCP (McCulloch e Pitts) e é ilustrado na Figura 3.

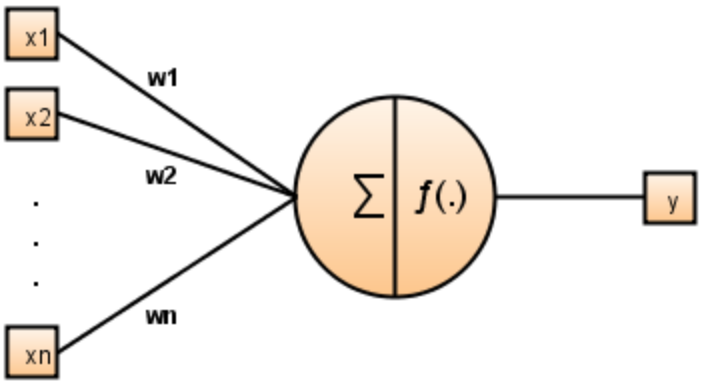

Figura 3. Neurônio MCP, onde: x1 a xn são as entradas, w1 a wn são os pesos, $\sum$ é a soma ponderada entre as entradas e os pesos e $f(.)$ é a função de ativação.

Ele é uma simplificação matemática com n terminais de entrada (representando os dendritos) e apenas um terminal de saída (representando um axônio). Cada terminal de entrada possui um peso associado que determina a influência do sinal daquela conexão. Cada neurônio possui uma função de ativação com um limiar de excitação (*threshold*) que determina se a saída será ativada.

Funções de ativação oferecem diferentes possibilidades de valores de saída. A do neurônio MCP é do tipo degrau deslocada do limiar de ativação θ, como mostra a Equação 3. Dependendo do tipo de problema a ser abordado, outras funções de ativação podem ser usadas, como a sigmoidal, a linear e a gaussiana, mostradas nas Equações 4, 5 e 6, respectivamente:

$$f(u) = \begin{cases} 1, \sum_{i=1}^{n} x_i w_i \geq \theta \\ 0, \sum_{i=1}^{n} x_i w_i < \theta \end{cases} \quad (3),$$

$$f(u) = \frac{1}{1 + e^{-\beta u}} \quad (4),$$

$$f(u) = u \quad (5) \text{ e}$$

$$f(u) = e \frac{-(u-\mu)^2}{r^2} \quad (6).$$

Individualmente, um neurônio possui pouca capacidade de processamento, mas ao se conectar a outros neurônios, forma uma estrutura capaz de resolver problemas complexos [15]. Há muitas opções de conexões entre neurônios, uma das mais simples é a *feedforward* de uma única camada, como ilustra a Figura 4. Da mesma forma que as funções de ativação, a escolha da estrutura de uma RNA dependerá do problema ao qual ela será aplicada.

Os pesos de uma RNA armazenam o conhecimento da rede ao passarem por um processo de treinamento, onde ocorre o aprendizado. No treinamento, o valor desses pesos é ajustado de forma que a resposta da rede seja precisa o suficiente para resolver o problema. Assim, a rede é capaz de generalizar o aprendizado e dar respostas coerentes para dados que não faziam parte do conjunto de treinamento.

No geral, há três tipos de aprendizado: supervisionado, por reforço e não supervisionado [6].

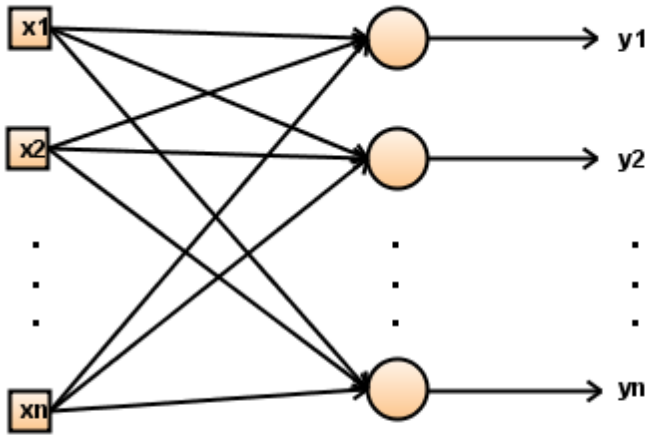

Figura 4. Rede *feedforward* de uma única camada, onde: x1 a xn são as entradas e y1 a yn são as saídas. Embora existam pesos wm nas conexões entre as entradas e os neurônios, eles não são exibidos.

No supervisionado, existem pares de entradas e saídas no domínio do problema que servirão de exemplo para que os pesos sejam ajustados. Para cada conjunto de dados de entrada, a rede produz uma resposta que é comparada com o conjunto de dados de saída. O erro é calculado e com base nele, o peso é ajustado. Dois dos algoritmos de aprendizado supervisionado mais conhecidos são a regra delta [36] o *back-propagation* [31]. Um esquema genérico do aprendizado supervisionado é apresentado na Figura 5.

O aprendizado por reforço é similar ao supervisionado, mas ao invés de um supervisor que provê os exemplos, há um crítico que maximiza o reforço por determinado comportamento da rede que seja considerado bom. Um esquema genérico do aprendizado por reforço é apresentado na Figura 6.

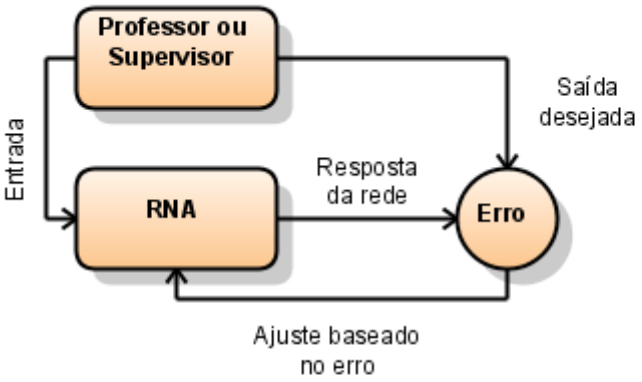

Figura 5. Aprendizado supervisionado.

No aprendizado não supervisionado não há supervisor ou crítico, apenas os dados de entrada que são interpretados pela rede. Esse tipo de aprendizado é adotado quando não há exemplos com os quais treinar a rede, apenas os estados do meio externo que servem de entradas. Um esquema genérico do aprendizado por reforço é apresentado na Figura 7.

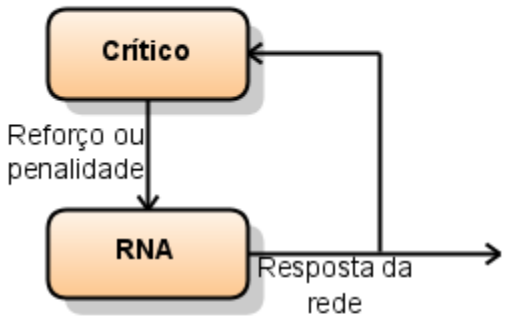

Figura 6. Aprendizado por reforço.

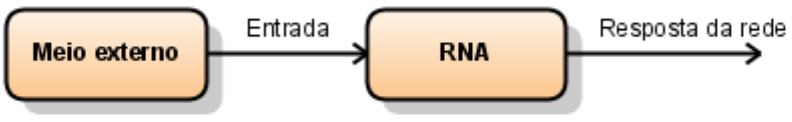

Figura 7. Aprendizado não supervisionado.

2.1.1 Modelos de Estimativa

Os modelos de RNA disponíveis são diversos, cada um com características particulares que o torna mais adequado a um determinado contexto. Os modelos utilizados neste trabalho são:

- Regressão Linear Simples: Não possui parâmetros. Modelos simples como esse são testados para determinar se o problema realmente exige algo mais complexo para ser resolvido. Vide Equação 5, na Seção 2.1.
- RBFN (Rede de Função de Base Radial, do inglês *Radial Basis Function Network*): O parâmetro *numClusters* determina quantos agrupamentos serão permitidos. Esse tipo de RNA utiliza uma função de ativação do tipo gaussiana, como mostra a Equação 6, na Seção 2.1.
- M5P: Consiste numa árvore de decisão regressiva (algoritmo M5). Seus parâmetros foram:
  - *minNumInstances*, que determina o número mínimo das instâncias;
  - *unpruned*, que determina se a árvore poderá, ou não, ser podada;
  - *useUnsmoothed*: que funciona como um filtro estatístico sobre os dados.
- MLP (*Perceptron* de Múltiplas Camadas, do inglês *Multilayer Perceptron*): É uma RNA com uma ou mais camadas escondidas composta por neurônios do tipo *Perceptron*. Seus parâmetros foram:
  - *learningRate*: É a taxa de aprendizado que determina a velocidade com a qual a rede aprende;
  - *momentum*: Valor que influencia no treinamento atuando sobre os pesos;
  - *hiddenLayers*: Determina o número de camadas escondidas;
  - *trainingTime:* Determina a duração de um ciclo, ou época, de aprendizado.

Estes parâmetros são valorados na Subseção 4.3.1, conforme os experimentos realizados.

2.1.2 Validação Cruzada

A validação cruzada é uma estratégia bastante utilizada para treinar, testar e comparar a capacidade de generalização das técnicas de aprendizagem de máquina [7]. Ela é normalmente aplicada sobre uma base de dados considerada pequena [25] e funciona da seguinte forma:

- Os dados disponíveis são divididos em k grupos de tamanhos iguais;
- k-1 grupos são usados no treinamento da rede;
- O grupo que não fez parte do conjunto de treinamento é usado para validar a rede;
- Esse processo é repetido k vezes, sempre variando o grupo que ficará de fora do treinamento para validação.

A Figura 8 ilustra a validação cruzada.

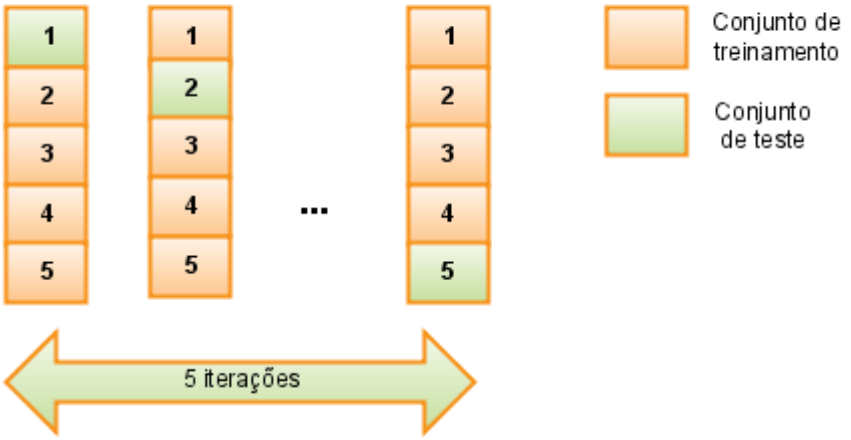

Figura 8. Exemplo de validação cruzada com 5 blocos.

O valor de k é experimental. No WEKA (vide Seção 4.2), seu valor padrão é 10.

2.1.3 MMRE

A Magnitude Média do Erro Relativo (MMRE, do inglês *Mean Magnitude of Relative Error*) é um critério de avaliação para medir a performance de modelos de estimativa de software [14]. Ele é calculado pela Equação 7, onde n é o número de dados, Y_i é o i-ésimo valor de saída e $\hat{Y}_i$ é o i-ésimo resultado estimado pela rede:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|Y_i - \hat{Y}_i\right|}{Y_i} \quad (7).$$

Segundo [12], um $MMRE \leq 0{,}25$ é considerado aceitável para modelos de previsão de esforço.

2.1.4 PRED(25)

PRED(25) é um caso particular do PRED(N) que representa a quantidade de estimativas dentro da margem de 25% dos valores das estimativas reais. Ele é uma medida que prioriza o controle das previsões e auxilia a análise do intervalo de risco das estimativas [7]. Na Equação 8, N é a porcentagem da margem considerada, T é a quantidade de testes, Y_i é o i-ésimo valor de saída e $\hat{Y}_i$ é o i-ésimo resultado estimado pela rede:

$$PRED(N) = \frac{100}{T}\sum_{i}^{T}\begin{cases}1, \dfrac{\left|Y_i - \hat{Y}_i\right|}{Y_i} \leq \frac{N}{100} \\ 0, contrário\end{cases} \quad (8).$$

Por exemplo, um $PRED(25) = 50\%$ significa que metade das estimativas estão dentro de 25% do valor real [23]. Segundo [18], um $PRED(25) \geq 75\%$ é considerado preciso para modelos de previsão de esforço.

2.2 Estimativas em Engenharia de Software

Estimativas em Engenharia de Software estão relacionadas às métricas, que são a medição de uma característica de processos, produtos, projetos e/ou recursos para ajudar no aperfeiçoamento do processo de desenvolvimento adotado [7].

Há várias técnicas de estimativa, como as baseadas no julgamento de especialistas e as por analogia [22]. Na primeira, vários especialistas são consultados através de entrevistas ou formulários para se obter um consenso sobre os valores que se deseja estimar, a técnica Delphi [30] é um exemplo. Na segunda, compara-se o software proposto a projetos anteriores, buscando semelhanças entre eles nas quais as estimativas possam ser baseadas.

Outras técnica são as paramétricas ou algorítmicas [22]. Supondo que o esforço, tamanho, prazo, qualidade, etc. do projeto de software estão relacionados matematicamente, estas técnicas tentam estabelecer equações através dos dados históricos para realizar estimativas para os novos projetos. COCOMO (*COnstructive COst MOdel*) é um exemplo que além do esforço, estima prazo e custo [5].

Também fazem parte das técnicas paramétricas e algorítmicas as técnicas de aprendizagem de máquina, entre elas, as RNAs [9][22][25][32]. Da mesma forma que as demais técnicas, são necessários dados sobre o ambiente de desenvolvimento para se obter uma estimativa. Só que ao invés de elaborar equações, arquiteturas de RNAs são propostas e treinadas com os dados históricos disponíveis. Após o treinamento, a rede generaliza o aprendizado e é capaz de fornecer estimativas.

2.3 Estimativas em Teste de Software

A importância do Teste de Software é reconhecida, mas essa atividade ainda consome muitos recursos no ciclo de vida do produto [1]. Muitos trabalhos buscam minimizar ou prever o esforço para realizar esta atividade [1][27][34].

Da mesma forma que as estimativas em Engenharia de Software, as estimativas em Teste de Software usam dados de suas métricas como entradas para a previsão. Esses dados variam de acordo com diversos fatores, como:

- A informação que será estimada;
- A abordagem dos testes que será aplicada (vide Seção 3.2);
- A natureza do software que será testado;
- Os dados históricos disponíveis, etc.

De acordo com [7], o contexto influencia as estimativas, como mostra a Figura 9. Por isso, na Seção 4.1, os dados de entradas que serão usados para estimar a quantidade de CTs são detalhados.

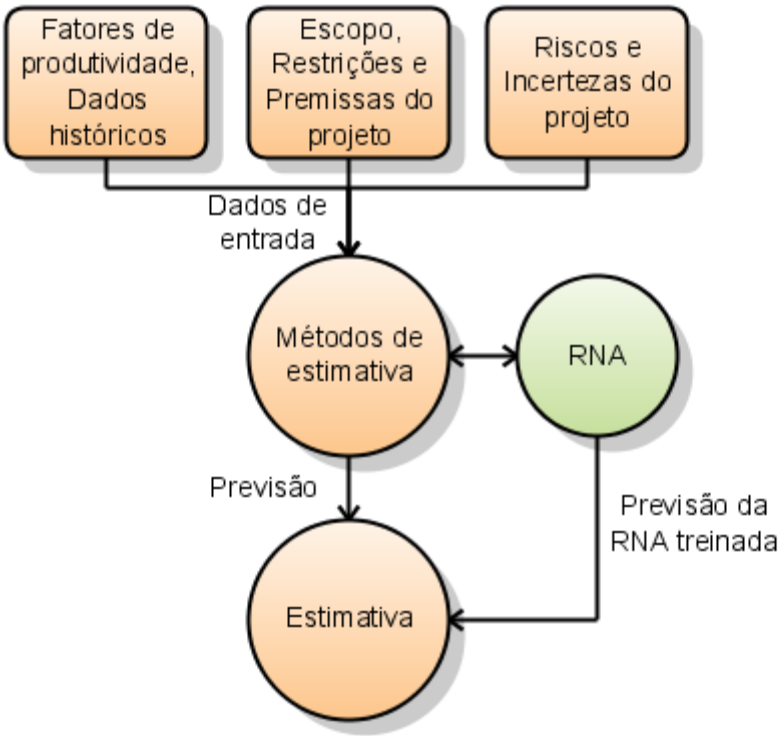

Figura 9. Fatores que influenciam as estimativas, seja RNA ou outros métodos.

Como destaca a Figura 9, o método de estimativa usado neste trabalho é a aplicação de RNAs, pois sua capacidade é reconhecida em diversas áreas, inclusive Engenharia de Software e, mais especificamente, Teste de Software.

Capítulo 3

Teste de Software

Este capítulo descreve um processo genérico de Teste de Software e técnicas de projeto de teste.

3.1 Processo de Teste de Software

Modelos e processos são usados com o objetivo de alcançar um desenvolvimento de software estruturado e controlável [20]. Alguns exemplos bem conhecidos são: Modelo em Cascata, Modelo em Espiral, Modelo V, XP (Programação Extrema, do inglês *eXtreme Programming*), etc. [29]

Testes aparecem em todos esses modelos, mas com significados e proporções diferentes [20]. Independentemente do modelo escolhido, há atividades básicas que existem no processo de Teste de Software. Nas subseções seguintes, essas atividades são descritas.

3.1.1 Planejamento e Controle

O planejamento das atividades de teste inicia-se juntamente com o projeto de desenvolvimento do software. O produto final dessa atividade é o Plano de Teste, que deve conter estimativas sobre os recursos necessários, como: pessoas, tempo, equipamentos, ferramentas, etc.

Uma estratégia de teste deve ser definida, pois, na maioria dos casos, há restrições de tempo que não permitem testes exaustivos, ou seja, é necessário priorizar as partes “mais importantes” do software que será testado. A importância é determinada pelos critérios do cliente, pela análise dos riscos e da complexidade associados ao software, etc. Assim, é possível concentrar esforços no que realmente precisa ser testado. As técnicas de teste que serão aplicadas e o critério de término (cobertura dos testes, por exemplo) também são determinados.

O controle refere-se ao monitoramento do que acontece durante os testes para comparação com o que foi planejado e o rastreamento do progresso das

atividades. Desvios resultam na alteração do plano ou em tentativas de restabelecer o andamento das atividades de acordo com o plano.

3.1.2 Análise e Projeto

Esta fase é dependente da técnica de teste escolhida na fase anterior. O mais importante desta etapa é desenvolver casos de teste (CTs). Para isso, a especificação do software é analisada a fim de determinar se é possível gerar CTs a partir dela. Essa análise verifica a clareza e a testabilidade dos requisitos. Dificuldades para gerar CTs sugerem problemas na especificação e a necessidade de revisá-la a fim de torná-la mais completa, compreensível e/ou detalhada [10].

Um CT deve ter basicamente quatro elementos:

- As pré-condições sob as quais deve ser executado;
- Os dados de entrada que devem ser usados;
- Os procedimentos que devem ser seguidos na execução;
- E, finalmente, o comportamento esperado do software a ser testado, que inclui os dados de saída esperados.

Nesta etapa, os recursos definidos anteriormente devem ser preparados para que estejam disponíveis na execução dos testes.

3.1.3 Execução

Concluídas todas as preparações, os testes podem ser executados manualmente, com auxílio de ferramentas ou automaticamente. As pré-condições e os dados de entrada estabelecidos na Subseção 3.1.2 são usados na execução.

Deve haver a preocupação em documentar a execução: os resultados, o testador responsável, o software testado, etc. Outro item importante são os passos necessários para reproduzir a falha encontrada, para que haja credibilidade nos resultados dos testes [20].

A execução de testes geralmente faz parte de um ciclo que consiste em:

- Executar os testes;
- Reportar as falhas encontradas e

- Re-testar as falhas para confirmar sua correção.

As falhas são classificadas por severidade, de acordo com o impacto que causariam ao cliente ou usuário. As falhas mais severas (com maior impacto) podem ser corrigidas e re-testadas antes das de menor severidade.

É importante destacar que a correção de falhas é tarefa da equipe de desenvolvimento. A equipe de teste busca e reporta falhas do sistema, mas investigar suas causas e corrigi-las não fazem parte de suas atividades.

3.1.4 Avaliação do Critério de Encerramento e Relatório

O critério de encerramento determinado no Plano de Teste é comparado às métricas aplicadas à execução dos testes. Um desses critérios, por exemplo, é a Percentagem de Detecção de Defeitos (DDP, do inglês *Defect Detection Percentage*), que determina o encerramento dos testes caso nenhuma falha seja detectada em mais de uma hora [16].

A comparação do critério com os valores reais do software pode resultar em três principais cenários [20]:

- Os critérios foram atingidos e os testes podem ser normalmente encerrados;
- Os critérios foram atingidos mas testes adicionais são necessários;
- Os critérios não foram atingidos. Nesse caso, os critérios podem não ter sido realistas e, na prática, não seria possível atingi-los.

Há dois fatores externos que influenciam no encerramento da execução: o tempo e o custo. Dependendo dos recursos alocados no planejamento, pode ser possível testar mais que o previsto ou pode ser necessário cancelar a execução dos testes restantes. Por isso, aplicam-se prioridades aos CTs, procurando-se testar as partes do software consideradas mais críticas primeiro. Assim, caso não seja possível completar todos os testes, pelo menos os mais prioritários devem ter sido executados.

Em alguns casos, pode ser necessário um relatório formal informando os resultados (quanto às falhas e ao critério de encerramento) e a finalização da execução dos testes.

3.2 Técnicas de Projeto de Teste

As técnicas descritas a seguir são de análise dinâmica, ou seja, o software deve ser executável para a realização dos testes. Em contraposição, também existem técnicas de análise estática, que não serão abordadas neste trabalho.

Análise dinâmica divide-se em duas abordagens: caixa branca e caixa preta. A primeira contempla o uso do código-fonte para projetar os CTs e não será abordada neste trabalho. Na segunda, os testes são projetados a partir da especificação ou dos requisitos do sistema que é tratado como uma caixa preta, onde somente as entradas e as saídas podem ser observadas, ou seja, o foco é a funcionalidade e não a estrutura do sistema [20].

A seguir, três técnicas caixa preta são brevemente explicadas. Elas representam as técnicas que mais influenciam a atividade de projeto de teste do estudo de caso (Vide Capítulo 4).

3.2.1 Particionamento em Classes de Equivalência

O testador assume que o domínio de valores aceitos pelo sistema pode ser dividido em classes equivalentes (particionamento) e que o sistema responderá da mesma maneira para quaisquer valores que pertençam a uma mesma classe [26]. Então, considera-se suficiente testar apenas um representante da classe. Classes de valores inválidos também são testadas.

Em testes de sistema, por exemplo, essa técnica aplica-se aos campos de entrada de dados numa tela. CTs podem ser sistematicamente derivados a partir desta técnica, verificando como o sistema deve se comportar, segundo a especificação, quanto ao processamento dos valores de cada classe.

3.2.2 Análise do Valor Limite

Esta técnica complementa as classes de equivalência. Geralmente, as falhas encontram-se nas fronteiras das classes, pois elas não são bem definidas ou são implementadas incorretamente [3]. Mas para que seja possível aplicar essa técnica, as fronteiras da classe devem ser identificáveis.

Se os valores pertencentes a uma classe correspondem aos meses do ano, por exemplo, valores limites seriam: 1, 12 (dentro da classe), 0 e 13 (fora da classe).

3.2.3 Teste de Caso de Uso

Esta técnica é útil principalmente para testes de sistema e de aceitação (nos quais o foco é a perspectiva do cliente ou usuário), pois casos de uso e seus diagramas fornecem uma visão externa do sistema, ideal para identificar interações entre os seus componentes [17][24].

Através dos UCs, é possível identificar pré-condições, pós-condições, fluxos alternativos (relacionamentos *extend*, num diagrama de UC) e interações (relacionamentos *include* num diagrama). Esses são elementos que podem ser usados para a geração de CTs.

Capítulo 4

Estudo de Caso

A seguir, o estudo de caso realizado neste trabalho é descrito em detalhes.

4.1 Dados de Entrada

Os dados de entrada são a representação do problema que a RNA solucionará. Eles são as variáveis consideradas na resposta da rede. Na Tabela 1, são listados os dados de entrada selecionados do contexto do problema.

A seguir, cada item é explicado quanto à motivação de sua escolha e como sua coleta foi realizada. Eles são agrupados em dados do caso de uso, do protótipo e do projeto de teste.

4.1.1 Dados do Caso de Uso

Os testes devem investigar cada fluxo do UC [20], logo, a quantidade de fluxos é muito relevante (Vide Subseção 3.2.3). Por exemplo, na Figura 10, a seta central indica o fluxo principal, que é dividido em dois subfluxos, há dois fluxos alternativos que desviam do principal e há três fluxos de exceção que encerram o UC prematuramente. Analisando a Figura 10, podemos determinar que no nosso projeto de teste deve haver pelo menos um CT para cada cenário proporcionado pelos fluxos, totalizando sete.

Os UCs utilizados neste trabalho seguem um padrão que inclui subfluxos. O fluxo principal por si só não executa nenhuma ação, ele só inicia o UC e transfere a responsabilidade para os subfluxos. Por exemplo, um UC chamado Manter Entidade, seguindo as funcionalidades básicas CRUD (Incluir, Consultar, Atualizar e Excluir, do inglês, Create, Read, Update and Delete) teria quatro subfluxos: Incluir Entidade, Consultar Entidade, Atualizar Entidade e Excluir Entidade. Neste caso, quando há subfluxos, o fluxo principal não é contado, pois não há nada para ser testado nele.

Tabela 1. Dados de entrada da RNA divididos por artefatos: caso de uso, protótipo e projeto de teste.

<table>
<tr><td colspan="2">Dados de Entrada</td><td colspan="2">Artefatos</td></tr>
<tr><td>1</td><td>Básico e Subfluxos</td><td rowspan="3">Fluxos</td><td rowspan="4">Caso de Uso</td></tr>
<tr><td>2</td><td>Alternativos</td></tr>
<tr><td>3</td><td>Exceções</td></tr>
<tr><td>4</td><td colspan="2">Regras de Negócio</td></tr>
<tr><td>5</td><td colspan="2">Telas</td><td rowspan="11">Protótipo</td></tr>
<tr><td>6</td><td>Button</td><td rowspan="10">Tags (html)</td></tr>
<tr><td>7</td><td>Checkbox</td></tr>
<tr><td>8</td><td>File</td></tr>
<tr><td>9</td><td>Password</td></tr>
<tr><td>10</td><td>Radio</td></tr>
<tr><td>11</td><td>Submit</td></tr>
<tr><td>12</td><td>Text</td></tr>
<tr><td>13</td><td>Textarea</td></tr>
<tr><td>14</td><td>Select</td></tr>
<tr><td>15</td><td>Granularidade</td><td colspan="2" rowspan="2">Projeto de Teste</td></tr>
<tr><td>16</td><td>Quantidade de CTs</td></tr>
</table>

Além disso, as regras de negócio de cada fluxo devem ser observadas (são as setas destacadas da Figura 10). Um fluxo pode ter mais de uma regra de negócio

associada. E uma regra de negócio pode aparecer em vários fluxos. A coleta não detalhou isso, apenas o número total de regras de negócio por UC foi contabilizado.

A coleta foi realizada manualmente, contando-se os itens no sumário de cada UC.

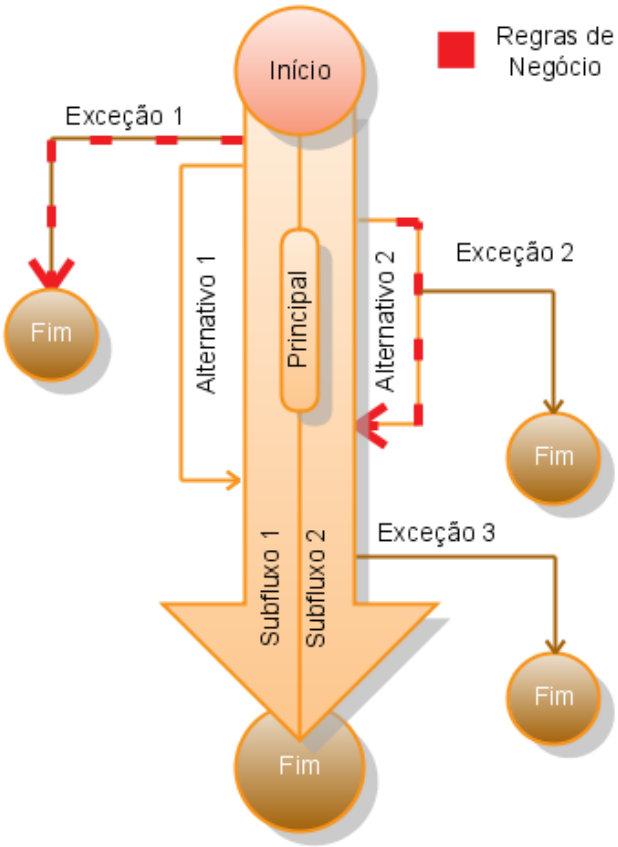

Figura 10. Fluxos de um caso de uso. A seta principal é dividida em dois subfluxos, as setas laranjas são fluxos alternativos, as marrons são fluxos de exceção e as destacadas em vermelho contêm regras de negócio.

4.1.2 Dados do Protótipo

O protótipo consiste num conjunto de arquivos, a maioria com extensão html (de HTML, Linguagem de Marcação de Hipertexto, do inglês *HyperText Markup Language*), para ambiente *Web*. Cada arquivo é uma tela do sistema. Para coletar os dados das telas, o protótipo foi analisado com a ajuda de um programa capaz de

identificar as *tags* da linguagem HTML no código-fonte dos arquivos. Essas *tags* estão listadas na Tabela 1, do item 6 ao 14, o item 5 é o total de telas do UC.

As *tags* representam elementos na tela que devem ser testados. A *tag text*, por exemplo, é um campo de texto de uma única linha. Três dos possíveis testes que podem ser executados para este elemento do protótipo são: limite inferior, limite superior e caracteres inválidos (Vide Subseções 3.2.1 e 3.2.2). Suponha que essa *tag text* refere-se a um campo chamado Nome de um formulário que não pode ser deixado em branco, assim:

- Poderia ser especificado que o limite inferior do campo é um caractere e o limite superior é duzentos;
- Da mesma forma, apenas letras e espaços poderiam ser inseridos nesse campo;
- Números e caracteres especiais não devem ser permitidos.

As demais *tags* também possuem testes associados da mesma forma que *text*, por isso, a contagem desses elementos presentes no protótipo é importante no contexto do problema. Espera-se que a rede consiga identificar relações entre a presença dessas *tags* e a quantidade de CTs.

4.1.3 Dados do Projeto de Teste

Como dado de entrada, definiu-se um atributo granularidade para diferenciar os diversos projetos de teste que são usados. A granularidade pode ter três valores de acordo com os critérios dos CTs num projeto:

- Baixa granularidade (valor igual a 1): mais de uma funcionalidade é testada por CT;
- Média granularidade (valor igual a 2): uma ou duas funcionalidades são testadas e podem existir documentos centralizadores de testes que são referenciados pelo CT. Esses documentos descrevem testes comuns que são realizados para vários UCs. Então, eles são referenciados para evitar sua repetição;
- Alta granularidade (valor igual a 3): apenas uma funcionalidade é testada por CT.

Essas condições influenciam na quantidade de CTs, pois projetos com alta granularidade tendem a ter um maior número que os de baixa granularidade. Para este trabalho, o valor da granularidade foi generalizado por sistema, ou seja, projetos de testes feitos para um mesmo sistema terão a mesma granularidade. Isso reflete a realidade do contexto do qual os dados foram colhidos, pois todos os projetos para um sistema devem seguir o mesmo padrão.

É importante ressaltar que, mesmo com os critérios de valoração definidos, este valor é subjetivo.

Quantidade de Casos de Teste

Embora encontre-se junto aos demais dados de entrada, a quantidade de CTs representa o valor a ser estimado pela RNA. Mas devido à necessidade de exemplos para o treinamento, esta informação é fornecida para ser comparada ao resultado da rede e, assim, calcular o erro que ajustará seus pesos.

Esta informação é obtida nos projetos de teste (executados ou não) que contêm os CTs projetados manualmente para um determinado UC.

4.2 Ferramenta WEKA

A ferramenta WEKA (*Waikato Environment for Knowlegde Analysis*) implementa diversos algoritmos de aprendizagem de máquina que podem ser aplicados a problemas de classificação, agrupamento, associação e regressão [37]. Ela é gratuita sob a licença GPL (*General Public License*) e sua tela inicial é exibida na Figura 11. Pré-requisitos de hardware e software para o uso do WEKA podem ser obtidos em [35].

A versão utilizada da ferramenta tem alterações implementadas por [8], que adicionam dois critérios de avaliação de performance para as RNAs: o MMRE (*Mean Magnitude of Relative Error*) e o PRED(25) (Vide Subseções 2.1.3 e 2.1.4, respectivamente).

Figura 11. Tela inicial da ferramenta WEKA. Esta versão contém alterações por [8].

4.2.1 Formato ARFF

O WEKA utiliza dois formatos de arquivo para importar as bases de dados: ARFF (*Attribute-Relation File Format*) e CSV (*Comma-Separated Value*). Neste trabalho, optou-se pelo ARFF porque ele é o mais comumente usado e possui um formato bem definido [7]. Os elementos contidos neste formato são descritos a seguir:

- @relation: Nome da relação que é descrita no arquivo;
- @attribute: Declara atributos e seus tipos, que podem ser numeric, real, integer, string e date;
- @data: A partir deste elemento, os dados sobre os atributos devem ser informados.

- %: Toda linha iniciada com “%” é um comentário.

O arquivo ARFF utilizado neste trabalho encontra-se no Apêndice A.

4.3 Resultados

Esta seção apresenta os resultados obtidos com a ferramenta WEKA e os dados descritos na Seção 4.1. Os modelos de RNAs e seus parâmetros são detalhados juntamente com a estratégia de treinamento.

4.3.1 Parâmetros e Valores dos Modelos de Estimativa

Os experimentos basearam-se naqueles realizados em [8][13][25]. Os modelos de RNA descritos na Subseção 2.1.1 foram treinados, alguns com diferentes parâmetros. Os parâmetros e valores dos modelos utilizados nos experimentos são resumidos na Tabela 2.

Tabela 2. Modelos de estimativas disponíveis no WEKA e utilizados no estudo de caso.

Modelo	Parâmetros	Valores
Regressão Linear Simples	-	
RBFN	*numClusters*	{2, 5, 10, 20, 30, 40, 50}
M5P	*minNumInstances*	{1, 4, 6, 7, 10}
	unpruned	{*TRUE*, *FALSE*}
	useUnsmoothed	{*TRUE*, *FALSE*}
MLP	*learningRate*	{0,01}
	momentum	{0,02}
	hiddenLayers	{1, 2, 3}
	trainingTime	{500}

4.3.2 Experimentos

Os experimentos foram realizados de acordo com os modelos, seus parâmetros e valores resumidos na Tabela 2. Todos eles utilizaram a estratégia de validação cruzada descrita na Subseção 2.1.2 com k igual a 10 e os critérios de avaliação MMRE e PRED(25), detalhados nas Subseções 2.1.3 e 2.1.4, respectivamente.

Os atributos da base de dados utilizada foram descritos na Seção 4.1. O total de dados foi 109, coletados de 4 sistemas distintos.

Os resultados do treinamento da RNA M5P (*minNumInstances* = 1, *unpruned* = TRUE, *useUnsmoothed*=FALSE) são mostrados na Tabela 3. Há 109 respostas da rede, isto é, uma para cada par entradas-saída, mas apenas os 30 primeiros são listados aqui. A coluna Real apresenta a quantidade real de CTs presentes no projeto de teste, a coluna Estimado apresenta os valores dos CTs estimados pela rede de acordo com os dados de entrada. A coluna erro exibe o valor da Equação 9 e a coluna Percentagem de Erro exibe o valor da Equação 10, respectivamente:

$$Erro = Estimado - \mathrm{Re}\,al \qquad (9),$$

$$Erro(\%) = \frac{|Estimado - \mathrm{Re}\,al|}{\mathrm{Re}\,al} * 100\% \qquad (10).$$

Tabela 3. 30 primeiros resultados do treinamento da RNA M5P (*minNumInstances* = 1, *unpruned* = TRUE, *useUnsmoothed* = FALSE).

	Real	Estimado	Erro	Percentagem de Erro (%)
1	5	34.775	29.775	595.491
2	66	63.645	-2.355	3.568
3	81	62.8	-18.2	22.469
4	44	66.878	22.878	51.995
5	36	51.138	15.138	42.05

6	50	55.229	5.229	10.458
7	294	134.493	-159.507	54.254
8	40	42.432	2.432	6.081
9	93	63.931	-29.069	31.257
10	28	59.667	31.667	113.096
11	117	113.188	-3.812	3.258
12	63	52.996	-10.004	15.879
13	141	150.177	9.177	6.508
14	146	230.136	84.136	57.627
15	7	12.328	5.328	76.112
16	184	146.269	-37.731	20.506
17	44	68.009	24.009	54.567
18	123	90.166	-32.834	26.694
19	48	66.076	18.076	37.659
20	59	51.381	-7.619	12.913
21	182	15.929	-166.071	91.248
22	99	82.32	-16.68	16.849
23	59	45.698	-13.302	22.545
24	14	41.037	27.037	193.118
25	45	64.023	19.023	42.274
26	11	46.122	35.122	319.295
27	37	79.631	42.631	115.218
28	34	51.932	17.932	52.742
29	17	56.342	39.342	231.424
30	2	127.843	-38.157	22.986

Supondo que o tempo médio para projetar testes por CT seja $3\text{min}/CT$ e aplicando a Equação 1, da Seção 1.1 para o resultado 2 da Tabela 3, tem-se que $Esforço = 63{,}645 CTs * 3\text{min}/CT = 190{,}935\,\text{min} \cong 3h$.

Da mesma forma, supondo que o preço por cada CT seja $1{,}99 reais$ e aplicando a Equação 2, da Seção 1.1 para o resultado 2 da Tabela 3, tem-se que

$Custo = 63{,}645 CTs * 1{,}99 reais/CT = 126{,}65 reais$. Assim, com a estimativa da quantidade de CTs é possível derivar o Esforço e o Custo.

A comparação das técnicas utilizadas encontra-se na Tabela 4. O PRED(25) e o MMRE são explicados nas Subseções 2.1.4 e 2.1.3, respectivamente.

Tabela 4. Comparação das técnicas para a base de dados utilizada no estudo de caso.

Técnica (parâmetros)	PRED(25)	MMRE
Regressão Linear Simples	27.5229%	1.6298
RBFN (n=2)	42.2018%	1.8650
RBFN (n=5)	32.1101%	1.6275
RBFN (n=10)	36.6972%	1.4001
RBFN (n=20)	33.0275%	2.0263
RBFN (n=30)	33.9450%	2.6074
RBFN (n=40)	25.6881%	4.3162
RBFN (n=50)	31.1927%	6.7221
M5P (m=1 un=FALSE us=FALSE)	38.5321%	1.2368
M5P (m=4 un=FALSE us=FALSE)	38.5321%	1.2368
M5P (m=6 un=FALSE us=FALSE)	41.2844%	1.3298
M5P (m=7 un=FALSE us=FALSE)	43.1193%	1.3469
M5P (m=10 un=FALSE us=FALSE)	37.6147%	1.2636
M5P (m=1 un=TRUE us=FALSE)	**47.7064%**	1.3408
M5P (m=4 un=TRUE us=FALSE)	**47.7064%**	1.3408
M5P (m=6 un=TRUE us=FALSE)	46.7890%	1.3465
M5P (m=7 un=TRUE us=FALSE)	44.9541%	1.4099
M5P (m=10 un=TRUE us=FALSE)	41.2844%	1.3867
M5P (m=1 un=FALSE us=TRUE)	35.7798%	1.2253
M5P (m=4 un=FALSE us=TRUE)	35.7798%	1.2253
M5P (m=6 un=FALSE us=TRUE)	35.7798%	1.3484
M5P (m=7 un=FALSE us=TRUE)	38.5321%	1.3651
M5P (m=10 un=FALSE us=TRUE)	32.1101%	1.2867

M5P (m=1 un=TRUE us=TRUE)	32.1101%	1.2867
M5P (m=4 un=TRUE us=TRUE)	30.2752%	**1.1235**
M5P (m=6 un=TRUE us=TRUE)	33.9450%	1.2534
M5P (m=7 un=TRUE us=TRUE)	36.6972%	1.7750
M5P (m=10 un=TRUE us=TRUE)	36.6972%	1.7750
MLP (l=0,01 m=0,02 h=1 t=500)	43.1193%	1.3133
MLP (l=0,01 m=0,02 h=2 t=500)	42.2018%	1.2570
MLP (l=0,01 m=0,02 h=3 t=500)	41.2844%	1.2414

A Tabela 4 mostra que os melhores resultados foram para a técnica M5P. Os modelos M5P (m=1 un=TRUE us=FALSE) e M5P (m=4 un=TRUE us=FALSE) obtiveram 47.7064%, o maior valor para o PRED(25). Isto significa que quase metade das estimativas estão dentro de 25% do valor real. Já o modelo M5P (m=4 un=TRUE us=TRUE) obteve o menor valor para o MMRE, com 1.1235, ou seja, a menor magnitude média do erro relativo.

Embora comparados aos valores considerados aceitáveis para previsão de esforço [12][18] esses resultados não sejam muito precisos, é importante ressaltar que não é esforço que está sendo estimado, e sim a quantidade de casos de teste.

Capítulo 5

Conclusões

Neste capítulo, encerra-se o trabalho com as considerações finais, as contribuições e os trabalhos futuros.

Os resultados do estudo de caso mostram que esta abordagem quantitativa tem potencial para auxiliar nos pontos propostos, embora tenha sido identificada a necessidade de ampliar o conjunto de treinamento para aumentar a precisão da resposta da rede.

5.1 Considerações Finais

Com base na forma como é obtida a estimativa da quantidade de CTs neste trabalho, não é possível responder a perguntas como as seguintes:

- Essa quantidade de CTs é suficiente para cobrir todo ou a maior parte do sistema?
- São realizados testes para as funcionalidades mais críticas ou importantes do sistema?
- Podemos parar de projetar CTs ao atingir a quantidade que foi estimada?

As respostas dependem do critério de teste adotado e também de análises que verifiquem, por exemplo, a cobertura e a conformidade dos testes com relação às especificações.

5.2 Contribuições

Prever a quantidade de CTs pode contribuir para:

- O planejamento das atividades de teste através das estimativas de esforço e custo derivadas do valor obtido pela RNA;
- Conhecer o perfil da equipe de teste.

5.3 Trabalhos Futuros

Ampliar a base de dados para melhorar o treinamento da RNA é a principal proposta de continuidade deste trabalho. Além de fazer experimentos com outras técnicas e parâmetros em busca de resultados mais precisos.

Bibliografia

(Ordenação aqui deve ser alfabética pelo sobrenome do primeiro autor)

[1] ANDERSON, Charles; VON MAYRHAUSER, Anneliese; MRAZ, Rick. **On the Use of Neural Networks to Guide Software Testing Activities**. Proceedings of the IEEE International Test Conference on Driving Down the Cost of Test, p. 720-729, 1995.

[2] ARANHA, E.; BORBA, P. **Test Effort Estimation Models Based on Test Specifications. Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION**, p. 67-71, 2007.

[3] BASILI, Victor; SELBY, Richard. **Comparing the Effectiveness of Software Testing Strategies**. IEEE Transactions on Software Engineering, v.13, n.12, p. 1278-1296, 1987.

[4] BLACK, Rex. **Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing**. 2nd ed., Wiley, 2002. 528 p.

[5] BOEHM, Barry W. **Software Engineering Economics**. Prentice Hall PTR, 1981. 688 p.

[6] BRAGA, Antônio; CARVALHO, André; LUDERMIR, Teresa. **Redes Neurais Artificiais: Teoria e Aplicações**. 2 ed. Livros Técnicos e Científicos (LTC), 2007. 238 p.

[7] BRAGA, Petrônio. **Ferramenta para Estimativa de Esforço de Projetos de Software Baseada em Técnicas de Computação Inteligente**. 2008. 130 f. Trabalho de Dissertação de Mestrado de Engenharia da Computação. Departamento de Sistemas e Computação, Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife.

[8] BRAGA, Petrônio; OLIVEIRA, Adriano; RIBEIRO, Gustavo; MEIRA, Silvio. **Bagging Predictors for Estimation of Software Project Effort**. International Joint Conference on Neural Networks, p. 1595-1600, 2007.

[9] CAPRETZ, Luiz; REN, Jing; HO, Danny; HUANG, Xishi. **Improving the cocomo model using a neuro-fuzzy approach**. Applied Soft Computing, v. 7, n. 1, p. 29–40, 2007.

[10] CHANDRASEKHARAN, M.; DASARATHY, B.; KISHIMOTO, Z. **Requirements-Based Testing of Real-Time Systems. Modeling for Testability**. Computer, v. 18, n. 4, p. 71-80, 1985.

[11] CHEMUTURI, Murali. **Test Effort Estimation**. Disponível em: <http://www.chemuturi.com/Test%20Effort%20Estimation.pdf> Acesso em: 24 de setembro de 2008.

[12] CONTE, Samuel D. **Software Engineering Metrics and Models**. Benjamin-Cummings Pub Co, 1986. 403 p.

[13] DELEN, Dursun; WALKER, Glenn; KADAM, Amit. **Predicting breast cancer survivability: a comparison of three data mining methods**. Artificial Intelligence in Medicine, p. 113-127, 2005.

[14] FOSS, Tron; STENSRUD, Erik; KITCHENHAM, Barbara; MYRTVEIT, Ingunn. **A Simulation Study of the Model Evaluation Criterion MMRE**. IEEE Transactions on Software Engineering, v. 29, n. 11, p. 985-995, 2003.

[15] HAYKIN, Simon. **Redes Neurais: Princípios e Práticas**. 2 ed. Bookman, 2007. 900 p.

[16] HUANG, Liguo. **A Value-Based Process for Achieving Software Dependability**. International Software Process Workshop, p. 108-121, 2005.

[17] KIM, Y. G.; HONG, H. S.; BAE, D. H.; CHA, S. D. **Test cases generation from UML state diagrams**. IEE Proceedings Software, v. 146, n. 4, p. 187-192, 1999.

[18] KUMAR, K.; RAVI, V.; CARR, Mahil; KIRAN, N. **Software development cost estimation using wavelet neural networks**. The Journal of Systems and Software, v.81, n.11, p. 1853-1867, 2008.

[19] LAST, Mark; KANDEL, Abraham; BUNKE, Horst. **Artificial Intelligence Methods In Software Testing** (Series in Machine Perception & Artifical Intelligence v.56). World Scientific Publishing Company, 2004. 208 p.

[20] LINZ, Tilo; SCHAEFER, Hans; SPILLNER, Andreas. **Software Testing Foundations**. 2nd ed., Rockynook, 2007. 288 p.

[21] MCCULLOCH, W.; PITTS, W. **A Logical Calculus of the Ideas Immanent in Nervous Activity**. Bulletin of Mathematical Biophysics, v.5, p. 115-133, 1943.

[22] MEIRA, Silvio; OLIVEIRA, Adriano; BRAGA, Petrônio. **A GA-based feature selection and parameters optimization for support vector regression applied to software effort estimation**. ACM Symposium on Applied Computing, p. 1788–1792, 2008.

[23] MENZIES, Tim; LUM, Karen; HIHN, Jairus. **The Deviance Problem in Effort Estimation**. Proceedings of the PROMISE workshop, 2006. Disponível em: <http://promisedata.org/pdf/phil2006MenziesLumHihn.pdf> Acesso em: 1 de dezembro de 2008.

[24] NEBUT, C.; FLEUREY, F.; LE TRAON, Y.; JÉZÉQUEL, J. M. **Automatic Test Generation: A Use Case Driven Approach**. IEEE Transactions on Software Engineering, v. 32, n. 3, p. 140-155, 2006.

[25] OLIVEIRA, Adriano. **Estimation of software project effort with support vector regression**. Neurocomputing, v. 69, n. 13-15, p. 1749–1753, 2006.

[26] OSTRAND, T. J.; BALCER, M. J. **The category-partition method for specifying and generating fuctional tests**. Communications of the ACM, v. 31, n. 6, p. 676-686, 1988.

[27] PHAM, Hoang; WANG, Hongzhou. **A quasi-renewal process for software reliability and testing costs**. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, v.31, p. 623-631, 2001.

[28] PINKSTER, Iris; JANSSEN, Dennis; VAN VEENENDAAL, Erik; VAN DE BURGT, Bob. **Successful Test Management: An Integral Approach**. Springer-Verlag, 2005. 373 p.

[29] PRESSMAN, Roger. **Engenharia de Software**. 6 ed., Mc Graw Hill, 2006. 720 p.

[30] Project Management Institute. **A Guide to the Project Management Body of Knowledge**. 3rd ed., 2004. 380 p.

[31] RUMELHART, David; HINTON, Geoffrey; WILLIAMS, Ronald. **Learning representations by back-propagating errors**. Nature, v.323, p.533-536, 1986.

[32] SANT'ANNA, Nilson; DA SILVA, José; TRONTO, Iris. **An investigation of artificial neural networks based prediction systems in software project management**. The Journal of Systems and Software, v.81, n.3, p. 356–367, 2008.

[33] SPILLNER, Andreas; LINZ, Tilo; ROSSNER, Thomas; WINTER, Mario. **Software Testing Practice: Test Management: A Study Guide for the Certified Tester Exam ISTQB Advanced Level**. Rocky Nook, 2007. 339 p.

[34] VANMALI, Meenakshi; LAST, Mark; KANDEL, Abraham. **Using a Neural Network in the Software Testing Process**. International Journal of Intelligent Systems, v.17, p. 45-62, 2002.

[35] **WEKA**. Software disponível em: <http://www.cs.waikato.ac.nz/ml/weka/> Acesso em: 10 de novembro de 2008.

[36] WIDROW, Bernard; HOFF, Marcian E. **Adaptive Switching Circuits**. Institute of Radio Engineers, Western Electronic Show and Convention, p. 96-104. 1960.

[37] WITTEN, Ian; FRANK, Eibe. **Data Mining: Practical machine learning tools and techniques**. 2nd ed. Morgan Kaufmann, 2005. 525 p.

Apêndice A

Arquivo ARFF

```
% 1. Title: Test Cases Quantity Estimation Database
%
% 2. Sources:
% (a) Company: Pitang Consultoria e Sistemas S/A
% (b) Creator: Renata Melo (renata.melo@gmail.com)
% (c) Date: October, 2008

@relation test-cases-quantity

% Total: 16
%
% Attributes from Use Cases
% Total: 4
@attribute primary-flow numeric
@attribute secondary-flow numeric
@attribute exception numeric
@attribute business-rule numeric

% Attributes from Prototypes
% Total: 10
@attribute screen numeric
@attribute button numeric
```

@attribute checkbox numeric

@attribute file numeric

@attribute password numeric

@attribute radio numeric

@attribute submit numeric

@attribute text numeric

@attribute textarea numeric

@attribute select numeric

% Attributes from Test Projects

% Total: 2

@attribute granularity numeric

@attribute test-case numeric

@data

% Total: 109

%

% Data from System-01 (Incomplete)

% Total: 2/20

1,5,3,4,7,14,27,0,0,26,0,10,0,3,3,28

3,0,2,1,18,43,50,0,0,13,0,19,0,0,3,23

% Data from System-02

% Total: 15

9,1,6,5,4,15,0,0,0,10,0,14,3,6,2,61

5,6,9,19,17,48,0,0,0,20,0,189,0,22,2,294

1,1,2,1,1,3,0,0,0,0,0,5,0,1,2,14

1,1,1,8,0,0,0,0,0,0,0,0,0,2,14

1,0,3,9,1,2,0,0,0,0,0,2,0,0,2,11

1,2,2,6,6,15,12,0,0,0,0,2,0,13,2,27

1,2,2,6,10,24,36,0,0,0,0,4,0,26,2,31

4,1,4,7,2,6,0,0,0,0,0,1,0,3,2,21

5,1,4,5,7,21,0,0,0,0,0,48,0,8,2,45

4,2,6,6,6,28,0,0,0,17,0,3,0,12,2,45

1,1,2,10,1,2,0,0,0,0,0,0,0,5,2,166

1,0,1,2,0,0,0,0,0,0,0,0,0,0,2,5

2,3,2,14,7,12,0,0,0,17,0,0,0,10,2,195

1,2,5,5,3,5,0,0,0,0,0,0,2,6,2,25

6,4,2,12,23,27,0,0,0,0,0,4,0,7,2,44

% Data from System-03

% Total: 17

5,8,5,13,9,32,0,0,0,3,0,23,0,30,1,97

5,3,4,5,5,13,0,0,0,3,0,6,0,3,1,56

5,8,7,7,5,21,0,0,0,11,0,6,0,4,1,125

5,13,14,11,9,41,21,0,0,17,0,22,0,13,1,174

3,1,3,3,5,10,0,0,0,0,0,0,10,0,25,1,47

4,3,5,6,6,19,0,0,0,2,0,14,0,4,1,58

4,4,5,6,6,20,0,0,0,9,0,8,0,6,1,124

6,2,4,9,6,14,0,0,0,1,0,29,2,4,1,69

1,2,2,3,12,28,4,0,0,4,0,6,0,2,1,30

2,1,4,5,7,10,0,0,0,2,0,14,0,0,1,42

5,6,4,5,9,24,0,0,0,8,0,12,0,4,1,73

9,7,7,10,12,33,0,0,0,5,0,15,7,28,1,128

5,16,12,17,12,32,0,0,0,6,0,7,2,4,1,141

1,1,15,11,1,1,0,1,0,0,0,1,0,0,1,51

1,1,5,3,2,1,0,0,0,0,0,5,0,0,1,71

4,3,10,15,4,10,0,0,0,2,0,16,0,5,1,81

6,6,16,18,37,96,0,0,0,20,0,43,5,45,1,184

% Data from System-04

% Total: 75

1,1,0,3,0,0,0,0,0,0,0,0,0,0,1,3

13,9,9,22,73,117,56,0,0,119,0,186,0,168,1,146

4,2,6,7,8,13,0,0,0,4,0,11,0,3,1,63

7,3,5,6,39,64,84,0,0,14,0,35,8,14,1,123

5,2,12,10,22,42,2,4,0,8,0,33,10,47,1,99

7,1,7,11,17,29,2,0,0,18,0,8,4,12,1,98

5,8,4,17,17,30,2,0,0,12,0,10,6,9,1,91

2,3,6,17,9,18,3,0,0,16,0,11,0,0,1,43

3,3,4,8,13,23,4,0,0,0,0,0,0,11,1,34

5,2,3,9,10,18,3,0,0,6,0,9,7,9,1,69

2,6,6,14,7,8,10,0,0,0,0,10,0,2,1,50

2,5,5,11,11,15,8,0,0,0,0,0,0,4,1,49

5,1,3,4,9,16,0,0,0,4,0,13,4,3,1,59

5,1,4,6,9,16,0,4,0,6,0,52,9,0,1,87

6,5,6,13,52,97,5,0,0,68,0,115,11,28,1,123

1,0,0,3,0,0,0,0,0,0,0,0,0,0,1,2

3,11,13,39,11,12,3,0,0,36,0,32,4,5,1,190

3,0,3,3,24,50,84,0,0,0,0,15,1,8,1,51

6,7,9,14,21,25,15,0,0,0,0,10,4,3,1,99

1,4,0,11,0,0,0,0,0,0,0,0,0,1,1

6,6,5,8,15,20,3,2,0,11,0,10,0,2,1,102

1,10,2,19,17,18,13,0,0,27,0,115,7,3,1,78

2,4,1,8,0,0,0,0,0,0,0,0,0,1,66

1,3,3,5,15,31,0,0,0,16,0,21,0,14,1,37

5,4,1,12,0,0,0,0,0,0,0,0,0,1,17

10,3,8,13,53,149,171,0,0,22,0,180,15,43,1,235

2,10,1,13,7,13,26,0,0,0,0,6,1,1,1,52

7,11,8,20,22,32,38,0,0,68,0,43,1,7,1,160

2,9,8,25,20,35,56,0,0,18,0,7,2,0,1,68

7,4,9,14,22,29,13,0,0,4,0,13,3,11,1,111

9,6,12,25,58,84,11,0,0,34,0,340,24,86,1,175

7,2,7,12,38,47,26,0,0,16,0,10,0,8,1,155

1,2,5,9,9,14,0,0,0,4,0,12,0,15,1,48

1,0,3,8,7,11,0,0,0,22,0,7,0,10,1,48

1,7,8,17,11,11,9,0,0,14,0,1,3,1,1,61

6,5,4,8,25,37,24,0,0,0,0,18,4,14,1,71

2,0,1,25,5,10,0,0,0,0,0,0,5,0,1,58

7,9,8,18,27,34,54,0,0,11,0,68,5,67,1,157

1,3,2,11,9,11,14,0,0,23,0,26,2,1,1,120

5,1,2,3,11,20,20,0,0,20,0,11,0,13,1,56

1,1,3,7,0,0,0,0,0,0,0,0,0,1,19

1,2,1,3,4,5,0,0,0,0,0,9,0,2,1,26

1,6,6,9,17,32,120,0,0,0,0,48,1,18,1,57

1,14,2,15,23,35,10,0,0,4,0,2,4,1,1,66

1,2,2,6,0,0,0,0,0,0,0,0,0,0,1,5

4,1,3,3,8,13,3,0,0,0,0,4,0,4,1,46

10,1,8,9,21,29,10,0,0,12,0,14,2,10,1,152

4,1,2,6,8,13,0,0,0,5,0,6,0,3,1,44

1,15,10,30,21,49,4,0,0,24,0,9,0,11,1,153

1,0,1,2,8,8,0,0,0,5,0,0,3,0,1,59

3,6,2,14,0,0,0,0,0,0,0,0,0,0,1,43

4,3,5,14,9,16,56,0,0,0,0,16,0,2,1,59

2,3,3,9,8,7,0,4,0,0,0,13,4,0,1,36

1,4,3,11,22,22,0,8,0,14,0,0,8,0,1,66

2,1,2,2,7,11,8,0,0,0,0,6,0,0,1,45

5,5,13,12,33,60,0,0,0,68,0,22,0,6,1,116

2,15,9,35,18,23,17,2,0,5,0,5,2,10,1,166

2,10,5,20,0,0,0,0,0,0,0,0,0,0,1,20

6,5,12,19,21,38,3,0,0,0,0,106,0,12,1,86

4,1,8,11,20,30,0,0,0,3,0,31,0,24,1,45

2,1,3,12,7,9,1,0,0,0,0,7,0,54,1,40

5,3,4,11,10,15,4,2,0,3,0,10,2,2,1,47

5,0,3,8,23,33,0,0,0,70,0,97,4,41,1,93

1,0,2,3,0,0,0,0,0,0,0,0,0,0,1,3

6,4,12,18,25,40,18,0,0,8,0,50,0,65,1,106

1,4,3,5,3,6,0,0,0,0,0,0,0,0,1,10

8,3,15,16,45,75,56,0,0,28,0,81,0,19,1,104

7,8,10,19,21,38,0,0,0,5,0,23,0,8,1,182

1,0,2,3,1,0,0,0,0,0,0,0,0,1,7

1,0,1,5,0,0,0,0,0,0,0,0,0,1,182

1,1,2,5,0,0,0,0,0,0,0,0,0,1,6

1,0,1,2,0,0,0,0,0,0,0,0,0,1,5

3,1,1,15,1,1,0,0,0,0,0,0,0,0,1,26

5,3,13,17,19,31,28,0,0,21,0,32,4,4,1,117

2,0,8,3,3,3,0,0,0,0,0,2,0,0,1,13