UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
MESTRADO EM ENGENHARIA ELÉTRICA

GISELE ALVES SANTANA

UMA ABORDAGEM PARA A IDENTIFICAÇÃO AUTOMÁTICA DE PROBLEMAS DE USABILIDADE EM INTERFACES DE SISTEMAS WEB ATRAVÉS DE RECONHECIMENTO DE PADRÕES

DISSERTAÇÃO

CORNÉLIO PROCÓPIO
2013

GISELE ALVES SANTANA

UMA ABORDAGEM PARA A IDENTIFICAÇÃO AUTOMÁTICA DE PROBLEMAS DE USABILIDADE EM INTERFACES DE SISTEMAS WEB ATRAVÉS DE RECONHECIMENTO DE PADRÕES

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica do Programa de Pós-Graduação em Engenharia Elétrica, Universidade Tecnológica Federal do Paraná. Área de concentração: Sistemas Eletrônicos Industriais.

Orientador: Prof. Dr. Luciano Tadeu Esteves Pansanato.

CORNÉLIO PROCÓPIO
2013

Dados Internacionais de Catalogação na Publicação

S232 Santana, Gisele Alves
Uma abordagem para a identificação automática de problemas de usabilidade em interfaces de sistemas web através de reconhecimento de padrões / Gisele Alves Santana. – 2013.
140 p. : il. ; 30 cm

Orientador: Prof. Dr. Luciano Tadeu Esteves Pansanato.
Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica. Cornélio Procópio, 2013.
Bibliografia: p. 113-120.

1. Sites da Web – Avaliação e classificação. 2. Compreensão de dados (Computação). 3. Projeto de sistema centrado no usuário. 4. Sistemas de reconhecimento de padrões. 5. Engenharia elétrica – Dissertações. I. Pansanato, Luciano Tadeu Esteves, orient. II. Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica. III. Título.

CDD (22. ed.) 621.3

Biblioteca da UTFPR, Câmpus Cornélio Procópio

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Diretoria de Pesquisa e Pós-Graduação
Programa de Pós-Graduação em Engenharia Elétrica
Mestrado em Engenharia Elétrica

TERMO DE APROVAÇÃO

UMA ABORDAGEM PARA A IDENTIFICAÇÃO AUTOMÁTICA DE PROBLEMAS DE USABILIDADE EM INTERFACES DE SISTEMAS WEB ATRAVÉS DE RECONHECIMENTO DE PADRÕES

por

Gisele Alves Santana

Esta Dissertação foi julgada adequada para obtenção do título de "Mestre em Engenharia Elétrica" e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná.

Cornélio Procópio, 11/04/ 2013.

_____
Alessandro Goedtel, Prof. Dr.
Coordenador do Curso

Banca Examinadora:

_____
Luciano Tadeu Esteves Pansanato, Prof. Dr.
Orientador

_____
Paulo Rogério Scalassara, Prof. Dr.

_____
Maria de Fátima Queiroz Vieira, Prof. Dr.

"A Folha de Aprovação assinada encontra-se na Coordenação do Curso do Programa".

Dedico este trabalho a minha família, por sempre me apoiar no decorrer deste curso.

AGRADECIMENTOS

A Deus por estar sempre me guiando e dando forças pra continuar.

Aos meus pais Nair Brancalhão Santana e Reinaldo Alves Santana, à minha irmã Juliana, à minha sobrinha Laura e ao meu namorado Rubinho pelo apoio, perseverança e compreensão.

Ao meu orientador, Prof. Dr. Luciano Tadeu Esteves Pansanato, pelos ensinamentos e grande orientação durante o decorrer do curso e no desenvolvimento deste trabalho.

A todos os professores do Programa de Pós-Graduação em Engenharia Elétrica da UTFPR – Câmpus Cornélio Procópio, em especial ao Prof. Dr. Paulo Rogério Scalassara, pelas discussões relacionadas ao desenvolvimento do trabalho.

Aos companheiros de curso, mesmo aqueles que não chegaram até o final.

A todos os professores com quem já tive aula, por todo o conhecimento passado.

A CAPES pelo apoio financeiro através da bolsa de estudo de demanda social.

Ao Campus Cornélio Procópio, pela liberação do espaço e ferramentas necessárias para o desenvolvimento e conclusão deste trabalho.

“A dúvida é o princípio da sabedoria”.
Aristóteles.

RESUMO

SANTANA, Gisele Alves. **Uma abordagem para a identificação automática de problemas de usabilidade em interfaces de sistemas web através de reconhecimento de padrões**. 2013. 141 f. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2013.

Recentemente, alguns sistemas estão sendo transferidos para a plataforma web. Muitos serviços e aplicações, incluindo sistemas de simulação e planejamento de energia e sistemas de automação, são desenvolvidos com interfaces baseadas na Internet. A usabilidade é a principal característica de uma interface e está associada com as funcionalidades de um sitema. Ela descreve o quão bem um produto pode ser utilizado para os fins propostos por seus usuários com eficácia, eficiência e satisfação. Este trabalho apresenta a aplicação de técnicas de Reconhecimento de Padrões na detecção e classificação automática de problemas de usabilidade na interface de um sistema web. O foco inicial do trabalho é centrado na identificação de possíveis problemas de usabilidade em formulários web. Os potenciais problemas de usabilidade do formulário web são definidos a partir das recomendações descritas na literatura. As tarefas realizadas pelo usuário são obtidas através da análise da interação do usuário armazenada em arquivos de *log*. A classificação de quais tarefas são realizadas conforme o esperado e quais são consideradas potenciais problemas de usabilidade é realizada através de uma Rede Neural Artificial.

Palavras-chave: Avaliação de Usabilidade. Análise de *Log*. Modelo de Tarefas. Reconhecimento de Padrões.

ABSTRACT

SANTANA, Gisele Alves. **An Approach for Automatic Identification of Usability Problems in Web System Interfaces using Pattern Recognition**. 2013. 141 f. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2013.

Recently, some systems have been transferred to the web-based platform. Many services and applications, including those of power systems simulating and planning and automation systems, are developed with Internet-based interface. Usability is mainly a characteristic of the interface and is associated with the functionalities of the systems. It describes how well a product can be used for its intended purpose by its users with efficiency, effectiveness and satisfaction. This paper presents the application of pattern recognition techniques in automatic detection and classification of usability problems in the interface of a web system. The initial focus of this work is focused on identifying potential usability problems in web forms. The potential usability problems of the web form are defined based on the recommendations described in the literature. The tasks performed by the user are obtained through analysis of user interaction stored in log files. The classification of tasks which are performed as expected and what are considered potential usability problems is performed by an Artificial Neural Network.

Keywords: Usability Evaluation. Log Analysis. Task Model. Pattern Recognition.

LISTA DE ILUSTRAÇÕES – FIGURAS E QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ACT	Análise Cognitiva das Tarefas
AJAX	*Asynchronous Javascript and XML*
AHT	Análise Hierárquica das Tarefas
AT	Análise de Tarefas
CLP	Controlador Lógico Programável
CSS	*Cascade Style Sheets*
CTT	*Concurrent Task Trees*
CTTE	*Concurrent Task Trees Environment*
GOMS	*Goals, Operators, Methods and Selection rules*
HTML	*Hypertext Markup Language*
HTTP	*HyperText Transfer Protocol*
ICO	*Interactive Cooperative Objects*
IHC	Interface Homem-Computador
ISO	*International Organization for Standardization*
IP	*Internet Protocol*
MAD	Método Analítico de Descrição de tarefas
MATLAB	*Matrix Laboratory*
MLP	*MultiLayer Perceptron*
MT	Modelo de Tarefas
PCA	*Principal Component Analysis*
PHP	*Hypertext Preprocessor*, ou originalmente *Personal Home Page*
RIA	*Rich Internet Applications*
RNA	Rede Neural Artificial
RP	Reconhecimento de Padrões
SGBD	Sistema Gerenciador de Banco de Dados

SQL	*Structured Query Language*
TCP	*Transmission Control Protocol*
TI	Tecnologia da Informação
UAN	*User Action Notation*
URL	*Uniform Resource Locator*
UTFPR	Universidade Tecnológica Federal do Paraná
WAUTT	*Web Application User Tracking Tool*

LISTA DE SÍMBOLOS

Rec	Conjunto de recursos oferecidos por uma página web
U	Usuário com acesso a um sistema web
l	Entrada / registro em um arquivo de *log*
h	Horário de acesso a uma página web
c	Código de status que identifica se houve sucesso ou fracasso na requisição
E	Conjunto de eventos capturados da interação com um sistema web
p	Página web
l_f	Entrada / registro em um arquivo de *log* filtrado
X	Vetor de características
a	Coeficientes dos componentes principais
$Pc1$	Primeiro componente principal
$Pc2$	Segundo componente principal
C	Matriz de covariância
x_m	Média
x^*	Valor normalizado
σ	Desvio padrão
R	Matriz de correlação
λ	Autovalor da matriz R
I	Matriz identidade
γ	Autovetor da matriz R
V	Porção da variância do autovetor
w	Vetor de pesos dos neurônios da RNA
x	Dados (amostras) de entrada da RNA
$f(.)$	Função de ativação
y	Valor da saída produzida pela RNA
Σ	Combinador linear
θ	Limar de ativação ou bias
u	Potencial de ativação
η	Taxa de aprendizado da RNA
d	Valor desejado de saída da RNA
b	Valor de inclinação da função de ativação

v	Campo local induzido
φ'	Diferenciação em relação ao argumento
δ	Gradientes locais
t	Tarefa
p_o	Página inicial de um sistema web
p_f	Página final de um sistema web
$temp$	Tempo gasto para execução da tarefa

SUMÁRIO

1 INTRODUÇÃO

A Internet tornou-se um meio conveniente para acesso às informações devido ao fato dos navegadores (*browsers*) serem capazes de integrar diferentes serviços de rede numa única e amigável interface com o usuário (SILVA et al., 2008). O uso de tecnologias web tem propiciado flexibilidade ao usuário de um sistema, seja no aspecto físico, pela capacidade de acesso remoto a partir de qualquer local distante geograficamente ou no aspecto tecnológico, pela capacidade de interação remota com as aplicações ou subsistemas presentes em uma indústria (FERNANDEZ et al., 2012; IDOUGHI et al., 2010).

Os sistemas de automação atuais podem possuir módulos voltados para a Internet. Através desses módulos, os navegadores podem disponibilizar interfaces gráficas nas quais o usuário interage remotamente com o processo industrial. Como exemplo, tem-se os trabalhos de Robles e Kim (2010), Idoughi et al. (2010), Yang et al. (2002), Lahti e Kankaanpaa (2011), Suresh et al. (2011), Fernandez et al. (2012), Yang e Yang (2005), entre outros.

Para o desenvolvimento de um sistema de automação com uma interface web, vários fatores devem ser considerados, como: fidelidade da informação, segurança de acesso, taxa de transferência de dados apropriada e adequada interação do usuário com o sistema, evitando, assim, que incidentes de operação ocorram e gerem prejuízos para o sistema como um todo (STOUFFER et al., 2006).

A interação do usuário com interfaces de sistemas de automação industrial apresenta um alto volume de informação, impondo uma grande carga cognitiva por parte do usuário destes sistemas (TURNELL et al., 2004). Além disso, o usuário deve reagir aos eventos e completar tarefas levando em consideração o prazo final e as restrições de segurança (TURNELL; FARIAS, 1996; IDOUGHI et al., 2010). Portanto, a qualidade do projeto de interface do ponto de vista da navegação do usuário pode comprometer a eficiência de execução da tarefa, o desempenho global do sistema e aspectos de segurança (VIEIRA et al., 2005). Assim, uma interface bem projetada pode reduzir a ocorrência de erros por parte do usuário do sistema.

O principal fator de qualidade em uma interface é a sua usabilidade, que estabelece o quanto os sistemas são projetados de forma a serem fáceis de utilizar e aprender (NIELSEN, 1993; ROBLES; KIM, 2010; LAHTI; KANKAANPAA, 2011). Deve-se pensar na usabilidade em termos de qualidade de uso que um sistema interativo apresenta para seus usuários, de modo que possam atingir um conjunto específico de objetivos e tarefas em um

ambiente particular de trabalho (INTERNATIONAL..., 1998). A usabilidade é importante sob a perspectiva do usuário, pois afeta diretamente a sua produtividade, seu desempenho e a sua carga de trabalho (NIELSEN, 1993). A aplicação de conceitos, metodologias e técnicas relativas à avaliação da usabilidade para o contexto do desenvolvimento de sistemas web é uma tarefa bastante difícil, uma vez que testes tradicionais de usabilidade, com base no uso de avaliadores especialistas em ambientes controlados, possuem desvantagens em relação ao custo e tempo (DIAS, 2007; PREECE, et al., 2012).

Para aprimorar esse cenário são conduzidos testes remotos e automáticos de usabilidade (NIELSEN; MACK, 1994; IVORY; HEARST, 2001; PALANQUE et al., 2011). Nesse tipo de avaliação os usuários e avaliadores estão separados no tempo e espaço, sendo desnecessário realizar a avaliação em tempo real ou no mesmo ambiente dos usuários, além de envolver uma grande quantidade de usuários.

A análise de *log* é uma das técnicas mais utilizadas para realizar avaliações remotas de usabilidade (NIELSEN; MACK, 1994; IVORY; HEARST, 2001; PALANQUE et al., 2011; VARGAS et al., 2011; JANSEN, et al., 2009). Nessa técnica, são analisadas as interações do usuário registradas em arquivos de *log* gerados durante a utilização de um sistema (DIAS, 2007; PREECE, et al., 2012). No contexto de sistemas web, geralmente são utilizados os arquivos de *log* do servidor web. No entanto, esses arquivos de *log* fornecem apenas as páginas (URLs) acessadas pelo usuário, não oferecendo informações detalhadas sobre a interação real do usuário com um sistema web.

Uma proposta para a identificação de problemas de usabilidade em interfaces de sistemas web consiste na análise da interação do usuário com base em um modelo de tarefas pré-estabelecido (PATERNÒ, 2001; AQUINO et al., 2011; DIAPER, 2011). Essa técnica é realizada através da comparação entre a sequência realizada pelo usuário na execução de uma tarefa e a sequência esperada, definida no modelo de tarefas.

Este trabalho apresenta uma abordagem para a identificação automática de problemas de usabilidade em interfaces de sistemas web através de técnicas de reconhecimento de padrões. Reconhecimento de Padrões (RP) é a área de pesquisa que tem por objetivo a classificação de objetos (padrões) em um número de categorias (classes) (DUDA et al., 2001). As tarefas realizadas pelo usuário são obtidas através da análise das informações relativas à interação do usuário armazenadas em arquivos de *log*. O modelo de tarefas desenvolvido, ao invés de considerar somente as páginas (URLs) acessadas pelo usuário, representa as sequências necessárias para a realização de uma tarefa com base em eventos realizados pelo usuário em elementos de interação que estão contidos numa página

web. O foco inicial do trabalho está centrado na identificação de possíveis problemas de usabilidade em formulários web. Os potenciais problemas de usabilidade com formulários web são definidos a partir das recomendações descritas por Wroblewski (2008) e Shneiderman e Plaisant (2010).

Na abordagem proposta, o registro das interações do usuário com um sistema web é realizado com o apoio da ferramenta *Web Application User Tracking Tool* (WAUTT) (RIVOLLI et al., 2008), que faz o rastreamento não somente das páginas visitadas pelo usuário, mas também dos eventos realizados em cada elemento contido nas páginas de um sistema web. A classificação das tarefas que são realizadas conforme o esperado e das que são consideradas potenciais problemas de usabilidade é realizada através de uma Rede Neural Artificial (HAYKIN, 2001).

A Figura 1 ilustra o contexto de funcionamento da metodologia proposta neste trabalho.

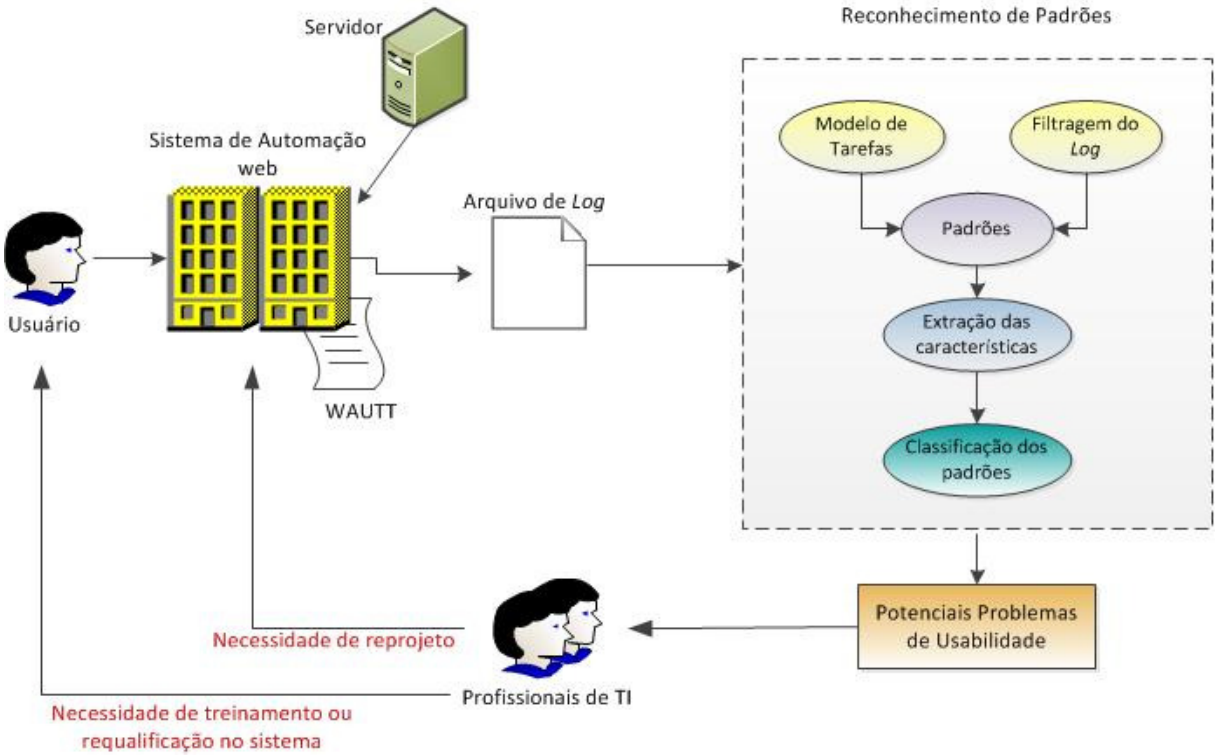

Figura 1 – Contexto de funcionamento da metodologia proposta
Fonte: Autoria própria.

Na figura 1, um usuário acessa um sistema de automação web que é rastreado pela ferramenta WAUTT. Essa ferramenta realiza a captura da interação entre o usuário e o sistema e armazena as informações em arquivos de *log*. A partir desses arquivos de *log* são aplicadas técnicas de reconhecimento de padrões, incluindo a definição de um modelo de tarefas e a filtragem dos arquivos de *log*, com o objetivo de identificar padrões que representam potenciais problemas de usabilidade. De posse da identificação e classificação desses problemas, os profissionais de Tecnologia da Informação (TI) de uma indústria podem analisar se existe necessidade de reprojeto da interface do sistema ou treinamento do usuário.

1.1 JUSTIFICATIVA

A integração do TCP/IP no *hardware* dos Controladores Lógicos Programáveis (CLPs) estimula e abre a possibilidade do uso de um sistema de automação via Internet (TURNELL; FARIAS, 1996; SURESH et al., 2011). A maioria desses sistemas é considerada crítica no tempo, onde a interface com o usuário desempenha um papel significativo no desempenho global de operação do sistema (HUSSAK; YAUG, 2004; TURNELL et al., 2004; GUERRERO et al., 2008; ROBLES; KIM, 2010).

Diante da constatação de que muitos incidentes nas indústrias ocorrem devido ao erro humano, a análise do contexto de ocorrência contribui para a revisão da realização da tarefa por parte do usuário e particularmente para a revisão do projeto da interface (TURNELL et al., 2004; GUERRERO, et al., 2004; NASCIMENTO NETO et al., 2009).

A interface de um sistema é o meio pelo qual o diálogo entre o programa e o ser humano é estabelecido (PRESSMAN, 2010). Quando fatores humanos são considerados, esse diálogo acontece em harmonia. Assim, a interface deve ser fácil de aprender, simples de usar, direta e amigável. A negligência dessas características pode implicar em possíveis problemas de usabilidade e provocar a ocorrência de erro humano.

Os problemas de usabilidade ocorrem quando um usuário encontra dificuldades para realizar uma tarefa com uma interface. Essas dificuldades podem ter origens variadas e ocasionar perda de dados, diminuição da produtividade, podendo chegar à rejeição total do sistema por parte dos usuários (NIELSEN, 1993). O acesso rápido à informação também depende da qualidade do projeto da interface. Uma interface mal projetada e sem clareza na

apresentação das informações pode resultar na interpretação errônea destas informações, ocasionando decisões equivocadas do usuário.

As respostas do usuário devem respeitar as restrições temporais do sistema associadas à execução das tarefas (FIELDS et al., 1995; TURNELL et al., 2004; GUERRERO, et al., 2004). Um exemplo típico dessas restrições temporais em sistemas de automação é o tempo disponível para o usuário reconhecer e atuar sobre condições de alarme do sistema. Se as restrições de tempo não forem respeitadas podem ocorrer situações catastróficas, que envolvem desde a perda de produção à quebra de máquinas e até risco de morte para as pessoas envolvidas no processo.

A produtividade de um sistema está diretamente ligada à qualidade da interface, que também depende da qualidade de interação entre usuário e sistema (IDOUGHI et al., 2010; ROBLES; KIM, 2010). Assim, as interfaces devem ser projetadas com o objetivo de prover ao usuário caminhos de interação eficientes. Desse modo, é notória a necessidade de sistemas de automação web fáceis de serem compreendidos e bem projetados, ou seja, desenvolvidos com a intenção não somente do bom funcionamento do sistema, mas também com uma boa usabilidade.

Considerando essas afirmações, justifica-se o desenvolvimento de uma abordagem capaz de identificar e classificar automaticamente potenciais problemas de usabilidade em interfaces de sistemas web a partir da interação do usuário.

A abordagem proposta emprega técnicas para a avaliação remota da usabilidade. Essas técnicas apresentam baixo custo operacional, uma vez que este é causado principalmente pelo deslocamento do usuário e necessidade de disposição de equipamentos e supervisores dedicados no local do teste. Além disso, a coleta dos dados é realizada sem interferência na qualidade de interação do usuário com o sistema.

1.2 OBJETIVOS

Os objetivos que conduzem a pesquisa deste trabalho são divididos em objetivo geral e objetivos específicos.

1.2.1 Objetivo Geral

O escopo principal deste trabalho é a identificação e classificação automática de problemas de usabilidade em interfaces de sistemas web. Pretende-se tornar as avaliações de usabilidade abrangentes, rápidas e com baixo custo, além de transparentes aos usuários.

O objetivo geral deste trabalho é desenvolver uma abordagem que emprega técnicas de Reconhecimento de Padrões para a identificação e classificação automática de potenciais problemas de usabilidade na interface de um sistema web. O processo é realizado através da análise de arquivos de *log*, utilizando como base um modelo de tarefas. O modelo de tarefas é definido a partir dos eventos realizados pelo usuário em elementos de interação contidos numa página web. O foco deste trabalho baseia-se, inicialmente, na análise de tarefas específicas relacionadas a formulários web.

1.2.2 Objetivos Específicos

Ao objetivo geral deste trabalho estão relacionados os seguintes objetivos específicos:

- Analisar e identificar as principais dificuldades relacionadas à interação do usuário com um sistema web, especificamente com formulários;
- Realizar um estudo sobre a classificação de problemas de usabilidade em formulários;
- Desenvolver um sistema web para a realização de testes;
- Estabelecer um ambiente informatizado capaz de registrar as informações da interação com o sistema web;
- Desenvolver um Modelo de Tarefas, que se trata de uma descrição lógica das atividades a serem executadas pelo usuário para alcançar seus objetivos na interação com um sistema; neste caso específico, descrever os eventos necessários para a realização de uma tarefa relacionada à interação do usuário com um formulário;

- Selecionar os dados (características) dos arquivos de *log* necessários para identificação de potenciais problemas de usabilidade;
- Desenvolver uma metodologia para filtragem dos arquivos de *log* com o objetivo de torná-los consistentes e sem dados inexpressíveis que possam influenciar erroneamente na identificação e classificação de problemas de usabilidade;
- Mensurar e normalizar os dados dos arquivos de *log* filtrados para a aplicação de técnicas de reconhecimento de padrões;
- Desenvolver uma metodologia para extração de características dos arquivos de *log* com o objetivo de reduzir a dimensionalidade do conjunto de dados original e consequentemente aumentar o desempenho do classificador;
- Implementar e testar um extrator de características baseado na Análise de Componentes Principais;
- Implementar e testar diferentes arquiteturas de Redes Neurais Artificiais (RNAs) para classificação dos potenciais problemas de usabilidade;
- Interpretar os resultados das RNAs identificando as arquiteturas com melhor desempenho.

1.3 ORGANIZAÇÃO DOS CAPÍTULOS

Este trabalho está organizado da seguinte maneira:

Inicialmente, no Capítulo 1 são apresentados conceitos relacionados à avaliação da usabilidade de sistemas interativos, mais especificamente de sistemas web, além dos objetivos que este trabalho busca realizar visando solucionar os problemas apontados.

No Capítulo 2 são apresentados os trabalhos relacionados, identificando algumas abordagens atuais ao problema de avaliação remota de usabilidade através da análise de arquivos de *log*.

No Capítulo 3 são apresentados conceitos sobre usabilidade, avaliação da usabilidade, análise de *log*, assim como a usabilidade em formulários web, visando descrever as principais técnicas que apoiam a avaliação de sistemas interativos.

No Capítulo 4 é descrita a forma de captura das informações sobre a interação entre o usuário e o sistema. É apresentada a ferramenta WAUTT e o formato dos arquivos de *log* gerados por esta ferramenta.

No Capítulo 5 são apresentados conceitos relacionados ao Modelo de Tarefas, assim como alguns tipos de modelos.

No Capítulo 6 é apresentada a técnica de Reconhecimento de Padrões, descrevendo as etapas necessárias para a sua aplicação. É apresentada a técnica de Análise de Componentes Principais, utilizada para a extração de características. Conceitos e definições matemáticas sobre Redes Neurais Artificiais também são apresentadas.

No Capítulo 7 são apresentados os resultados da aplicação da abordagem proposta por este trabalho, assim como a análise dos dados obtidos.

No Capítulo 8 são apresentadas as conclusões e trabalhos futuros.

2 TRABALHOS RELACIONADOS

Muitas pesquisas estão sendo desenvolvidas na área de avaliação remota e automática de usabilidade. Alguns fatores encorajam esse tipo de avaliação (IVORY; HEARST, 2001), como: aumento da consistência dos erros descobertos principalmente na utilização de modelo de tarefas; aumento da cobertura de características avaliadas e possibilidade de comparação entre interfaces projetadas; redução do custo da avaliação de usabilidade em decorrência da redução do tempo de avaliação através de algoritmos e ferramentas; redução da necessidade de avaliação por um especialista.

Para a avaliação remota da usabilidade, os trabalhos mais recentes utilizam análise de *log* com a finalidade de obter detalhes sobre os eventos gerados pelas ações do usuário durante a interação. As abordagens empregadas pelas ferramentas WAUTER (BALBO et al., 2005), AWUSA (TIEDTKE et al., 2002), WELFIT (SANTANA; BARANAUSKAS, 2010), WebHint (VARGAS, et al., 2010), WebRemUSINE (PAGANELLI; PATERNÒ, 2002) e Ergo-Monitor (MORANDINI et al., 2007) realizam esse modo de coleta e análise de dados.

A ferramenta WAUTER analisa os arquivos de *log* baseando-se em tarefas, porém necessita da instalação de um *software* no computador do cliente, impedindo o uso transparente da aplicação por parte do usuário. A ferramenta utiliza uma notação para a definição de tarefas e heurísticas pré-definidas para comparação entre as ações realizadas pelos usuários e o modelo de tarefas do sistema web. Entretanto, essa ferramenta não oferece uma forma intuitiva para definição de tarefas e possui limitações para a definição de possíveis caminhos de uma tarefa em sistemas web dinâmicos.

A ferramenta AWUSA também compara sequências de eventos para encontrar diferenças entre a interação ou as ações do usuário e uma tarefa definida pelo avaliador. No entanto, utiliza arquivos de *log* de servidor, que não possuem granularidade fina em relação às ações dos usuários. Para definir as sequências de eventos como referência para comparação com as ações dos usuários, a AWUSA utiliza páginas e *links* identificando-os a partir da análise do código HTML (*Hypertext Markup Language*). Entretanto, essa abordagem dificulta a associação de diversos caminhos alternativos e uma única tarefa.

As ferramentas WELFIT e WebHint não capturam atributos como folhas de estilo (*CSS*[1]) e outros atributos HTML, o que impossibilita a análise de heurísticas em termos

[1] CSS é uma linguagem para estilos que define o *layout* de documentos HTML. Por exemplo, CSS controla fontes, cores, margens, linhas, alturas, larguras, imagens de fundo, posicionamentos, entre outros.

visuais dos elementos das páginas. Essas ferramentas utilizam métodos para análise dos dados de *log* capturados. A WELFIT utiliza um algoritmo de grafos para identificar padrões nas ações do usuário em uma página individualmente, mas não considera o caminho entre as páginas de um sistema web. A ferramenta WebHint não utiliza notações para a definição de tarefas, permitindo que uma tarefa seja definida apenas navegando na interface.

A ferramenta WebRemUSINE realiza a avaliação remota da usabilidade capturando e analisando de forma automática a interação em sistemas web para a detecção de problemas de usabilidade. A abordagem é baseada na comparação entre um modelo de tarefas e os caminhos realizados pelos usuários. Para definir o modelo de tarefas, o avaliador deve utilizar um editor de tarefas, bem como uma tabela específica de mapeamento entre as entradas dos arquivos de *log* e o modelo de tarefas. As desvantagens em relação ao uso desta ferramenta são: necessidade da instalação de um *plugin*[2] Java para salvar os arquivos de *log* referentes às interações dos usuários em um servidor; o usuário deve selecionar as tarefas que está realizando para que os eventos sejam associados à tarefa; a definição de tarefas utiliza notações, o que força o avaliador a conhecê-las para definir as tarefas em uma interface; e a redução do espaço útil da tela para exibir as tarefas disponíveis.

O ambiente Ergo-Monitor é um sistema de monitoramento da usabilidade de sistemas web por meio da coleta seletiva da interação e da análise de arquivos de *log* referentes às interações estabelecidas entre usuários e um sistema web. Com base em dados selecionados, o sistema calcula por meio de modelos de comportamentos esperados para tarefas específicas, as métricas que visam quantificar a usabilidade do sistema web. O Ergo-Monitor realiza todos esses procedimentos de forma invisível e imperceptível ao usuário. Assim, o usuário interage normalmente com o sistema web e os dados relativos às suas interações são armazenados, sendo posteriormente analisados. Esses dados servem como base na elaboração de medidas significativas para uma análise da usabilidade do sistema web. A principal desvantagem relacionada a esse ambiente é a construção do modelo de tarefas baseado exclusivamente em URLs, o que restringe a identificação de problemas específicos de usabilidade em interfaces de sistemas web.

A abordagem proposta neste trabalho é semelhante a do ambiente Ergo-Monitor em termos de filtragem de *log* e desenvolvimento de um modelo de tarefas. Porém, a análise dos arquivos de *log* para a definição do modelo de tarefas deste trabalho é realizada considerando os eventos realizados em cada elemento da página web e não apenas com base nas páginas

[2] *Plugin* é a denominação utilizada para designar componentes que são adicionados aos *browsers* incorporando capacidade de manipulação de dados em outros formatos além do HTML.

visitadas (URLs). Além disso, são empregadas técnicas de reconhecimento de padrões para classificação automática de problemas de usabilidade.

A principal diferença da abordagem proposta neste trabalho em relação às outras ferramentas apresentadas refere-se ao fato de que os problemas de usabilidade são encontrados e classificados automaticamente em padrões distintos, através de técnicas de reconhecimento de padrões. Os padrões são baseados em diretrizes de usabilidade para o projeto de interfaces, com foco inicial centrado em formulários. As demais ferramentas oferecem medidas relacionadas à eficiência e eficácia de um sistema web, porém não classificam possíveis problemas de usabilidade em interfaces de sistemas web de forma automática.

3 USABILIDADE

Usabilidade é a capacidade de um produto ser usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso (INTERNATIONAL..., 1998). O termo usabilidade começou a ser utilizado no início da década de 80 como um substituto da expressão *"user-friendly"*, a qual era considerada vaga e excessivamente subjetiva (DIAS, 2007). Na verdade, não é necessário que as máquinas sejam amigáveis, basta que elas não interfiram nas tarefas realizadas pelo usuário. Além disso, usuários diferentes têm necessidades diferentes, da maneira que um sistema pode ser amigável para um usuário e não tão amigável para outro.

Vários autores utilizam abordagens diferentes para definir a usabilidade, como Dias (2007):

- Definições orientadas ao produto – associadas às características ergonômicas do produto;
- Definições orientadas ao usuário – relacionadas ao esforço mental ou atitude do usuário frente ao produto;
- Definições baseados no desempenho do usuário – associadas à forma de interação do usuário, com ênfase na facilidade de uso e no grau de aceitação do produto;
- Definições orientadas ao contexto de uso – relacionadas às tarefas específicas realizadas por usuários específicos do produto, em determinado ambiente de trabalho.

A primeira norma que definiu o termo usabilidade versou sobre qualidade de *software* (INTERNATIONAL..., 1991). Sua abordagem é claramente orientada ao produto e ao usuário, pois considera a usabilidade como um conjunto de atributos de *software* relacionado ao esforço necessário para seu uso e para o julgamento individual de tal uso por determinado conjunto de usuários.

A partir dessa norma, o termo usabilidade ultrapassou os limites do ambiente acadêmico passando a fazer parte do vocabulário técnico de várias áreas do conhecimento. O conceito de usabilidade evoluiu e foi redefinido na parte 1 da norma em 2000 (INTERNATIONAL..., 2000), incluindo as necessidades do usuário. Essa norma define ainda outras características de qualidade de *software*, como:

- Funcionalidade – capacidade do *software* de prover funções que atendem necessidades expressas e implícitas, quando usado nas condições especificadas.
- Confiabilidade – capacidade do *software* de manter seu nível de desempenho, quando usado nas condições especificadas.
- Usabilidade – capacidade do *software* de ser compreendido, aprendido, usado e apreciado pelo usuário, quando usado nas condições especificadas.
- Eficiência – capacidade do *software* de operar no nível de desempenho requerido, em relação à quantidade de recursos empregados, quando usado nas condições especificadas.
- Possibilidade de manutenção – capacidade do *software* de ser modificado. Modificações podem abranger correções, melhorias ou adaptações do *software*, mudanças de ambiente ou nas especificações funcionais e de requisitos.
- Portabilidade – capacidade do *software* de ser transferido de um ambiente para outros.

Considerando mais o ponto de vista do usuário e seu contexto de uso do que as características ergonômicas do produto, a norma definiu a usabilidade como a capacidade de um produto ser usado por usuários específicos para atingir objetivos com eficácia, eficiência e satisfação em um contexto específico de uso (INTERNATIONAL..., 1998).

A norma também esclarece outros conceitos, visualizados na Figura 2, onde o usuário é a pessoa que interage com o produto. O contexto de uso abrange usuários, tarefas, equipamentos, ambiente físico e social em que o produto é usado. A eficácia está relacionada com a precisão e a completeza com que os usuários atingem objetivos específicos, acessando a informação correta ou gerando os resultados esperados. A precisão é uma característica associada à correspondência entre a qualidade do resultado e o critério especificado, enquanto que a completeza é a proporção da quantidade-alvo que foi atingida.

A eficiência refere-se à precisão e completeza com que os usuários atingem seus objetivos em relação à quantidade de recursos gastos. A satisfação está relacionada com o conforto e a aceitabilidade do produto, medidos por meio de métodos subjetivos e/ou objetivos. As medidas objetivas de satisfação podem se basear na observação do comportamento do usuário ou no monitoramento de suas respostas fisiológicas. As medidas subjetivas são produzidas pela quantificação das reações, atitudes e opiniões expressas subjetivamente pelos usuários.

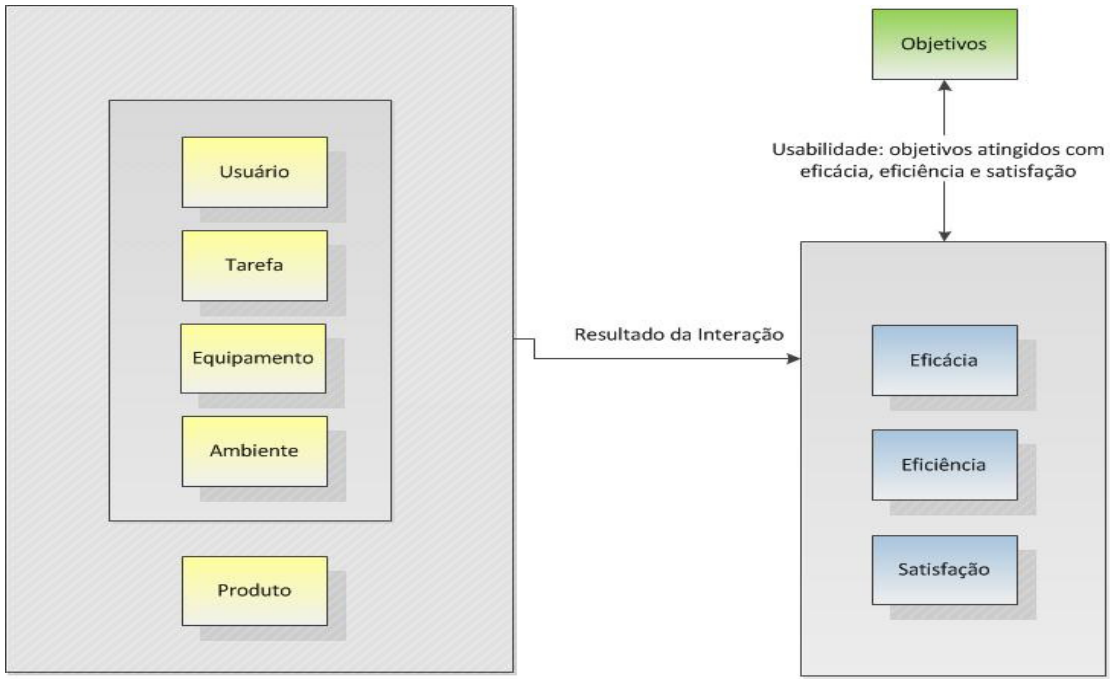

Figura 2 – Esquema do conceito de usabilidade
Fonte: Adaptado de Dias (2007, p. 37).

Um sistema interativo é considerado eficaz quando possibilita que os usuários atinjam seus objetivos. A eficácia é a principal motivação que leva um usuário a utilizar um sistema. Se um sistema é fácil de usar, aprender e é agradável ao usuário, mas não consegue atender a objetivos específicos de usuários específicos, este não será usado, mesmo que seja oferecido gratuitamente (DIAS, 2007). O segundo elemento da usabilidade é a eficiência, definida quantitativamente pelo tempo de resposta, tempo total para realizar uma tarefa específica ou a quantidade de erros. O terceiro elemento, a satisfação do usuário, refere-se a percepções, sentimentos e opiniões dos usuários a respeito de um sistema, normalmente mapeados a partir de questionamentos escritos ou orais realizados com os próprios usuários.

A usabilidade pode ser considerada uma qualidade de uso, isto é, qualidade de interação entre usuário e sistema, que depende das características tanto do sistema quanto do usuário. Além disso, a usabilidade também depende das tarefas específicas que os usuários realizam com o sistema, assim como do ambiente físico. Portanto, usabilidade é a qualidade de uso de um sistema, diretamente associada ao seu contexto operacional e aos diferentes tipos de usuários, tarefas, ambientes físicos e organizacionais.

Alguns aspectos de eficácia e eficiência da norma são reunidos em outra medida de usabilidade: o desempenho do usuário, que é descrito como a habilidade de realizar tarefas para as quais o sistema foi desenvolvido, no contexto em que o sistema será utilizado (DIAS, 2007).

A implementação de um sistema revela a opinião de seus projetistas sobre a aceitabilidade social e prática. Conforme Nielsen (1993), a aceitabilidade social dá maior ênfase às escolhas éticas e morais dos programadores. A aceitabilidade significa custo, compatibilidade, confiabilidade e utilidade do sistema. Assim, para Nielsen (1993), a usabilidade possui muitos componentes e geralmente está associada aos seguintes atributos:

- Facilidade de aprendizado – o sistema deve ser fácil de aprender de tal forma que o usuário consiga rapidamente explorá-lo e realizar suas tarefas.
- Eficiência de uso – o sistema deve ser eficiente a tal ponto que permita ao usuário, tendo aprendido a interagir com este sistema, atingir altos níveis de produtividade na realização de suas tarefas.
- Facilidade de memorização – após certo período sem utilizar o sistema, o usuário não frequente é capaz de retornar ao sistema e realizar suas tarefas sem a necessidade de reaprender como interagir com este.
- Baixa taxa de erros – em um sistema com baixa taxa de erros, o usuário é capaz de realizar tarefas sem maiores transtornos, recuperando erros, caso estes ocorram.
- Satisfação subjetiva – o usuário considera agradável a interação com o sistema e se sente subjetivamente satisfeito com este.

Em última análise, a usabilidade pode ser considerada o meio pelo qual atributos e critérios específicos são empregados, visando proporcionar ao usuário condições de realizar suas tarefas interativas com facilidade de uso, eficiência, eficácia e de forma intuitiva (ROBLES; KIM, 2010). Uma interface com usabilidade proporciona a diminuição da sobrecarga cognitiva[3] do usuário e promove condições que facilitam a retenção de informações ou aprendizagem do sistema por parte do usuário (CUMMINGS et al., 2010).

[3] Carga cognitiva é um fator presente no projeto de interfaces, visto que cada um dos elementos ou objetos da interface deve ser interpretado pelo usuário e consequentemente ocupa alguma energia mental deste. Um *design* de interface complexo que usa diferentes fontes, objetos, ferramentas de navegação e padrões de *layout* terá geralmente uma sobrecarga cognitiva porque cada componente necessitará ser percebido e interpretado pelo usuário.

3.1 INTERAÇÃO E INTERFACE

Interação é o processo de comunicação entre usuários e sistemas interativos (PREECE et al., 2012). Nesse processo, usuário e sistema trocam informações onde um espera e o outro envia alguma informação, em seguida interpreta-se e realiza-se uma ação e assim vice-versa constantemente. Esse processo é estudado principalmente do ponto de vista do usuário: quais as ações que ele realiza usando a interface e suas interpretações das respostas transmitidas pelo sistema. O processo de interação ocorre através da interface, que engloba tanto *software* quanto *hardware*.

Considerando a interação como um processo de comunicação, a interface pode ser vista como o sistema de comunicação utilizado neste processo (OLIVEIRA NETTO, 2004). A interface pode ser entendida como um componente de um sistema computacional com a qual um usuário entra em contato físico, perceptivo ou conceitual (PREECE et al., 2012).

A dimensão física de uma interface é composta por todos os elementos de interface que podem ser manipulados pelo usuário, enquanto que a dimensão perceptiva engloba aqueles que são percebidos pelo usuário. Por sua vez, os processos de interpretação e raciocínio do usuário são resultados da dimensão conceitual, desencadeados pela interação com o sistema, com base em características físicas e cognitivas, objetivos e ambiente de trabalho.

Antes de declarar um *software* pronto para uso é importante saber se este apóia adequadamente o usuário em suas tarefas no ambiente em que será utilizado. Assim como testes de funcionalidade são necessários para se verificar a robustez da implementação, a avaliação da interface é necessária para se analisar a qualidade de uso de um sistema; quanto mais cedo detectados os problemas de interação ou de interface, menor o custo para consertá-los (KARAT, 1993; JOHNSON et al., 2010).

Alguns dos principais objetivos da avaliação de sistemas interativos são (PREECE et al., 2012):

- Identificar as necessidades de usuários ou verificar o entendimento dos projetistas sobre estas necessidades;
- Identificar problemas de interação ou de interface;
- Investigar como uma interface afeta a forma de trabalhar dos usuários;
- Comparar alternativas de projeto de interface;

- Alcançar objetivos quantificáveis em métricas de usabilidade;
- Verificar conformidade com um padrão ou conjunto de heurísticas.

Uma interface com pouca interação possivelmente causará rendimento do usuário abaixo do esperado, diminuição na produção total e custo elevado.

3.2 AVALIAÇÃO DE USABILIDADE

Avaliação é o processo sistemático de coleta de dados responsável por obter informações sobre o modo como um determinado usuário deve utilizar um sistema para uma determinada tarefa num determinado ambiente (PREECE et al., 2012).

Alguns objetivos a serem atingidos pelas avaliações são: constatar, observar e registrar problemas durante a interação; calcular métricas objetivas para eficácia, eficiência e produtividade do usuário na interação com o sistema; diagnosticar as características do projeto que provavelmente atrapalhem a interação; e prever dificuldades de aprendizado na operação do sistema (CYBIS et al., 2010).

Os obstáculos que impedem o usuário de realizar suas tarefas, como problemas de visualização e até mesmo o acesso a determinados conteúdos, são considerados problemas de usabilidade. Nesse contexto, um problema de usabilidade é uma interferência na capacidade do usuário em completar suas tarefas de forma efetiva e eficiente (INTERNATIONAL..., 1998).

Geralmente os problemas de usabilidade são classificados como uma barreira ou obstáculo, de acordo com suas conseqüências na interação do usuário com o sistema. Diz-se que o problema é uma barreira quando o usuário esbarra sucessivas vezes e não aprende a suplantá-lo (CYBIS et al., 2010), e ainda quando impede o cumprimento da tarefa desejada pelo usuário ou compromete seu desempenho.

A avaliação de usabilidade pode ser realizada em qualquer fase do desenvolvimento de sistemas interativos; deve verificar o desempenho (eficiência e eficácia) da interação usuário/sistema e obter indícios do nível de satisfação do usuário, identificando problemas de usabilidade durante a realização de tarefas específicas em seu contexto de uso (NIELSEN; MACK, 1994).

A finalidade de métodos de avaliação de usabilidade é identificar e diagnosticar problemas de usabilidade. O conjunto de métodos de avaliação existentes na literatura pode ser subdividido em três grandes grupos: métodos de inspeção, métodos baseados em modelos e métodos de teste com usuários (DIAS, 2007).

Os métodos de inspeção, também conhecidos como métodos analíticos ou de prognóstico, caracterizam-se pela não participação direta dos usuários do sistema na avaliação. Os avaliadores são especialistas em usabilidade ou desenvolvedores de sistema e se baseiam em regras, recomendações, princípios e/ou conceitos previamente estabelecidos para identificar problemas de usabilidade relacionados à interação dos usuários reais com o sistema.

Os métodos baseados em modelos, também chamados de modelagem analítica, têm como objetivo prever a usabilidade de um sistema a partir de modelos ou representações da interface e/ou de seus usuários. Esses métodos pretendem representar como os usuários interagem com um sistema, isto é, modelam aspectos do entendimento, conhecimento, intenções ou reações dos usuários.

Os métodos de teste com usuários caracterizam-se pela participação direta dos usuários do sistema no processo de avaliação. Esses métodos podem ser prospectivos, como questionários e entrevistas, ou empíricos, ao adotar técnicas de observação ou monitoramento do uso do sistema em situações reais.

Deve-se também considerar aspectos relacionados à realização das tarefas do usuário em uma avaliação de usabilidade (DIAS, 2007). Uma tarefa é realizada visando atingir um objetivo. As tarefas possuem diferentes níveis de abstração, variando desde tarefas de alto nível (preenchimento de um formulário) a tarefas de baixo nível (seleção de um botão na tela). Em ambos os casos, deve-se pensar nas tarefas que podem ou não ser realizadas na interface e em sua ordem temporal.

No caso de tarefas de alto nível, as relações temporais são determinadas pelas dependências lógicas entre as tarefas. No caso de tarefas de baixo nível, as relações temporais podem depender também das restrições apresentadas pela implementação da interface (por exemplo, o usuário deve preencher os campos obrigatórios antes de selecionar um botão de confirmação de um cadastro) (LACEROF; PATERNÒ, 1998).

Uma proposta que tem se mostrado eficaz na avaliação de problemas de usabilidade consiste na análise dos arquivos de *log* com dados da interação do usuário com um sistema com base em um modelo de tarefas pré-estabelecido (PATERNÒ, 2001). Essa técnica é

realizada através da comparação entre a sequência de eventos realizada pelo usuário na execução de uma tarefa e a sequência de eventos esperada, definida em um modelo de tarefas.

3.2.1 Avaliação de Usabilidade na Web

Dentro do contexto de avaliações de usabilidade de sistemas web, pode-se afirmar que um sistema web é um *software* interativo, pois interage com pelo menos dois tipos de usuários: os usuários finais que tentam alcançar alguma meta na execução de uma tarefa, e os desenvolvedores que se esforçam para manter o funcionamento do sistema (NIELSEN, 2000).

O emprego de técnicas relativas à avaliação da usabilidade para o desenvolvimento de sistemas web é uma tarefa difícil, pois testes tradicionais possuem desvantagens em relação ao custo e tempo. Para aprimorar este cenário são conduzidos testes remotos e automáticos de usabilidade, permitindo envolver uma grande quantidade de usuários com baixo custo (NIELSEN; MACK, 1994). Uma das técnicas mais utilizadas para realizar avaliações remotas de usabilidade é a análise de *log*.

3.2.2 Análise de *Log*

A análise de *log* é uma técnica na qual são analisadas as interações do usuário registradas em arquivos de *log* gerados durante a utilização de um sistema (PREECE et al., 2012; IVORY; HEARST, 2001; PALANQUE et al., 2011; VARGAS et al., 2011; JANSEN et al., 2009). É um método de avaliação que, além de apontar problemas de usabilidade, pode ajudar a compreender o comportamento dos usuários em relação à interface do sistema.

Uma característica que merece destaque é que a análise de *log* é uma técnica que não interfere na interação do usuário com as tarefas executadas no sistema. Outro ponto de destaque é o fato de que as informações obtidas a partir da aplicação dessa técnica geram dados estatísticos confiáveis, relativos a várias questões, como: padrões de uso, usabilidade de produtos, estratégias de integração e utilidade percebida de produtos que envolvam o uso de sistemas computacionais (PREECE et al., 2012).

A análise de *log* pode ser considerada um método de teste com usuários para a avaliação da usabilidade de um sistema, pois se caracteriza pela participação direta do usuário no processo de avaliação. No entanto, de uma forma empírica, pois são adotadas técnicas de observação ou monitoramento do uso do sistema em situações reais de uma forma remota.

Como mencionado anteriormente, uma das técnicas mais utilizadas para realizar avaliações remotas de usabilidade de sistemas web é a análise de arquivos de *log* do servidor web (IVORY; HEARST, 2001). No entanto, o uso desses arquivos de *log* como fonte para a coleta dos dados relativos às interações fornecem, em sua maioria, apenas as páginas (URLs) visitadas pelo usuário, não trazendo informações detalhadas sobre a interação real do usuário com um sistema web.

Assim, este trabalho utiliza a ferramenta WAUTT (RIVOLLI et al., 2008) para a captura das interações do usuário com um sistema web, pois a abordagem desta ferramenta é baseada na captura de eventos realizados pelo usuário em elementos de interação contidos numa página web.

3.2.2.1 Limitações da análise de *log*

A avaliação automática da usabilidade através da análise de *log* pode ser proposta como uma abordagem para avaliação de usabilidade de sistemas web. A avaliação automática tem baixo custo, é de fácil condução e tem grande abrangência de usuários. No entanto, essa forma de avaliação apresenta problemas relativos à própria automatização, como (FARENC et al., 1999):

- A automatização da avaliação de usabilidade pode não levar em consideração algumas informações qualitativas e subjetivas que podem ser importantes, por exemplo, as preferências dos usuários ou interpretações errôneas dos textos ou mecanismos de interação;
- Na maioria dos casos, a inspeção automatizada das regras é feita nos níveis sintáticos e léxicos a partir do código de implementação da interface;
- Algumas regras ergonômicas requerem descrições de alto nível dos objetos usados, mas estas informações não podem ser recuperadas de forma automática.

Entretanto, a utilização da análise de *log* tem apresentado experiências positivas, como a possibilidade de identificar os caminhos (*links*) percorridos pelo usuário, as páginas mais visitadas e locais da interface nunca explorados. Adicionalmente, essa técnica pode ajudar a compreender o comportamento dos usuários com relação à interface e a melhorar a qualidade das interações.

3.3 USABILIDADE EM FORMULÁRIOS WEB

Os formulários web são um estilo de interação para entrada de dados e são utilizados por todas as categorias de usuários (NIELSEN; LORANGER, 2006). Na interação através de um formulário, o sistema solicita dados do usuário através de campos que precisam ser preenchidos. Os formulários encontrados em sistemas web se encaixam nesse estilo de interação. Os recursos necessários para a interação com formulários são somente o teclado e/ou o dispositivo de seleção e apontamento.

As vantagens do uso de formulários para entrada de dados é a realimentação de ações, a validação automática de entradas durante o preenchimento e o sentimento de controle do usuário (nos casos de formulários nos quais o preenchimento dos campos pode se realizado em qualquer ordem).

Os formulários podem ser instrumentos úteis e eficazes para monitorar métricas quantitativas e qualitativas de usabilidade de um sistema web, como: taxas de conclusão; ponto no qual os usuários pararam no formulário se não concluído; como os usuários acessam o formulário (caminho percorrido); quais os elementos do formulário foram utilizados; quais dados foram inseridos; informações sobre o navegador e sistema operacional, entre outras (WROBLEWSKI, 2008). Esses itens podem auxiliar na identificação do sucesso no cumprimento de uma tarefa.

Shneiderman e Plaisant (2010) e Wroblewski (2008) apontam várias recomendações para criação de formulários. Para este trabalho, algumas dessas recomendações são agrupadas em 5 categorias, de acordo com as características comuns encontradas entre elas e para melhor distinção e classificação entre os padrões de problemas de usabilidade:

1) Navegação

- Estabelecer uma hierarquia de apresentação de campos, na qual os campos obrigatórios precedam os opcionais (por exemplo, a interação do usuário pode indicar que a distribuição dos campos não é adequada);
- Possibilitar a navegação lógica entre campos (por exemplo, de cima para baixo e da esquerda para direita);
- Agrupar campos contendo informações afins com recursos de contorno e cor para facilitar a identificação de campos correlatos;
- Fornecer ao usuário os comandos-padrão de manipulação do formulário (por exemplo, tecla "Tab", clique de mouse, rolagem de página, etc).

2) Frequência de Interação

- Indicar claramente ao usuário como acessar e preencher cada campo implementado no formulário (por exemplo, o tempo alto de preenchimento pode indicar alguma dificuldade);
- Empregar nos campos implementados rótulos e abreviações consistentes e familiares ao usuário (isso pode diminuir o tempo de preenchimento).

3) Entrada de Dados

- Fornecer mensagens explicativas para os campos (por exemplo, se não fornecidas, podem dificultar o preenchimento dos campos pelo usuário);
- Oferecer valores padrão (*default*) para os campos sempre que possível (por exemplo, valores padrão podem não ser adequados e isso provoca a necessidade de alterar o valor; quando o campo não é obrigatório pode ser enviado o valor errado);
- Empregar o recurso de máscaras (por exemplo, o usuário não terá que adivinhar qual o formato de dados aceito por algum campo).

4) Layout

- Considerar a partição do formulário por páginas e/ou abas (por exemplo, formulários longos são mais suscetíveis a erros de preenchimento);
- Utilizar apenas uma coluna para implementação de formulários (por exemplo, é fácil ignorar a coluna da direita no preenchimento).

5) Funcionalidades

- Distinguir claramente entre ações primárias e secundárias (por exemplo, ações secundárias normalmente têm consequências indesejadas, por isso deve-se usar apenas as principais ações sempre que possível);
- Evitar convenções de nomenclatura não usuais (por exemplo, evitar palavras genéricas como “Enviar” para as ações primárias, pois dão a impressão de que a forma em si é genérica).

Pressupõe-se que cada uma dessas categorias de recomendações representa uma classe (padrão) de problema de usabilidade. A partir disso, são empregadas técnicas de reconhecimento de padrões para classificar automaticamente os potenciais problemas de usabilidade encontrados de acordo com os padrões acima citados.

4 CAPTURA DA INTERAÇÃO E FILTRAGEM DOS ARQUIVOS DE *LOG*

A Internet foi construída baseada em uma premissa simples: todo o material foi basicamente formatado em um padrão geral e uniforme, denominado HTML e todas as requisições de chamadas e respostas devem estar de acordo com as similaridades propostas neste protocolo padrão (NIELSEN; LORANGER, 2006). Qualquer comunicação entre o *browser* de um usuário e um servidor web resulta em uma entrada no arquivo de *log* do servidor web que armazena dados relativos a esta transação.

O emprego dos arquivos de *log* do servidor como forma de se apoiar a avaliação remota de sistemas web é uma abordagem interessante por representar uma estratégia prática e confiável. Os dados registrados variam de acordo com o tipo do servidor usado e do formato do arquivo de *log* que este servidor suporta. De forma geral, um arquivo de *log* do servidor contém:

- endereço IP que faz a requisição;
- data e horário da requisição;
- URL requisitada;
- tamanho do arquivo requisitado;
- protocolo empregado na requisição;
- navegador e sistema operacional usado pelo computador requisitante.

No Quadro 1 é apresentada uma entrada de um arquivo de *log* do servidor Apache[4]. Nessa entrada pode-se verificar o IP da máquina, a hora e data em que esta máquina efetuou o acesso, o tipo de requisição efetuada, a localização do componente requerido, o tamanho deste componente, endereço onde ocorreu a requisição do componente, o navegador utilizado e o sistema operacional da máquina do usuário.

127.0.0.1 - - [13/Dec/2011:15:09:57 -0200] "GET /index.html HTTP/1.1" 304 "/navigation/top_nav/jamba_dips_stat.html HTTP/1.0" "Mozilla/7.05 (Win7; I)"

Quadro 1 – Entrada de um arquivo de *log* do Servidor Apache
Fonte: Autoria própria.

[4] Apache é um servidor de páginas web robusto e estável, que permite a comunicação com o interpretador de páginas PHP.

A navegação de um usuário num sistema web está fortemente relacionada às suas necessidades, interesses e conhecimentos. A exploração dos dados de navegação presentes em arquivos de *log* do servidor tem sido utilizada para obter informações sobre a forma com que os usuários navegam em um sistema web (IVORY; HEARST, 2001). Entretanto, esses arquivos não possuem informações detalhadas que descrevam as interações do usuário na navegação, visto que somente os endereços (URLs) das páginas acessadas são registrados nos arquivos de *log*.

Devido a essa limitação na análise de *log* do servidor, este trabalho utiliza a ferramenta WAUTT para captura da interação do usuário, uma vez que esta ferramenta faz o rastreamento não somente das páginas visitadas pelo usuário, mas também dos eventos realizados em cada elemento contido nas páginas de um sistema web.

4.1 WAUTT (*Web Application User Tracking Tool*)

Nesta seção, é apresentada a ferramenta utilizada para a coleta dos dados da interação do usuário com um sistema web, denominada *Web Application User Tracking Tool* (WAUTT) (RIVOLLI et al., 2008). A abordagem dessa ferramenta usa técnicas *Asynchronous Javascript and XML* (AJAX[5]) como base para construir um código *JavaScript*[6] que é capaz de realizar a tarefa de rastreamento da interação do usuário no cliente e registrar os dados coletados em um servidor para posterior análise e interpretação.

A principal vantagem da utilização dessa ferramenta é a capacidade de captura de eventos da interação do usuário com um sistema web. As ações realizadas na interface durante a interação com um sistema web são rastreadas pela WAUTT. Essa característica facilita a classificação de problemas de usabilidade específicos.

Na Figura 3 é mostrada a organização da arquitetura da ferramenta WAUTT. O "Cliente" corresponde à página do sistema web aberta no navegador, a qual possui um código WAUTT (wautt.js) incorporado que faz o rastreamento (captura) da interação do usuário e envia uma requisição com os dados coletados para registro em um arquivo de *log*.

[5] AJAX é uma técnica usada para criar aplicações web mais interativas, usando uma combinação de tecnologias e solicitações assíncronas de informações do servidor.
[6] *Javascript* é uma linguagem interpretada embutida dentro de arquivos HTML.

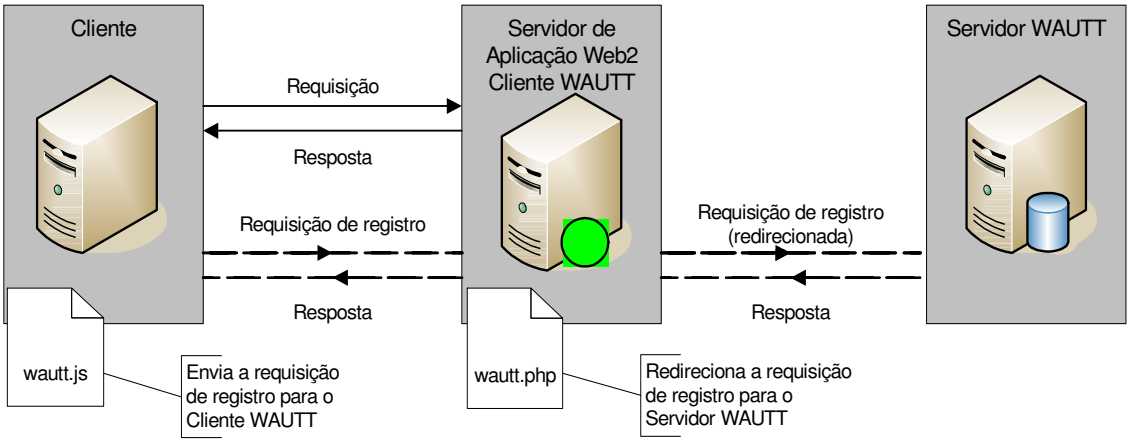

Figura 3 – Arquitetura da ferramenta WAUTT
Fonte: Rivolli et al (2008).

O "Servidor de Aplicação Web2" hospeda a Aplicação Web2[7] sendo rastreada e possui um código WAUTT, denominado "Cliente WAUTT" (wautt.php), para redirecionar a requisição do cliente para o "Servidor WAUTT". O "Servidor WAUTT", por sua vez, fornece os códigos WAUTT para o "Cliente" e para o "Servidor de Aplicação Web2", e também registra os dados coletados enviados com as requisições redirecionadas.

Outra questão que está associada à simplicidade e tem importância significativa é que o código do "Cliente WAUTT" não possui nenhum tipo de parâmetro, exceto a identificação do cliente, uma vez que os demais parâmetros são obtidos pelo código *JavaScript* ("wautt.js") incluído na página do cliente.

O próprio uso do "Cliente WAUTT" é uma decisão de projeto devido à questão das requisições AJAX *cross-domain*, que se trata de uma restrição de segurança dos navegadores e impede a comunicação com outro domínio que não seja o de origem da página. A maneira adotada para resolver essa questão é usar o próprio "Servidor de Aplicação Web2", através do "Cliente WAUTT" (wautt.php), para redirecionar as requisições de registro recebidas do Cliente para o "Servidor WAUTT".

[7] Novas tecnologias, linguagens e metodologias tornam possível a criação de aplicações interativas que representam um novo modelo, frequentemente chamado de aplicações Web 2.0 ou *Rich Internet Applications* (RIA).

O Cliente WAUTT é um código PHP[8] que inclui o código *JavaScript* (wautt.js) no conteúdo da página que é enviada ao Cliente. O código *JavaScript* faz o rastreamento da interação do usuário e envia uma requisição com os dados coletados para registro.

4.1.1 Formato dos Dados Coletados

A ferramenta WAUTT foi projetada para coletar dados detalhados sobre a interação do usuário com o sistema web. Os dados coletados são gravados em um arquivo de *log* e também armazenados em uma base de dados. O formato básico de cada registro (linha) do arquivo de *log* é o seguinte: data, hora, IP, sistema operacional, navegador, URL, <...>. A parte referenciada como "<...>" corresponde aos detalhes específicos da interação rastreada, que variam de acordo com os seguintes eventos:

- Clique do mouse: As coordenadas do clique são medidas a partir do canto superior esquerdo da página. A informação sobre qual elemento é o destino do clique também é registrada, assim como os seus atributos, por exemplo: *name*, *id*, *class*, etc.
  event:click, x:28, y:415, element:FORM, class:aba, id:lnk2, value: nome

- Rolagem da página: A coordenada da posição vertical da barra de rolagem, medida a partir do topo da página, é registrada a cada intervalo de tempo, da mesma maneira que os movimentos do mouse.
  event:scroll, y:55

- Mudança de foco: A mudança do foco da entrada de dados é registrada sempre que um elemento receber ou perder o foco.
  event:focus, element:textarea, class:msg, id:msg01, value: erro

- Carregamento de página: O registro ocorre quando uma página é carregada completamente no navegador.
  event:load, title:Entrar width:800, height:600

[8] PHP é uma linguagem de programação de páginas web.

- Movimento do mouse: As coordenadas da posição do mouse são acumuladas em um *buffer* a cada intervalo de tempo (300 ms) e são enviadas periodicamente para registro no servidor.
  event:mousemove, x:169, y:130

- Redimensionamento da janela: Um registro para esse evento é adicionado sempre que o usuário redimensionar a janela do navegador.
  event:resize, width:620, height:440

O formato de cada entrada de um arquivo de *log* gerado pela ferramenta é apresentado no Quadro 2.

05/06/2012, 13:50:12, 127.0.0., mozilla – 8.0.1, http:// localhost/ cadastro.php, event: click:, x:507, y:47, element: INPUT, value/text: Voltar

Quadro 2 – Entrada de um arquivo de *log* da WAUTT
Fonte: Autoria própria.

Na Tabela 1 pode-se visualizar o significado dos atributos presentes na entrada de um arquivo de *log* da WAUTT, apresentada no Quadro 2.

Tabela 1 – Atributos do arquivo de *log* da ferramenta WAUTT

Campos	Valores
Data da solicitação	05/06/2012
Horário da solicitação	13:50:12
Endereço do IP	127.0.0.1
Navegador/versão	mozilla – 8.0.1
Documento solicitado	cadastro.php
Evento	Clique do mouse
Coordenadas	x:506, y: 47
Elemento	Input (Botão)
Valor/Texto	Voltar

Fonte: Autoria própria.

O emprego da ferramenta WAUTT permite que os dados sejam mostrados basicamente de forma bruta, cabendo ao avaliador analisá-los e tirar conclusões quanto à usabilidade de um sistema web. Neste trabalho é realizada uma filtragem dos arquivos *log* procurando eliminar entradas não importantes para o problema de identificação de potenciais problemas de usabilidade, e consequentemente reduzir a dimensão do volume de dados a serem analisados pela técnica de reconhecimento de padrões.

4.2 FILTRAGEM DOS ARQUIVOS DE *LOG*

A ideia principal no processo de filtragem dos arquivos de *log* é analisar o Modelo de Tarefas (base para o processamento da filtragem), o arquivo de *log* para cada tarefa e armazenar num arquivo de *log* refinado somente as linhas comuns ao Modelo de Tarefas (MT) e ao arquivo de *log* original. Portanto, o arquivo de *log* filtrado é constituído somente pelas entradas identificadas nesses dois arquivos.

4.2.1 Formalização do Problema da Filtragem

Um sistema web pode ser considerado um conjunto de recursos, isto é, páginas e seus elementos, por exemplo: imagens, *links*, folhas de estilo, etc. Assim, um sistema web pode ser representado por um conjunto de recursos, $Rec = \{ rec_1, rec_2, rec_3, ..., rec_v \}$, onde v é a quantidade de recursos oferecidos. Adicionalmente, um sistema web possui um conjunto de usuários, $U = \{ u_1, u_2, u_3,..., u_w \}$, com acesso a este sistema.

Dessa forma, é possível representar uma entrada no arquivo de *log* por $l_i = \{ u_i, d, h, c, E, rec_i \}$, onde $u_i \in U$, $rec_i \in Rec$; d representa a data do acesso; h representa o horário do acesso; c o código de status que identifica se houve sucesso ou fracasso na requisição; e E representa o conjunto de eventos capturados da interação com o sistema web.

Quando um usuário acessa uma página, ele visualiza um conjunto de recursos que representa a mesma (MORANDINI et al., 2007). Assim, pode-se representar a visualização de uma página por $p_i = \{ rec_{i1}, rec_{i2},..., rec_{ip} \}$, onde $rec_{ij} \in Rec$. Sabe-se que rec_{i1} representa

a página desejada pelo usuário, sendo os demais recursos automaticamente embutidos e requisitados pelo navegador web.

Por outro lado, o MT apresenta sequências de entradas para cada tarefa, ou seja, sequências de eventos específicos que definem uma tarefa, $E = \{ e_1, e_2,..., e_n \}$, onde n é o número máximo de eventos necessários para a realização de uma tarefa.

A filtragem do arquivo de *log* é realizada de forma que cada registro não represente mais de um acesso a um recurso, mas sim, a realização de um evento efetivo do usuário. Desse modo, uma entrada no arquivo de *log* filtrado (l_{fi}) é representado por $l_{fi} = \{ u_i, d, h, c, E, p_i \}$.

Logo, o problema da filtragem é definido como: dado um arquivo de *log* com entradas l_i de um determinado sistema web e um modelo de tarefas MT, transforme os registros deste arquivo de tal forma que todas as entradas representem reais interações do usuário (MORANDINI, et al., 2007).

4.2.2 O Problema da Filtragem em Termos Práticos

Um sistema web é formado por um conjunto de recursos e uma página web é constituída por um subconjunto destes recursos. Dessa forma, pode-se afirmar que o problema da filtragem consiste em retirar do arquivo de *log* alguns dos recursos embutidos na página em questão.

Analisando alguns recursos embutidos, pode-se identificar o acesso a arquivos de imagem (jpg e gif), animação (swf) e arquivos de estilo (CSS); a maioria dos arquivos embutidos possui estas extensões. No entanto, não é possível afirmar que para realizar a filtragem do arquivo de *log* devem ser removidos todos os acessos aos arquivos que possuam essas extensões, pois o *download* de uma imagem pode ser considerado um acesso válido no modelo de tarefas. Isso pode ocorrer em sistemas web que possuam tarefas que incluam a visualização de uma determinada imagem, ou seja, o usuário realizou efetivamente a ação de visualizar a imagem.

Assim, o mecanismo de processamento dos arquivos de *log* neste trabalho foi elaborado para permitir a remoção de registros de acesso à recursos não úteis na avaliação da usabilidade e na aplicação de técnicas de reconhecimento de padrões.

Ao desconsiderar os recursos não-úteis deve-se avaliar também as requisições realizadas por ferramentas que fazem a varredura do sistema web para extrair informações sobre o conteúdo. Essas ferramentas, também chamadas de *spiders*, são utilizadas periodicamente por mecanismos de busca, como o Google. Portanto, a filtragem deve tornar o arquivo de *log* consistente e sem dados inexpressíveis que possam influenciar na identificação e classificação de problemas de usabilidade.

5 MODELO DE TAREFAS

Sistemas computadorizados são projetados para auxiliar as pessoas a executarem tarefas. Logo, tarefas devem ser de interesse central para os desenvolvedores de *software*. De fato, diversos autores convergem para a idéia central de que para projetar sistemas com maior usabilidade deve-se compreender melhor as tarefas executadas pelos usuários de modo a aplicar seu entendimento das tarefas no desenvolvimento de sistemas (PREECE et al., 2012; AQUINO et al., 2011; DIAPER, 2011; KADLEC, 2011; YASMEEN; GUNTER, 2011).

Pode-se controlar a navegação do usuário em um sistema web, porém o usuário controla fundamentalmente a navegação entre as páginas deste sistema (NIELSEN; LORANGER, 2006). É possível forçar o usuário por caminhos definidos e evitar que estabeleçam *links* com determinadas páginas, mas os sistemas web que usam esta estratégia são considerados rígidos. Assim, os sistemas web devem ser projetados visando à liberdade de movimento do usuário.

Considerando que o interesse do usuário em visitar determinada página web é um indicativo de seu comportamento, existe também a necessidade de se definir onde começa e termina uma instância comportamental. Nielsen (2000) afirma que a estrutura de um sistema web deve ser determinada pelas tarefas que o usuário deseja realizar. Um sistema web estruturado dessa forma favorece a delimitação de uma instância que pode iniciar com a página inicial do sistema web e terminar quando o usuário alcança a página de conclusão da tarefa.

A principal meta dos enfoques de projeto de interface é aumentar a qualidade da interface com o usuário produzindo sistemas interativos não somente funcionais e confiáveis, mas também usáveis. Os enfoques de projeto baseados em modelo (*model-based*) permitem representar várias informações da interface e *design* em um alto nível de abstração (DIAPER; STANTON, 2004; DIAPER, 2011). Isso permite várias vantagens, como: acoplamento a um enfoque metodológico de concepção, rastreabilidade e reuso de modelos, geração de interfaces a partir destes modelos, e melhor reflexão sobre as decisões e exploração de alternativas de *design*.

Entre os modelos comumente presentes nesses enfoques, estão os modelos de: usuário, diálogo, apresentação, domínio, contexto, plataforma tecnológica e tarefas. Em particular, o uso de Análise de Tarefas e Modelo de Tarefas visa um melhor entendimento de

propriedades das tarefas realizadas pelos usuários em suas atividades e a aplicação deste entendimento no processo de construção da interface.

5.1 DEFINIÇÕES

Inicialmente, deve-se considerar o significado do termo "tarefa" que pode assumir interpretações diferentes segundo vários autores, como Draper e Norman (1985), Benyon (1992), Storrs (1995), Bodart et al. (1994) e Preece et al. (2012). A existência de várias definições de tarefa implica na existência de diferentes métodos para análise de tarefas e de diferentes modelos de tarefas para representar seus resultados segundo estas diferentes perspectivas. Neste trabalho, uma tarefa é um conjunto ordenado de ações para satisfazer um objetivo em um contexto específico (STORRS, 1995).

A Análise de Tarefa (AT) é definida como um método empírico que permite descrever e analisar como os usuários realizam suas atividades (PREECE et al., 1994).

A Análise de Tarefas é utilizada basicamente para entender as tarefas de um ambiente de trabalho e tem como objetivo obter informações para discutir causas e soluções dos problemas encontrados e possivelmente determinar necessidades de treinamento (PISO, 2011).

Existem vários métodos de AT e entre os principais, destacam-se (WINCKLER; PIMENTA, 2004): Análise Hierárquica de Tarefas (*Hierarchical Task Analysis*) e Análise Cognitiva de Tarefas (*Cognitive Task Analysis*).

A Análise Hierárquica de Tarefas (AHT) decompõe uma tarefa de modo *top-down* (de cima para baixo) para formar uma hierarquia de subtarefas, que por sua vez, pode ser decomposta sucessivamente. Em geral, sugere-se que a decomposição acabe quando for atingido um nível baixo de descrição em termos de ações elementares que não podem ser decompostas, por exemplo, os movimentos e cliques de mouse, pressionamento de teclas, etc.

A Análise Cognitiva de Tarefas (ACT) está centrada nos princípios psicológicos ou cognitivos relacionados à tarefa. Assim, o conhecimento ou habilidade necessária para a realização da tarefa são investigados e por isto se concentra mais na análise do desempenho do usuário e as razões de sua variabilidade (WINCKLER; PIMENTA, 2004).

Embora existam várias formas de realizar a análise de tarefas, o processo pode ser resumido nas seguintes etapas (GREENBERG, 2004):

- Inventariar tarefas – utilizar diferentes técnicas de coleta; o ideal é observar e/ou entrevistar o usuário tentando identificar objetivos gerais e construir uma lista de tarefas relacionadas a estes objetivos.
- Selecionar tarefas a serem analisadas e descritas com mais detalhes – tipicamente as tarefas selecionadas incluem as mais frequentemente realizadas ou as mais críticas.
- Descrever as tarefas – decompor e ordenar as tarefas; decompor significa detalhar as tarefas, descrevendo-as em termos de subtarefas. Nesta etapa é construído o modelo de tarefas (MT).
- Validar as tarefas descritas – uma alternativa simples é apresentar o MT a alguém não envolvido na AT, mas que conheça as tarefas para verificar sua consistência.

A análise de tarefas realizada em um domínio específico pode produzir uma descrição explícita de tarefas chamada Modelo de Tarefas (WINCKLER; PIMENTA, 2004). Muitos modelos de tarefas são usados para representar os resultados da análise de tarefa, cada modelo enfatizando uma perspectiva.

Um Modelo de Tarefas (MT) é uma descrição lógica das atividades a serem executadas para alcançar os objetivos do usuário (PATERNÒ, 2001). Geralmente, um MT descreve certos conceitos relacionados de forma a representar aspectos relevantes das tarefas e dos usuários. Alguns conceitos e relacionamentos são comuns e presentes em quase todos os tipos de MT, enquanto outros são especificamente propostos e usados em um modelo em particular. Os conceitos e relacionamentos mais comuns de MT são (WINCKLER; PIMENTA, 2004):

- Decomposição da tarefa: uma tarefa é tipicamente descrita de maneira hierárquica, onde o nível mais alto contém a tarefa principal sucessivamente dividida em subtarefas menores recursivamente até que, nos níveis mais baixos, são descritas ações físicas que não podem mais ser decompostas;
- Relacionamentos causais/temporais: descrevem o fluxo da tarefa, indicando a ordem em que as subtarefas são executadas, geralmente através de construtores típicos (sequência, simultaneidade, entrelaçamento, etc.);

Além desses, existem outros elementos que descrevem objetos (manipulados pelas tarefas), papéis e atores (responsáveis pelas tarefas), restrições e propriedades (frequência de

realização, prioridade, etc.) associados às tarefas. Mais detalhes sobre todos esses elementos estão presentes em Welie et al. (1998), Balbo (2004) e Limbourg e Vanderdonckt. (2004). Outros trabalhos envolvem conceitos mais avançados de MT, como padrões de tarefas (*task patterns*) estão descritos em Breedvelt et al. (1997) e Paternò (2001); modelos de tarefa cooperativos podem ser encontrados em Van der Veer et al. (1996); e *templates* de interação são descritos em Paquette e Schneider (2005).

5.2 TIPOS DE MODELOS DE TAREFAS

Existem vários modelos de tarefa descritos na literatura, cada um com um subconjunto de aspectos associados à maneira pela qual os usuários realizam suas tarefas. Em geral, uma notação está tipicamente direcionada a modelar/representar alguma perspectiva sobre a tarefa, entre as quais:

- Modelos de Tarefas Cognitivas do Usuário (também denominados Modelos Cognitivos – *Cognitive Models*) – representam o conhecimento que o usuário necessita para a realização de uma tarefa, por exemplo: GOMS (*Goals, Operators, Methods and Selection rules*) (CARD et al., 1983).
- Modelos de Tarefas Interativas (também chamados Modelos de Interação ou Modelos de Diálogo – *Interactive Task Model*) – descrevem as ações que o usuário precisa realizar em um sistema para atingir seus objetivos, por exemplo: UAN (HARTSON et al., 1990);
- Modelos de Tarefas de Usuário (*User Tasks*) – descrevem como as ações do usuário se organizam no decorrer de suas atividades, por exemplo: MAD (SCAPIN; PIERRET-GOLBREICH, 1989; SCAPIN; BASTIEN, 2001).

Os modelos de tarefas são uma maneira adequada de considerar o ponto de vista do usuário na concepção, pois representam conceitos relacionados às atividades que o usuário realiza. Consequentemente, os modelos de tarefas permitem que o usuário compreenda, valide e participe ativamente do processo de análise.

Dentre os modelos de tarefas mais comuns, podem ser destacados o Método Analítico de Descrição de tarefas (MAD) (SCAPIN; PIERRET-GOLBREICH, 1989), *User Action Notation* (UAN) (HARTSON et al., 1990), *Interactive Cooperative Objects* (ICO)

(PALANQUE et al., 1995), GOMS (*Goals, Operators, Methods and Selection Rues*) (CARD et al., 1983) e *Concurrent Task Trees* (CTT) (PATERNÒ, 2001), descritos a seguir.

5.2.1 Método Analítico de Descrição de tarefas (MAD)

O Método Analítico de Descrição de tarefas (MAD) (SCAPIN; PIERRET-GOLBREICH, 1989; CAFFIAU et al., 2010) é uma notação e metodologia que visa à descrição de tarefas com o objetivo de melhor inserir preocupações ergonômicas na concepção de interfaces com usuários.

O MAD baseia-se na descrição da tarefa e na decomposição hierárquica de tarefas em subtarefas com auxílio de construtores pré-definidos que descrevem relações temporais e lógicas entre estas diferentes subtarefas. Uma tarefa ou subtarefa é representada como uma árvore cujos nodos são tarefas, subtarefas ou ações elementares, e assim sucessivamente.

A descrição da tarefa consiste em um conjunto de propriedades que caracterizam esta tarefa: um estado inicial, um estado final, pré-condições, pós-condições e atributos (prioridade, tarefa opcional ou obrigatória, etc.).

5.2.2 *User Action Notation* (UAN)

A *User Action Notation* (UAN) (HARTSON et al., 1990; BASUKI et al., 2009) consiste em uma notação orientada a tarefas e ações do usuário para descrever o projeto de interfaces assíncronas de manipulação direta. A UAN é utilizada tipicamente para descrever como o usuário executa uma tarefa interativa, mas não como o sistema é implementado para interpretar o comportamento do usuário. Essa notação foi concebida para servir como uma forma de comunicar a lógica de operação da interface a um projetista de *software* visando a sua implementação.

Em UAN, uma interface é representada como uma estrutura quase hierárquica de tarefas assíncronas. No seu mais baixo nível de abstração, UAN é uma notação orientada a tarefas que descreve ações do usuário, os correspondentes *feedbacks* da interface e a informação de estado (WINCKLER; PIMENTA, 2004).

A unidade básica para a UAN é uma ação física: o movimento do cursor, clique do mouse. No entanto, não fornece detalhes explícitos sobre como ações físicas são implementadas e não existem regras rápidas sobre a notação. Os relacionamentos temporais são focados principalmente nas ações do usuário, negligenciando aspectos temporais das categorias de tarefa não consideradas nas ações do usuário.

5.2.3 *Interactive Cooperative Objects* (ICO)

Interactive Cooperative Objects (ICO) (PALANQUE et al., 1995; NAVARRE et al., 2009) é um formalismo baseado em Redes de Petri a Objetos[9] para modelagem do comportamento de um sistema interativo e que pode servir também para modelar as tarefas interativas entre o usuário e sistema. O formalismo considera que um objeto ICO é composto de:

- Estrutura de dados (atributos);
- Conjunto de operações (serviços internos ou serviços disponíveis como mensagens);
- Estrutura de controle do objeto para modelar seu comportamento definido através de uma Rede de Petri de alto nível;
- Apresentação para descrever sua aparência externa e que está estruturada como um conjunto de *widgets* (botões, *checkboxes*, etc.);
- Função de ativação que, a cada par (*widget*, ação sobre o *widget*) associa um serviço do objeto ICO.

Além disso, o sistema interativo é descrito através de um conjunto de ICOs e de uma Rede de Petri para descrever o comportamento do sistema e as interações entre os ICOs. O princípio básico de modelagem ICO é utilizar um protocolo cliente-servidor para assegurar que o conjunto de serviços solicitados aos outros objetos esteja incluso no conjunto de

[9] Existem várias extensões para Redes de Petri, dentre elas está a extensão para Redes de Petri hierárquicas, nas quais a hierarquia é representada na forma de diferentes visões suportando níveis de refinamento e abstração. Isso pode ser encontrado nas chamadas Redes de Petri a Objeto ou Sistemas Objeto, nas quais uma Rede de Petri pode conter outra Rede de Petri, introduzindo o conceito de Redes de Petri aninhadas que se comunicam ao sincronizar transições entre diferentes níveis.

serviços oferecidos por eles. Para avaliar a cooperação com os usuários, cada usuário é considerado como um ICO (WINCKLER; PIMENTA, 2004).

Com ICO pode-se verificar se o modelo de concepção do sistema suporta o modelo de tarefa do usuário. Como os dois modelos têm uma base comum, pode-se através de técnicas formais associadas à teoria das redes de Petri: validá-los de maneira isolada, isto é, mostrar que cada ICO é capaz de fornecer seus serviços; compará-los e validar sua cooperação (NAVARRE et al., 2009).

O uso de formalismos como ICO tem uma série de vantagens, mas a principal é a possibilidade de especificar o modelo de tarefas com precisão e sem ambigüidades. Isso permite a análise de características do modelo sem a necessidade de uma versão executável do sistema ou a participação do usuário.

Apesar dos novos conceitos propostos por ICO suportarem a especificação de sistemas multimodais e a simulação dos modelos, sua abordagem não oferece uma forma de geração de código. Consequentemente, projetistas e desenvolvedores necessitam conhecer todos os recursos do formalismo para concepção dos sistemas.

5.2.4 *Goals, Operators, Methods and Selection rules* (GOMS)

O modelo GOMS (acrônimo de *Goals, Operators, Methods and Selection rules*) foi desenvolvido como um modelo preditivo do desempenho de um ser humano ao interagir com um sistema (CARD et al., 1983; OYEWOLE; HAIGHT, 2011). Basicamente, o GOMS é um modelo de conhecimento e dos processos cognitivos do usuário quando interage com sistemas. Este modelo tem sido usado principalmente para predizer o desempenho de usuários ao comparar diferentes aplicações e equipamentos e para ajudar a análise da eficiência de novos sistemas sem a necessidade de envolver usuários.

O GOMS pode ser usado no projeto de tarefas e por isto a noção de método é muito importante. Um método é uma sequência de operadores que descreve o desempenho de uma tarefa. Operadores são ações físicas (pressionar uma tecla, apontar um item com mouse, “clicar” o mouse, etc) e cognitivas (decidir qual comando usar, esperar, etc) que o usuário executa. As tarefas são realizadas para alcançar objetivos, isto é, estados que o usuário quer atingir e podem ser decompostas em subtarefas, que correspondem a objetivos intermediários.

Apesar de suas vantagens, o GOMS não é frequentemente usado em avaliação por envolver um conjunto muito limitado de tarefas de entrada de dados, por não modelar erros adequadamente e não levar em consideração informações contextuais (interrupções nas tarefas, fadiga, etc) que podem mudar comportamentos e desempenhos previstos. O método também não leva em conta a mínima hesitação do usuário, erro ou incidente de interação, somente prevê o tempo da interação perfeita.

5.2.5 *Concurrent Task Trees* (CTT)

Concurrent Task Trees ou resumidamente *ConcurTaskTrees* (CTT) (PATERNÒ, 2001; KOLB, 2012) é uma notação que tem a capacidade de expressão na modelagem de sistemas interativos e vem sendo muito difundida e utilizada pela comunidade internacional de IHC. Além disto, a existência de uma ferramenta de edição e de suporte a análise de modelos, chamada *Concurrent Task Trees Environment* (CTTE) (PATERNÒ, 2001) facilita a criação de modelos de tarefas. A seguir são apresentados os fundamentos do modelo CTT e os principais elementos da notação (WINCKLER; PIMENTA, 2004):

- Foco nas atividades: permite que o projetista se concentre nas atividades que os usuários precisam realizar com a interface sem se preocupar com detalhes de como a tarefa será implementada pelo sistema;
- Estrutura hierárquica: suporta a representação de vários níveis de abstração pela decomposição de tarefas complexas em subtarefas menores;
- Representação gráfica: modelos são descritos graficamente na forma de uma árvore de tarefas;
- Suporte a relacionamentos temporais: um vasto conjunto de operadores é disponível, permitindo definir relacionamentos temporais entre tarefas;
- Alocação de tarefas: permite descrever os agentes que realizam a tarefa (usuário e/ou sistema);
- Objetos: permite indicar objetos do domínio do problema que são manipulados durante a execução da tarefa.

Os elementos da notação CTT são definidos em torno da ideia de que o objetivo do usuário ao realizar uma tarefa pode ser traduzido como uma modificação do estado do sistema ou uma consulta a um recurso do sistema (KOLB, 2012). Assim, os modelos criados em CTT podem ser interpretados como uma representação dos estados possíveis do sistema e do usuário durante a realização de uma tarefa específica.

Segundo essa abordagem, a execução de cada tarefa individual é capaz de modificar a configuração de estados do sistema. O comportamento do modelo de tarefas é definido em CTT pela adição de operadores temporais (como: escolha, repetição, sequência, etc.) entre as tarefas. A existência desses operadores temporais torna possível a modelagem de tarefas em sistemas interativos.

O Modelo de Tarefas desenvolvido neste trabalho é apresentado no Capítulo 7 (Seção 7.1.3) e utiliza uma combinação dos conceitos de alguns modelos de tarefas apresentados neste Capítulo.

6 RECONHECIMENTO DE PADRÕES

O Reconhecimento de Padrões trata da classificação de uma estrutura de dados através de um conjunto de características (DUDA et al., 2001). O RP envolve técnicas para a atribuição dos padrões às suas respectivas classes de forma automática ou com a menor intervenção humana possível. Um padrão é uma descrição de um objeto e a classe de padrões é uma família de objetos que compartilham uma mesma propriedade (THEODORIDIS; KOUTROUMBAS, 2008). Assim, um sistema clássico de RP tem por objetivo extrair informações dos objetos a serem reconhecidos, analisar as informações extraídas e agir de acordo com a análise destas informações.

Existem duas abordagens básicas que podem ser tomadas no conjunto de dados no qual se deseja fazer o reconhecimento: Reconhecimento de Padrões Sintático e Reconhecimento de Padrões Estatístico ou Numérico. Neste trabalho, é utilizada a abordagem de reconhecimento de padrões numérico, pois se conhecem a priori a natureza estatística da informação que se pretende representar e a natureza dos resultados esperados. O objetivo do RP numérico é a classificação de um padrão a partir de um vetor numérico, conhecido como vetor de características. Esse vetor é comparado com um vetor representativo de uma classe de padrões. Assim, o vetor de entrada é classificado de acordo com a maior similaridade entre este e o vetor representativo de uma das classes.

Um sistema clássico de RP engloba três grandes etapas (DUDA et al., 2001): 1) representação dos dados, 2) extração ou seleção de características e 3) classificação do objeto em estudo, conforme ilustrado na Figura 4.

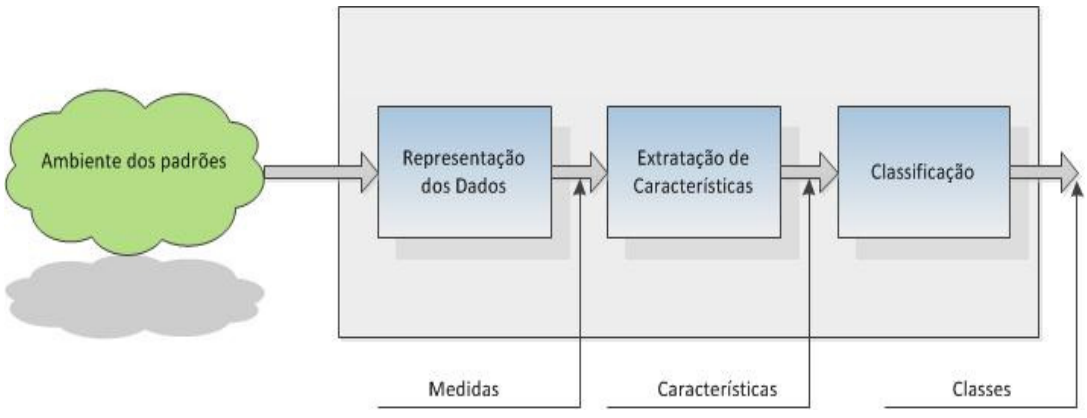

Figura 4 – Sistema Clássico de Reconhecimento de Padrões
Fonte: Adaptado de Duda et al. (2001, p. 10).

A primeira etapa refere-se à representação dos dados de entrada, que podem ser mensurados a partir do objeto a ser estudado. Essa mensuração descreve padrões característicos do objeto, possibilitando a sua posterior classificação numa determinada classe. Em geral, padrões são descritos por uma sequência de números reais ou caracteres que podem, sem perda de generalidade, serem representados por um vetor num espaço multidimensional.

A segunda etapa consiste na extração de características e atributos do objeto e consequente redução da dimensionalidade do vetor padrão. A escolha das características é de fundamental importância para um bom desempenho do classificador, sendo realizada objetivando os fenômenos que se pretende classificar (THEODORIDIS; KOUTROUMBAS, 2008). A redução de dimensionalidade permite diminuir a quantidade de características necessárias para a identificação dos padrões.

A terceira etapa do Reconhecimento de Padrões trata da classificação. Nessa etapa é abordada a concepção do classificador. O aprendizado de máquina em reconhecimento de padrões é um método que permite organizar uma sequência de padrões em vários conjuntos de padrões denominados classes, de maneira que os padrões organizados em cada conjunto são similares entre si e dissimilares dos padrões armazenados nos outros conjuntos (DUDA et al., 2001).

O Aprendizado de Máquina (*Machine Learning*) é a área de Inteligência Artificial cujo objetivo é o desenvolvimento de técnicas computacionais sobre o processo de aprendizado. Um sistema de aprendizado tem a função de analisar informações e generalizá-las para a extração de novos conhecimentos (BISHOP, 2007).

Alguns exemplos de aplicações atuais que requerem técnicas eficientes e robustas de reconhecimento de padrões podem ser citados:

- Inspeção visual para automação industrial;
- Bioinformática: análise de sequências do genoma;
- Diagnóstico médico;
- Mineração de Dados (*Data Mining*);
- Classificação de documentos recuperados da Internet;
- Análise de imagens de documentos para reconhecimento de caracteres;
- Reconhecimento biométrico, incluindo faces, íris ou impressões digitais;
- Reconhecimento de fala.

Um ponto em comum nessas aplicações é que usualmente as características disponíveis nos padrões de entrada não são diretamente utilizadas. Geralmente, utilizam-se as características extraídas dos padrões de entrada, otimizadas através de procedimentos guiados pelos dados. Um problema de reconhecimento de padrões bem definido e restrito permite uma representação compacta dos padrões e uma estratégia de decisão simples.

Neste trabalho é utilizado o extrator de características denominado Análise de Componentes Principais ou *Principal Component Analysis* (PCA), pois este é o melhor extrator de características linear conhecido (ABDI; WILLIAMS, 2010), além de proporcionar a redução da dimensionalidade do conjunto original de dados sem perda significativa das informações. Como classificador dos padrões é adotada uma Rede Neural Artificial do tipo *Perceptron* de Múltiplas Camadas ou *MultiLayer Perceptron* (MLP), pois este tipo de rede é o mais indicado para problemas de reconhecimento de padrões. Nas próximas subseções são abordadas as técnicas e conceitos utilizados em cada fase do reconhecimento de padrões.

6.1 REPRESENTAÇÃO DOS DADOS DE ENTRADA

Como etapa inicial do reconhecimento de padrões é necessária uma análise preliminar do objeto de estudo com o objetivo de trabalhar somente com os dados que tragam informações essencialmente relevantes sobre as classes. Esse processo deve ser realizado antes da mensuração dos dados de entrada, ou seja, somente as informações selecionadas são mensuradas e farão parte do vetor de entrada.

Para este trabalho, é realizada uma seleção inicial dos atributos (características) dos dados contidos em arquivos de *log*, considerando somente aqueles considerados relevantes para o processo de reconhecimento. Após essa seleção, é realizada uma filtragem dos arquivos de *log* com o objetivo de eliminar as entradas não significativas para o objeto de estudo. Para a realização da filtragem é desenvolvido um modelo de tarefas. Nesse modelo de tarefas, os dados são organizados de acordo com a ocorrência de comportamentos nas tarefas, com o objetivo de posterior montagem de um vetor de entrada.

Também é realizada a mensuração e normalização dos dados armazenados no arquivo de *log*. A mensuração consiste em transformar os dados em valores numéricos para o emprego das técnicas de RP. A normalização consiste em arranjar esses dados numéricos

dentro de uma faixa de valores, para que a classificação dos padrões seja realizada corretamente.

6.2 EXTRAÇÃO DE CARACTERÍSTICAS

Esta etapa do reconhecimento de padrões consiste na extração de características e atributos do objeto e consequente redução da dimensionalidade do vetor de características (DUDA et al., 2001). O termo dimensionalidade é atribuído ao número de características de uma representação de padrões, ou seja, a dimensão do espaço de características.

Um problema fundamental em reconhecimento de padrões é a definição das características dos dados que são realmente importantes para o processo de classificação de padrões, transformando amostras de entrada em um novo espaço (espaço de características), onde a informação acerca das amostras é retida, mas a ordem de dimensão é reduzida (JAIN et al., 2000).

A escolha das características é de fundamental importância para um bom desempenho do classificador. Essa escolha é feita objetivando os fenômenos que se pretende classificar. O Extrator de Características tem como função determinar e extrair as características mais significativas que contribuam para a descrição do objeto, dentre as infinitas características que possam descrevê-lo. Outro dado relevante é que o extrator de características varia de acordo com o sistema a ser analisado.

Nesta etapa, os objetivos básicos a serem alcançados são:

- Redução da dimensionalidade do vetor de características, sem que isso implique em perda de informação relevante para a classificação;
- Extração das características significativas para a tarefa de classificação.

6.2.1 Redução da Dimensionalidade

Existem quatro principais razões para que a dimensionalidade seja a menor possível: custo de medição, precisão do classificador, rapidez do classificador e menor uso de memória (JAIN et al., 2000). No entanto, se a redução de dimensionalidade for excessiva, o

classificador pode ter seu poder de discriminação reduzido. Por isso, é importante analisar a variação do comportamento do classificador com o número de características, de forma que seja possível estimar a dimensionalidade ideal para determinado classificador e conjunto de dados.

Neste trabalho é aplicada a técnica de Análise de Componentes Principais, descrita na próxima subseção.

6.2.2 Análise de Componentes Principais (PCA)

A técnica de Análise dos Componentes Principais ou *Principal Component Analysis* (PCA) é o melhor extrator de características linear conhecido (ABDI; WILLIAMS, 2010). Nesta técnica, os dados ou a combinação destes podem ser tratados como padrões em um espaço linear para efetuar reconhecimento estatístico (JOLLIFFE, 2005).

Em reconhecimento de padrões é sempre desejável dispor de uma representação compacta e de um bom poder de discriminação de classes de padrões. Para isso, é importante que não ocorra redundância entre as diferentes características dos padrões, ou seja, que não exista uma covariância significativa entre os vetores da base do espaço de características.

Um espaço vetorial com a propriedade de não haver covariância entre os vetores da base do espaço possui uma base cuja matriz de covariância de seus vetores é diagonal. Para diagonalização da matriz de covariância dos padrões de treinamento, deve-se efetuar uma mudança de base (ABDI; WILLIAMS, 2010).

O PCA é matematicamente definido como uma transformação linear ortogonal que transforma os dados para um novo sistema de coordenadas, de forma que a maior variância por qualquer projeção dos dados fica ao longo da primeira coordenada, chamada primeiro componente, a segunda maior variância fica ao longo da segunda coordenada, chamada segundo componente, e assim por diante (JOLLIFFE, 2005).

Com a transformação linear ortogonal da matriz de dados originais são criadas novas variáveis chamadas de componentes, as quais não apresentam correlação. Teoricamente, o número de componentes é sempre igual ao número de variáveis originais. Entretanto, um número mais restrito de componentes é responsável por grande parte da representação dos dados.

Os componentes são extraídos na ordem do mais explicativo para o menos explicativo, sendo que o primeiro componente explica o máximo possível da variação do conjunto de dados original, e cada sucessivo componente explica o máximo possível da variação restante não explicada pelos componentes anteriores (ABDI; WILLIAMS, 2010).

Os componentes principais são combinações lineares de características, identificadas por X_i, para $i = 1, 2,..., p$, com observações designadas por: $\{ x_{i1}, x_{i2},..., x_{in} \}$. Essas combinações são obtidas a partir da matriz de covariâncias ou de correlação, associadas às p características.

A análise de componentes principais se inicia com os dados de p características para n observações. O primeiro componente é então a combinação linear das características X_1, X_2, ..., X_p, conforme Equação (1):

$$Pc_1 = a_{11}X_1 + a_{12}X_2 + \ldots + a_{1p}X_p \qquad (1)$$

Dessa forma, o segundo componente é dado pela Equação (2), assim como os demais componentes:

$$Pc_2 = a_{21}X_2 + a_{22}X_2 + \ldots + a_{2p}X_2 \qquad (2)$$

Se existem p características, então existirão no máximo p Componentes Principais, ou *Principal Components* (PCs). De fato, para obter uma PCA é necessário determinar os autovalores de uma matriz de covariância, C. A matriz de covariância é simétrica e sua forma pode ser observada na Equação (3):

$$C = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{p1} & c_{p2} & \cdots & c_{pp} \end{pmatrix} \qquad (3)$$

Nessa matriz, o elemento c_{ii} da diagonal principal é a variância de X_i. Os demais termos, $c_{ij,}$ são covariâncias entre as características X_i e X_j. As variâncias dos PCs são os autovalores da matriz C. Existem p autovalores, alguns dos quais podem ser zero; autovalores negativos não são possíveis para uma matriz de covariância. Nos casos em que os PCs obtidos

são influenciados pelas variáveis de maior variância, costuma-se utilizar a matriz de correlação, pois esta normaliza os dados, eliminando problemas de características com unidades e ordens de grandeza diferentes. Para isso, é necessário normalizar os dados amostrais. Esta normalização é realizada de acordo com a Equação (4), que gera a matriz de correlação, *R*:

$$x_{ij}^{*} = \frac{x_{ij} - x_m}{\sigma} \tag{4}$$

onde:

x_{ij}^{*} – valor normalizado da variável *i*, para *i* = 1, 2,..., *p*, na sua *j*-ésima observação, para *j* = 1, 2,..., *n*;

x_m – média;

σ – desvio padrão das *n* observações de x_i respectivamente.

A matriz de correlação também é simétrica, de acordo com a Equação (5):

$$R = \begin{pmatrix} 1 & r_{12} & \cdots & r_{1p} \\ r_{21} & 1 & \cdots & r_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p1} & r_{p2} & \cdots & 1 \end{pmatrix} \tag{5}$$

Com a matriz de correlação pode-se calcular os autovalores da matriz *R* utilizando a Equação (6), também chamada de equação característica:

$$\left| R - \lambda I \right| = 0 \tag{6}$$

onde λ é o conjunto de escalares designados por { λ_1, λ_2, ..., λ_p }, chamados de autovalores da matriz *R*.

De posse dos autovalores, calculam-se os autovetores de acordo com a Equação (7):

$$\left| R - \lambda I \right| \gamma_i = 0 \tag{7}$$

onde *I* é uma matriz identidade.

A proporção da variância descrita por cada PC é calculada através da Equação (8), para cada um dos autovalores encontrados na Equação (6):

$$V_i = \frac{\lambda_i}{\sum_{i=1}^{p} \lambda_i} \tag{8}$$

onde:

λ_i – autovalores de *R*;

I – matriz identidade;

γ_i – *i* autovetores da matriz *R*, para $i = 1, 2, ..., p$;

V_i – porção da variância representada pelo autovetor relacionado ao *i*-ésimo autovalor.

Cada autovetor representa os coeficientes dos PCs (a_{i1}, a_{i2}, ..., a_{ip}), descritos nas Equações (1) e (2).

Quando se aplica a análise de componentes principais com o objetivo de reduzir o número de características, espera-se que os primeiros componentes expliquem uma proporção significativa da variância total dos dados, isto é, que os dados possam ser representados por um pequeno número de componentes principais sem que haja uma perda significativa de informação. No entanto, existem várias regras práticas para determinar quantos componentes deve-se excluir da análise (MINGOTI, 2007). Neste trabalho, são retidos os componentes principais suficientes para explicar de 80 a 90 % da variância total.

6.3 CLASSIFICAÇÃO

A classificação de padrões pode ser definida como um problema relacionado com a determinação de uma fronteira de decisão que consegue distinguir diferentes padrões em classes dentro de um espaço de características *d*-dimensional, onde *d* é o número de características (DUDA et al., 2001). Então, a classificação pode ser realizada pela partição do espaço de atributos em um número finito de regiões de maneira que os objetos de uma mesma classe estejam sempre dentro de uma mesma região.

Na classificação, o desenvolvimento do classificador é abordado de forma abstrata e independente da natureza do problema, possibilitando a aplicação de técnicas em contextos variados, sem perda de sua eficiência (DUDA et al., 2001).

A grande dificuldade na implementação de um projeto de reconhecimento de padrões está na escolha da técnica adequada para que as fases desta técnica ocorram de modo a representar satisfatoriamente os fenômenos do mundo real. Assim, não há soluções gerais e a escolha do classificador é influenciada pela natureza dos dados disponíveis.

A utilização de Redes Neurais Artificiais (RNAs) pode ser vista como um método alternativo para resolução de problemas de reconhecimento de padrões, pois ao invés de criar procedimentos lógicos, a construção destas redes envolve o entendimento informal do comportamento desejado para atender ao problema (SILVA et al., 2010).

Este trabalho emprega a técnica de RNAs para a classificação dos padrões de potenciais problemas de usabilidade. Na próxima subseção serão abordados os conceitos referentes a essa técnica.

6.3.1 Classificação utilizando Redes Neurais Artificiais

As RNAs são projetadas para simular a estrutura e funcionamento do cérebro humano. São sistemas de processamento de informação intrinsecamente paralelos e distribuídos, constituídos de unidades elementares denominadas neurônios, que têm a capacidade de armazenar conhecimentos experimentais e disponibilizá-los para uso. As RNAs possuem capacidade de aprendizado e generalização (HAYKIN, 2001).

A capacidade de generalização de uma RNA também se apresenta como vantagem no problema de reconhecimento de padrões (HAYKIN, 2001; SILVA et al., 2010). A capacidade de reconhecer padrões nunca antes vistos, porém semelhantes aos apresentados na fase de treinamento é um diferencial perante muitas técnicas tradicionais, além de ajudar a superar ruídos indesejáveis nos dados de entrada do sistema de reconhecimento. As RNAs são altamente recomendáveis para lidar com sistemas de reconhecimento pouco entendidos e que não podem ser adequadamente descritos por um conjunto de regras.

De maneira geral, um conjunto de entradas é aplicado ao neurônio, que por sua vez, responde com uma saída. Cada entrada tem uma influência própria no seu peso na saída. A

conexão de vários neurônios, organizados em uma ou mais camadas, constitui uma Rede Neural Artificial.

O processamento das informações acontece por meio de cálculos matriciais matemáticos onde cada unidade de processamento imputa certo peso sináptico (W_n) ao dado de entrada (x_n) e este é processado por uma função de ativação ($f(.)$), característica de cada modelo de RNA e de cada neurônio, e assim obtém-se uma resposta (y_k) para a qual a rede foi treinada (ROSENBLATT, 1958). Esse processo é ilustrado na Figura 5.

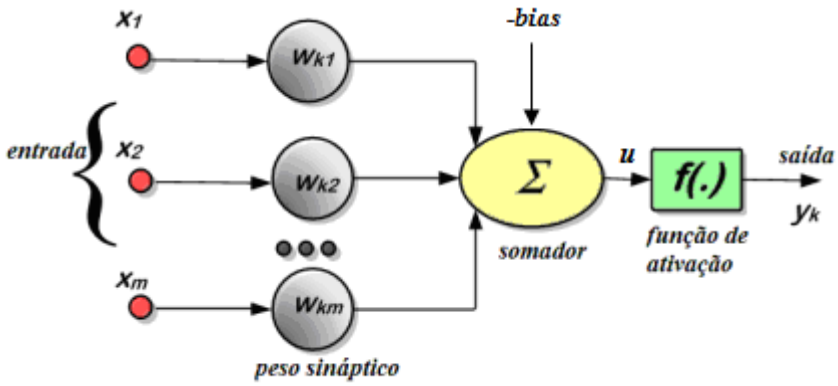

Figura 5 – Neurônio Artificial
Fonte: Adaptado de Silva et al (2001, p. 34).

O combinador linear (Σ) tem a função de agregar todos os sinais de entrada que foram ponderados pelos respectivos pesos sinápticos a fim de produzir um valor de potencial de ativação. O limar de ativação ou bias (θ) é uma variável que especifica qual será o patamar apropriado para que o resultado produzido pelo combinador linear possa gerar um valor de disparo em direção à saída do neurônio. O potencial de ativação (u) é o resultado obtido pela diferença do valor produzido entre o combinador linear e o limiar de ativação. O objetivo da função de ativação (f) é limitar a saída do neurônio dentro de um intervalo de valores (SILVA et al., 2010).

Uma rede neural extrai seu poder computacional máximo através de sua estrutura e habilidade de aprender e generalizar (HAYKIN, 2001), produzindo saídas adequadas mesmo recebendo entradas que não estavam nos processo de treinamento, podendo assim até interpretar falsas leituras sem alterar resultados. Essa capacidade de generalização, aliada à competência de aprendizado, permite o uso das redes neurais no contexto deste trabalho.

6.3.1.1 Características das redes neurais artificiais

Modelos de RNAs são especificados através da topologia da rede, características dos neurônios e regras de aprendizagem ou treinamento. O termo topologia refere-se à estrutura da rede como um todo, especificando como as entradas, saídas e as camadas escondidas são interconectadas (HAYKIN, 2001).

Neste trabalho, é utilizada a topologia *Perceptron* de Múltiplas Camadas (MLP) com arquitetura do tipo *feedforward* (propagação do sinal ocorre apenas da entrada para a saída, ou seja, é apenas no sentido positivo), pois este tipo de rede é considerado um dos mais versáteis no processo de classificação de padrões (SILVA et al., 2010).

Por serem baseadas nas redes neurais biológicas, as RNAs apresentam um surpreendente número de características observadas no processo cognitivo humano (WASSERMAN, 1989), como o aprendizado pela experiência, a generalização a partir de exemplos e a abstração de características essenciais de informações.

6.3.1.1.1 Generalização

Segundo Wasserman (1989), uma RNA é sensível às variações que podem ocorrer em informações procedentes de suas unidades de entrada, reconhecendo ruído e distorção. A capacidade da rede em se adaptar às novas situações, gerando valores de saída consistentes com os esperados, é vital para a aplicabilidade do modelo em um ambiente do mundo real.

Embora a maioria das pesquisas em RNAs tenha concentrado seus esforços na redução do tempo de aprendizagem, a característica mais importante de uma RNA é a habilidade em generalizar sobre o domínio do problema (HAYKIN, 2001).

O bom desempenho da generalização depende, entre outros fatores, do número de parâmetros livres da rede neural artificial (LECUN, 1989). É desejável diminuir o tamanho das conexões sem, entretanto, reduzir o tamanho da rede ao ponto de não se conseguir computar a função desejada.

6.3.1.1.2 Abstração

Alguns modelos de redes neurais artificiais são capazes de abstrair a essência do conjunto de dados a elas apresentados (WASSERMAN, 1989), permitindo a classificação ou reconhecimento de padrões incompletos.

6.3.1.1.3 Aprendizado

O conceito de aprendizado é definido como a habilidade de realizar novas tarefas que não podiam ser realizadas anteriormente, ou melhorar a realização de tarefas antigas, como resultado de mudanças produzidas pelo processo de aprendizado (CARBONELL, 1990).

O aprendizado ou treinamento corresponde ao processo de ajuste dos parâmetros livres da rede através de um mecanismo de apresentação de estímulos ambientais, conhecidos como padrões (dados) de entrada ou de treinamento (HAYKIN, 2001).

As Redes Neurais Artificiais podem modificar seu comportamento em resposta aos estímulos produzidos pelo ambiente, regulando a força da conexão entre unidades de processamento adjacentes pela adaptação dos pesos sinápticos, reconhecendo as informações apresentadas às suas unidades visíveis (WASSERMAN, 1989).

Conforme Másson e Wang (1990), o aprendizado em um modelo de RNAs é decorrente do treinamento da rede através da apresentação de padrões às suas unidades visíveis. O objetivo do treinamento consiste em atribuir os pesos sinápticos com valores apropriados, de modo a produzir o conjunto de saídas desejadas ou ao menos consistentes com um intervalo de erro estabelecido (FREEMAN; SKAPURA, 1991). Dessa forma, o processo de aprendizado consiste na busca de um espaço de pesos pela aplicação de alguma regra que defina esta aprendizagem.

Denomina-se algoritmo de aprendizado o conjunto de regras bem definidas para a solução de um problema de aprendizado. Existem muitos tipos de algoritmos de aprendizado específicos para determinados modelos de RNAs, estes algoritmos diferem entre si principalmente pelo modo como os pesos são modificados. Existem basicamente três paradigmas de aprendizado:

- Aprendizado por reforço – quando um crítico externo avalia a resposta fornecida pela rede;
- Aprendizado supervisionado – a rede é treinada pela apresentação dos vetores de entrada e seus respectivos vetores de saída, chamados de pares de treinamento;
- Aprendizado não supervisionado – o treinamento consiste em apresentar apenas os vetores de entrada, a partir dos quais são extraídas as características desse conjunto de padrões, agrupando-os em classes.

Os algoritmos de classificação supervisionada subdividem-se em paramétricos e não paramétricos. O classificador paramétrico é treinado com uma grande quantidade de amostras rotuladas (padrões cujas classes são conhecidas a priori) para que se possam estimar os parâmetros estatísticos de cada classe de padrão. Pode-se citar como exemplos de classificadores supervisionados: distância mínima; máxima verossimilhança e o algoritmo *Backpropagation*.

Na classificação não paramétrica, os parâmetros estimados do conjunto de treinamento não são levados em consideração. Um exemplo de classificador não paramétrico é o K-vizinhos mais próximos. Na classificação não supervisionada, o classificador particiona o conjunto de dados de entrada a partir de algum critério de similaridade, resultando em um conjunto de agrupamentos (*clusters*), cada um dos quais normalmente associado a uma classe.

Em geral, as regras de aprendizado podem ser consideradas variantes da Regra de Hebb[10] (HEBB, 1949). Na essência, Hebb propõe que a sinapse conectando dois neurônios seja reforçada sempre que ambos os neurônios estiverem ativos. Uma RNA que utilize a regra de Hebb para o processo de aprendizado modifica os pesos sinápticos entre as conexões das unidades de processamento proporcionalmente ao produto dos níveis de excitação desses neurônios. Em termos matemáticos, as regras de ajuste dos pesos sinápticos (w_i) e do limiar de ativação (θ) do neurônio podem ser expressas, respectivamente, pelas Equações (9) e (10).

$$w_i^{(atual)} = w_i^{(anterior)} + \eta.(d^{(k)} - y).x^{(k)} \quad (9)$$

$$\theta_i^{(atual)} = \theta_i^{(anterior)} + \eta.(d^{(k)} - y).x^{(k)} \quad (10)$$

[10] Em 1949, Donald Hebb mostrou como a plasticidade da aprendizagem de redes neurais é conseguida através da variação dos pesos de entrada dos neurônios. Esta teoria deu origem à chamada "Regra de Hebb", utilizada em vários algoritmos de treinamento de Redes Neurais Artificiais.

onde:

w_i – vetor de pesos;

η – constante que define a taxa de aprendizado da rede;

$d^{(k)}$ – valor desejado para a *k*-ésima amostra de treinamento;

y – valor da saída produzida pela rede;

$x^{(k)}$ – *k*-ésima amostra de treinamento.

Entretanto, em termos de implementação computacional, torna-se mais conveniente tratar as expressões anteriores em sua forma vetorial. Como a mesma regra de ajuste é aplicada tanto para os pesos sinápticos como para o limiar, pode-se então inserir o valor do limiar de ativação dentro do vetor de pesos sinápticos. De fato, o valor do limiar é também uma variável que deve ser ajustada a fim de se realizar o treinamento. Portanto, as Equações (9) e (10) podem ser representadas por uma única expressão vetorial dada pela Equação (11):

$$w = w + \eta.(d^{(k)} - y).x^{(k)} \tag{11}$$

Como adaptação à regra de Hebb, a regra Delta modifica os pesos de acordo com a variação entre a saída desejada e a saída observada no treinamento (SILVA et al., 2010). A ideia básica envolvida com a aplicação da regra Delta está em minimizar a diferença entre a saída desejada (d) e a resposta do combinado linear (u). Mais especificamente, utiliza-se a minimização do erro quadrático médio entre *u* e *d* com o objetivo de ajustar o vetor de pesos (*w*) da rede. O objetivo consiste em obter um *w* ótimo, tal que o erro quadrático seja o mínimo possível. A Equação (12) atualiza os pesos de uma RNA pela aplicação da regra Delta.

$$w = w + \eta.(d^{(k)} - u).x^{(k)} \tag{12}$$

6.3.1.2 Redes *MultiLayer Perceptron*

As redes *MultiLayer Perceptron* (MLP), ou *Perceptron* de Múltiplas Camadas tem por base a estrutura apresentada por Rosenblatt (1958), porém apresenta uma ou mais camadas intermediárias conhecidas como camadas ocultas.

Cada uma das camadas da MLP possui uma função específica. A camada de entrada é composta por unidades sensoriais e é responsável pela recepção e propagação das informações de entrada para a camada seguinte, sem modificá-las. As camadas intermediárias são compostas por unidades computacionais que transmitem as informações por meio das conexões entre as unidades de entrada e saída; estas conexões armazenam os pesos que serão multiplicados pelas entradas, garantindo o conhecimento da rede. As camadas intermediárias também funcionam como detectores de características. A camada de saída é formada por neurônios computacionais que recebem os dados da camada intermediária fornecendo a resposta à rede. A estrutura típica de um MLP é ilustrada na Figura 6.

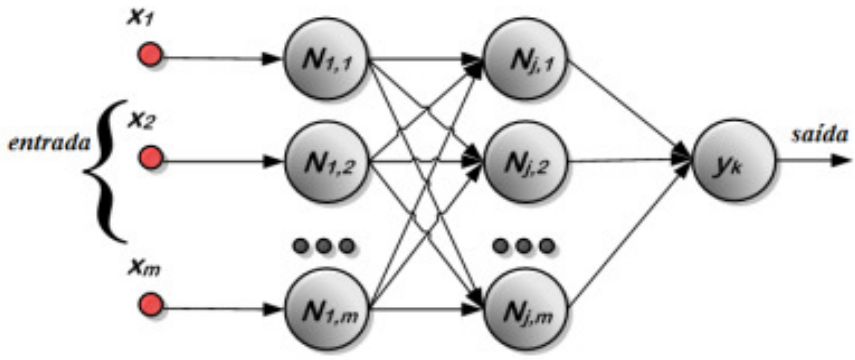

Figura 6 – *Perceptron* de Múltiplas Camadas
Fonte: Adaptado de Silva et al (2010, p. 92).

Quanto ao número de neurônios da camada oculta, o uso de muitas unidades pode levar a rede a memorizar os padrões de treinamento em vez de extrair as características gerais que permitirão a generalização ou o reconhecimento de padrões não vistos durante o treinamento. O uso de um número muito pequeno pode forçar a rede a gastar tempo em excesso tentando encontrar uma representação ótima.

Na tentativa de contornar esse problema, neste trabalho é utilizada a técnica de validação cruzada (*cross-validation*) (HAYKIN, 2001; SILVA et al., 2010) para a definição

da topologia mais indicada e com maior desempenho para o problema em questão. Além disso, também é aplicado o método denominado de validação cruzada por amostragem aleatória (SILVA et al., 2010), no qual o conjunto total de dados é aleatoriamente dividido em dois subconjuntos: treinamento e validação.

Nas redes MLP, o sinal de saída de cada neurônio é o resultado da aplicação da função de ativação sobre a soma ponderada dos sinais de entrada. A função de ativação representa o efeito que a entrada interna e o estado atual de ativação exercem na definição do próximo estado de ativação da unidade. Para este trabalho, é utilizada a função de ativação denominada logística sigmoidal, que apresenta a característica de ser derivável em todo o seu intervalo, visualizada na Equação (13).

$$F(x) = \frac{1}{1 + e^{-bx}} \tag{13}$$

onde *b* determina a inclinação da função.

6.3.1.3 Algoritmo de treinamento *Backpropagation*

Existem diferentes tipos de algoritmos de aprendizado específicos para determinados modelos de RNAs, os quais diferem entre si principalmente pela maneira como os pesos são modificados. O algoritmo de treinamento da rede MLP é baseado no treinamento supervisionado. Neste trabalho, a rede MLP é treinada conforme o Algoritmo *Backpropagation* com gradiente descendente.

O algoritmo *Backpropagation* é a generalização da Regra Delta proposta por Widrow-Hoff (SILVA et al., 2010). Este algoritmo utiliza o princípio da minimização de uma função custo, no caso, a soma dos erros quadráticos médios sobre um conjunto de treinamento, utilizando a técnica de gradiente descendente (HAYKIN, 2001).

A principal modificação em relação à Regra Delta é a utilização de funções contínuas como função de saída dos neurônios ao invés da função de limiar lógico. Por serem deriváveis, o uso de funções contínuas permite a utilização da busca do gradiente descendente também para os elementos das camadas intermediárias.

Neste algoritmo, é apresentado à rede um conjunto de padrões de treinamento, composto dos valores de entrada e as respectivas saídas desejadas. O objetivo do treinamento

é ajustar os pesos sinápticos de forma a minimizar o erro entre a resposta desejada e a saída real da rede.

O funcionamento do algoritmo *Backpropagation* pode ser explicado da seguinte forma: os sinais de entrada são propagados pela rede, camada por camada, até a apresentação do resultado na camada de saída. Em seguida, o resultado obtido é comparado com o desejado; se a saída obtida não estiver correta, o erro é calculado. O erro, então, é retropropagado da camada de saída para a camada de entrada. Em resumo, as fases de propagação e retropropagação do erro fazem com que os pesos sinápticos e limiares dos neurônios se ajustem automaticamente em cada interação, implicando na gradativa diminuição dos erros produzidos pelas respostas da rede (SILVA et al., 2010).

O algoritmo *Backpropagation* baseia-se no Aprendizado Supervisionado por Correção de Erros, constituído de:

a) Propagação: Depois de apresentado o padrão de entrada, a resposta de uma unidade é propagada como entrada para as unidades da camada seguinte, até a camada de saída, onde é obtida a resposta da rede e o erro é calculado. Esse processo pode ser visualizado na Figura 7.

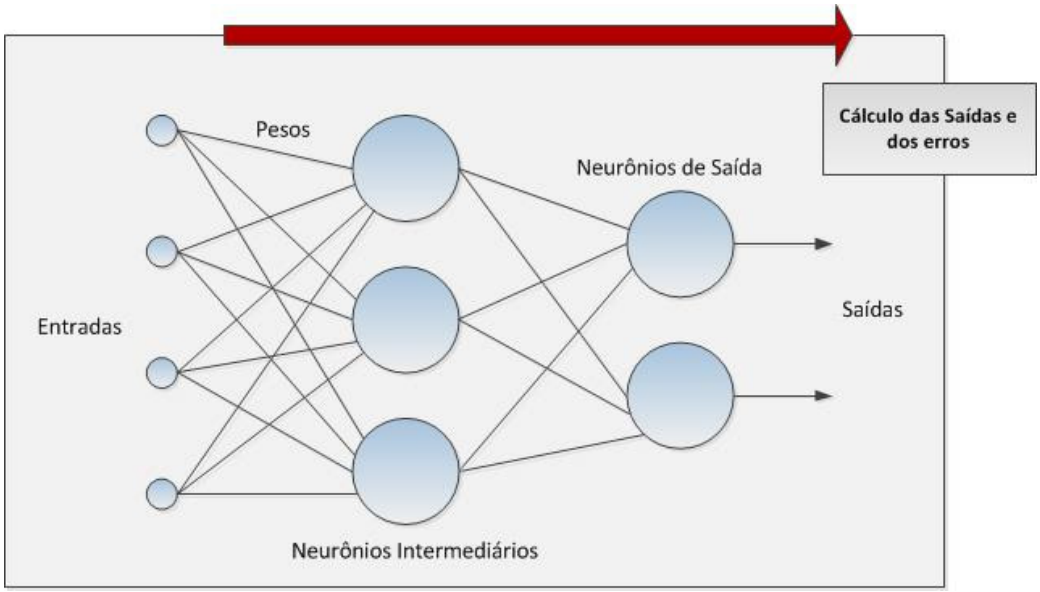

Figura 7 – Fase de Propagação
Fonte: Adaptado de Silva et al. (2010, p. 94).

b) Retropropagação: Desde a camada de saída até a camada de entrada, são feitas alterações nos pesos sinápticos, conforme Figura 8.

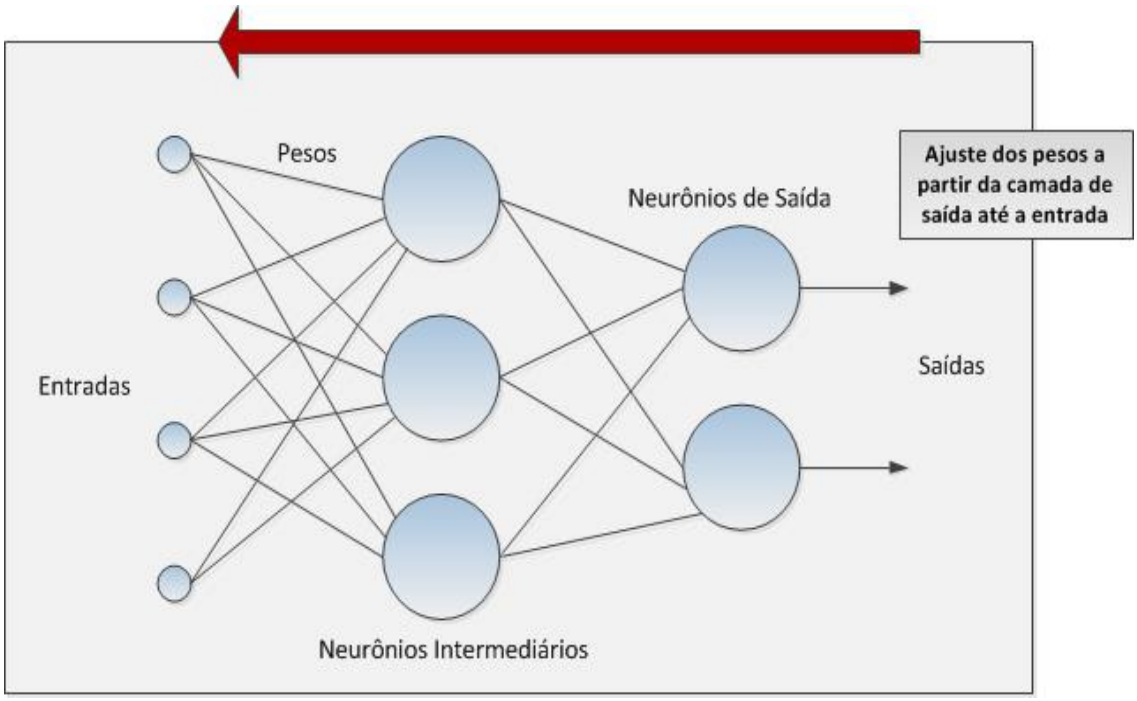

Figura 8 – Fase de Retropropagação
Fonte: Adaptado de Silva et al. (2010, p. 94).

Durante a fase treinamento deve-se apresentar um conjunto formado pelo par: entrada da rede e valor desejado de resposta. A saída é comparada ao valor desejado e é computado o erro global, que influencia na correção dos pesos no passo de retropropagação. Apesar de não existir garantias que a rede forneça uma solução ótima para o problema, esse processo é muito utilizado por apresentar uma boa solução para o treinamento de *Perceptron* Múltiplas Camadas.

O algoritmo *Backpropagation* pode ser resumido em:

a) Inicialização – inicialize os pesos sinápticos (w) e o limiar de ativação (θ) aleatoriamente, com valores no intervalo [-1;1];

b) Apresentação dos exemplos de treinamento – para cada exemplo do conjunto de treinamento, efetue os passos *c* e *d*;

c) Computação para frente (Propagação) – Supondo que um exemplo de treinamento da época seja representado por { $x(n)$, $d(n)$ }, com o vetor de entrada $x(n)$ aplicado à camada

de entrada de nós sensoriais e o vetor de resposta desejada seja *d(n)*. Calcule os campos locais induzidos e os sinais funcionais da rede prosseguindo para frente através da rede, camada por camada. O campo local induzido $v_j^{(l)}(n)$ para o neurônio *j* na camada *l* é dado pela Equação (14):

$$v_j^{(l)}(n) = \sum_{i=0}^{m0} w_{ji}^{(l)}(n).y_i^{(l-1)}(n) \qquad (14)$$

onde:

$y_i^{(l-1)}(n)$ – sinal (função) de saída do neurônio *i* na camada anterior *l-1*, na interação *n;*

$w_{ji}^{(l)}(n)$ – peso sináptico do neurônio *j* da camada *l*, que é alimentado pelo neurônio *i* da camada *l-1*.

Para *i* = 0, tem-se: $y_0^{(l-1)}(n) = +1$ e $w_{j0}^{(l)}(n) = b_j^{(l)}(n)$ é o limiar de ativação aplicado ao neurônio *j* na camada *l*. Assumindo-se o uso de uma função sigmoidal, o sinal de saída do neurônio *j* na camada *l* é calculado pela Equação (15):

$$y_j^{(l)} = \varphi_j(v_j(n)) \qquad (15)$$

Se o neurônio *j* está na primeira camada oculta (por exemplo, *l* = 1), o cálculo é efetuado de acordo com a Equação (16):

$$y_j^{(0)}(n) = x_j(n) \qquad (16)$$

onde $x_j(n)$ é o *j*-ésimo elemento do vetor de entrada *x(n)*.

Se o neurônio *j* está na camada de saída (por exemplo, *l* = *L*, onde *L* é denominado como a profundidade da rede), aplique a Equação (17):

$$y_j^{(L)} = o_j(n) \qquad (17)$$

Assim, o sinal de erro é calculado de acordo com a Equação (18):

$$e_j(n) = d_j(n) - o_j(n) \tag{18}$$

onde *d(n)* é o *j*-ésimo elemento do vetor de resposta desejada *d(n)*.

d) Computação para trás (Retropropagação) – Calcule os gradientes locais (δ) da rede, definidos pela Equação (19):

$$\delta_j^l(n) = \begin{cases} e_j^L(n)\varphi_j'(v_j^{(L)}(n)) & \text{Para o neurônio } j \text{ da camada de saída } L. \\ \varphi_j'(v_j^{(l)}(n))\sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n) & \text{Para o neurônio } j \text{ na camada oculta } l. \end{cases} \tag{19}$$

onde:

$\varphi_j'(.)$ – representa a diferenciação em relação ao argumento;

$e_j^L(n)$ – sinal do erro;

$\delta_k^{(l+1)}$ – erro das unidades da camada anterior conectada a unidade *j*;

$w_{kj}^{(l+1)}$ – pesos das conexões com a camada anterior.

Ajuste os pesos sinápticos da rede na camada *l* de acordo com a regra delta generalizada, através da Equação (20):

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \eta \delta_j^{(l)}(n) y_i^{(l-1)}(n) \tag{20}$$

onde η é o parâmetro da taxa de aprendizagem.

e) Iteração **–** Itere as computações para frente e para trás dos itens *c* e *d*, apresentando novas épocas de exemplos de treinamento para a rede, até que seja satisfeito o critério de parada.

As RNAs que utilizam o algoritmo *Backpropagation* podem ser vistas como "caixas pretas", pois não se sabe exatamente como a rede chega a um determinado resultado, uma vez que os modelos não apresentam justificativas para suas respostas (HAYKIN, 2001). Nesse sentido, muitas pesquisas vêm sendo realizadas visando à extração de conhecimento de RNAs e na criação de procedimentos explicativos, onde se tenta justificar o comportamento da rede em determinadas situações.

Outra limitação refere-se ao tempo de treinamento de RNAs utilizando *Backpropagation*, que tende a ser muito lento. Algumas vezes são necessárias milhares de interações para se chegar a níveis de erros aceitáveis, o que pode ser problemático em redes muito grandes ou com grande quantidade de dados.

É difícil definir a arquitetura ideal da rede de forma que ela seja tão grande quanto o necessário para conseguir obter as representações necessárias e ao mesmo tempo pequena o suficiente para que o treinamento seja mais rápido. Não existem regras claras para se definir quantas unidades devem existir nas camadas intermediárias, o número exato de camadas intermediárias ou como devem ser as conexões entre estas unidades.

Apesar dessas limitações, o *Backpropagation* fornece um método computacional eficiente para o treinamento de *Perceptrons* de múltiplas camadas.

7 IDENTIFICAÇÃO AUTOMÁTICA DE PROBLEMAS DE USABILIDADE EM INTERFACES DE SISTEMAS WEB

Neste Capítulo, é apresentada a aplicação de técnicas de Reconhecimento de Padrões (RP) na identificação e classificação automática de potenciais problemas de usabilidade na interface de um sistema web. As tarefas realizadas pelo usuário são obtidas através da análise da interação do usuário armazenada em arquivos de *log*. As tarefas são representadas por meio de um modelo de tarefas.

Um sistema clássico de RP engloba três grandes etapas (DUDA et al., 2001): representação dos dados de entrada, extração de características e classificação dos padrões. O desenvolvimento deste Capítulo está organizado de acordo com as fases do RP e são descritas as atividades realizadas em cada etapa.

Na etapa de representação dos dados de entrada é realizada a coleta dos dados referentes à interação do usuário com um sistema web para a seleção dos atributos mais importantes do arquivo de *log*, descartando aqueles não relevantes para a identificação de problemas de usabilidade. Nessa etapa também é desenvolvido um modelo de tarefas que representa a sequência desejada de execução da tarefa por parte do usuário. Adicionalmente, é realizada a filtragem dos arquivos de *log* com o objetivo de selecionar apenas as entradas destes arquivos que representam a interação do usuário com o sistema web. Finalmente, é realizada a mensuração e normalização dos dados do arquivo de *log*, transformando esses dados em valores numéricos normalizados e necessários para a formação de um vetor de entrada, utilizado nas etapas seguintes do RP.

Na etapa de extração de características é implementada a técnica PCA que reduz a dimensionalidade do conjunto original de dados em um número menor de componentes, responsável pela representação dos dados originais.

Na etapa de classificação é implementada uma RNA do tipo MLP, responsável pela classificação dos padrões. Também é realizada a interpretação ou decodificação das saídas da rede, atribuindo cada saída a uma das 6 classes específicas que representam potenciais problemas de usabilidade.

A Figura 9 ilustra as técnicas de RP empregadas neste trabalho, assim como as atividades desenvolvidas em cada etapa.

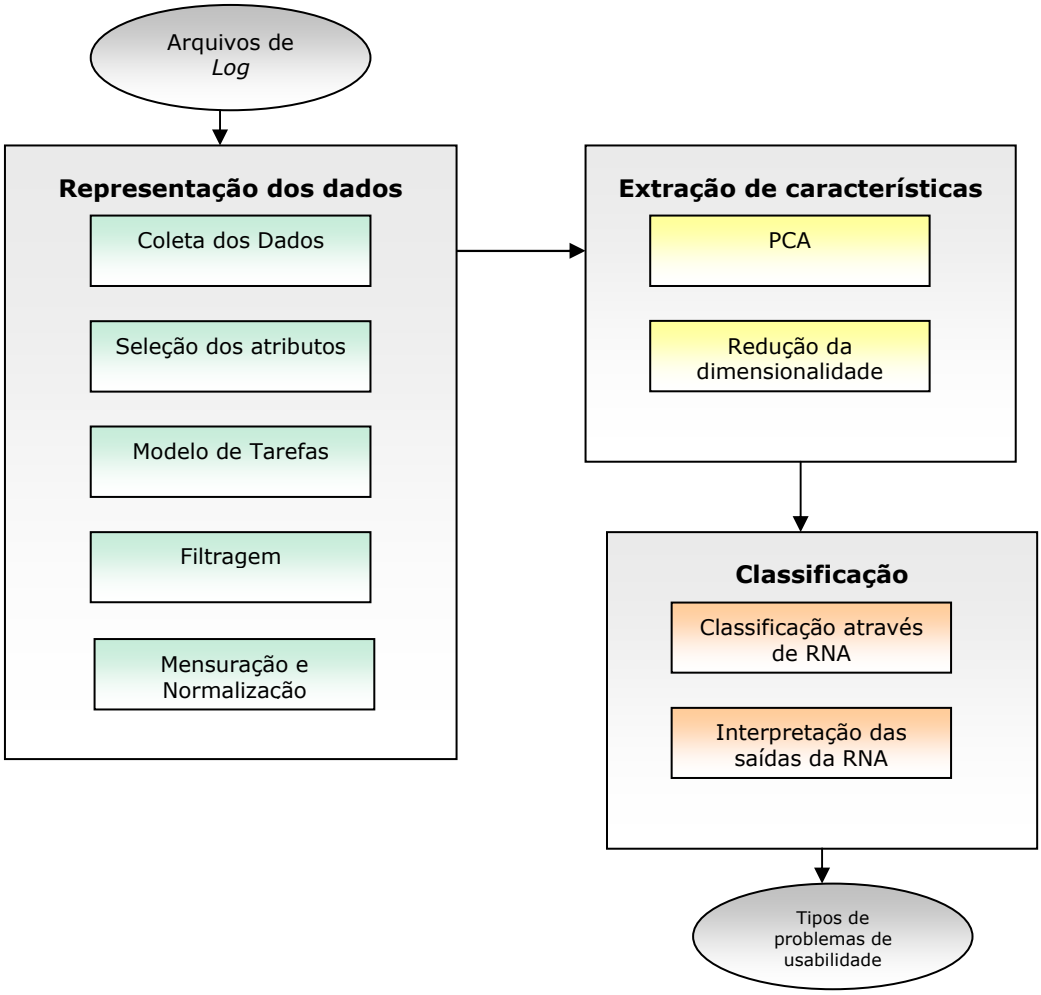

Figura 9 – Tarefas de cada etapa do RP
Fonte: Autoria própria.

Nas próximas seções serão abordadas as atividades desenvolvidas e os resultados obtidos em cada etapa.

7.1 REPRESENTAÇÃO DOS DADOS DE ENTRADA

Nesta etapa é realizada a representação dos dados de entrada armazenados nos arquivos de *log* gerados pela ferramenta WAUTT. Os dados são referentes à interação do usuário com um sistema web. A primeira fase do RP é resumida nos seguintes passos:

- Coleta dos dados referentes à interação do usuário com o sistema web;

- Seleção dos atributos relevantes dos arquivos de *log*;
- Desenvolvimento de um modelo de tarefas;
- Filtragem dos arquivos de *log* para eliminação de informações irrelevantes;
- Organização dos dados sobre a ocorrência de comportamentos nas tarefas, através da mensuração e normalização dos dados de entrada.

Nas próximas subseções esses passos são apresentados, assim como os resultados.

7.1.1 Coleta dos Dados

Para os testes da abordagem proposta foi desenvolvido um sistema web que possibilitou a captura da interação do usuário na realização da tarefa especificada. Esse sistema web representa uma agência fictícia de empregos e disponibiliza serviços referentes ao cadastro de currículos de usuários em diferentes áreas de atuação. A página inicial do sistema web pode ser visualizada na Figura 10.

Figura 10 – Página inicial (index.php)
Fonte: Autoria própria.

Para obter dados referentes à interação com o sistema web foram realizados testes com usuários. Os participantes dos testes com usuários realizaram a tarefa "Cadastrar currículo", resultando em 120 sequências desta tarefa, resultando em um conjunto de amostras bastante rico de situações de sucesso e fracasso na realização da tarefa proposta.

O sistema web foi desenvolvido na linguagem PHP e foi utilizado o Apache como servidor web, devido à sua popularidade e fácil administração. Além disso, trata-se de um servidor que se comunica com o interpretador de páginas PHP. O sistema foi testado e as interações do usuário puderam ser capturadas pela ferramenta WAUTT. Como sistema gerenciador de banco de dados (SGBD) foi empregado o PostgreSQL[11], pois este possui ferramentas próprias para o gerenciamento dos dados, possibilitando consultas SQL[12] sobre a base de dados. Além dessas características, esse SGBD pode ser acessado através de códigos PHP.

Cabe ressaltar que esse sistema web foi desenvolvido sem o uso de recursos, como máscaras, rótulos, mensagens explicativas etc., com o objetivo de induzir o usuário a cometer erros na realização da tarefa, viabilizando a análise de diversos tipos de problemas de usabilidade.

7.1.2 Seleção dos Atributos do Arquivo de *Log*

Alguns elementos do arquivo de *log* foram descartados por não terem relevância para a identificação de tarefas, como por exemplo: *browser*, sistema operacional, título (*title*) etc. Os eventos relacionados ao movimento de mouse e redimensionamento da janela também não são considerados, pois não tiveram relevância e influência nos testes realizados.

Assim, a seleção de apenas alguns atributos (características) foi realizada inicialmente para a eliminação de dados não significativos, evitando a geração de resultados e análises inconsistentes.

A Tabela 2 ilustra os atributos do arquivo de *log* considerados mais importantes. Os atributos "horário" e "data de solicitação" são utilizados apenas para o cálculo do tempo total de realização de uma tarefa. O atributo IP serve para a identificação de usuários específicos.

[11] Disponível para *download* gratuito em: http://www.postgresql.org/
[12] É uma linguagem de pesquisa declarativa para banco de dados relacional.

Assim, os atributos mais relevantes para a definição do MT são: documento solicitado, evento, elemento e valor.

Tabela 2 – Atributos selecionados do arquivo de *log*

Campos	Exemplos de valores
Data da solicitação	05/06/2012
Horário da solicitação	13:50:12
Endereço do IP	127.0.0.1
Documento solicitado	Index.php
Evento	Clique do mouse
Elemento	Input (Botão)
Valor/Texto	Voltar

Fonte: Autoria própria.

7.1.3 Modelo de Tarefas

A ideia subjacente ao modelo de tarefas desenvolvido é semelhante ao ambiente denominado Ergo-Monitor (MORANDINI et al., 2007). O modelo proposto baseia-se na modelagem das tarefas executadas pelo usuário na interação com um sistema web e posterior análise do comportamento do usuário perante estas tarefas. Porém, a análise dos arquivos de *log* para a definição do modelo de tarefas é realizada considerando os eventos realizados em cada elemento da página web e não apenas com base nas páginas visitadas (como no ambiente Ergo-Monitor).

Devido à estrutura não linear das interfaces web baseadas em hipertexto, um sistema web é composto por várias páginas que são formadas por elementos como *links*, imagens, formulários, botões etc. Entende-se por "evento" a interação do usuário com um elemento de uma página web utilizando o *mouse*, teclado ou outro dispositivo de entrada de dados.

De maneira formal, uma tarefa k é definida pela seguinte tupla: $t_k = \{E, p_o, p_f, temp\}$, na qual E é um conjunto, de tamanho n, de eventos que devem ser realizados pelo usuário em cada página para a tarefa k; p_o corresponde à página inicial; p_f é a página final que identifica

o sucesso na realização da tarefa; *temp* é o tempo gasto para execução da tarefa. Assim, pode-se definir um Modelo de Tarefas (MT) como: MT = { t_1, t_2, ..., t_m }, onde m é a quantidade de tarefas do modelo.

Outro fato importante a ser considerado diz respeito às páginas que compõem o MT. Essas páginas (URLs) podem estar representadas em mais de uma entrada no arquivo de *log*, uma vez que o usuário pode realizar vários eventos dentro de uma única página.

Na página inicial do sistema web o usuário tem algumas opções para cadastrar seu currículo através dos *links*: "Currículos", "Cadastre seu currículo", "Estagiários" e "Profissionais". Todos esses *links* direcionam o usuário imediatamente à página de cadastro de currículo (cadastro.php), apresentada na Figura 11.

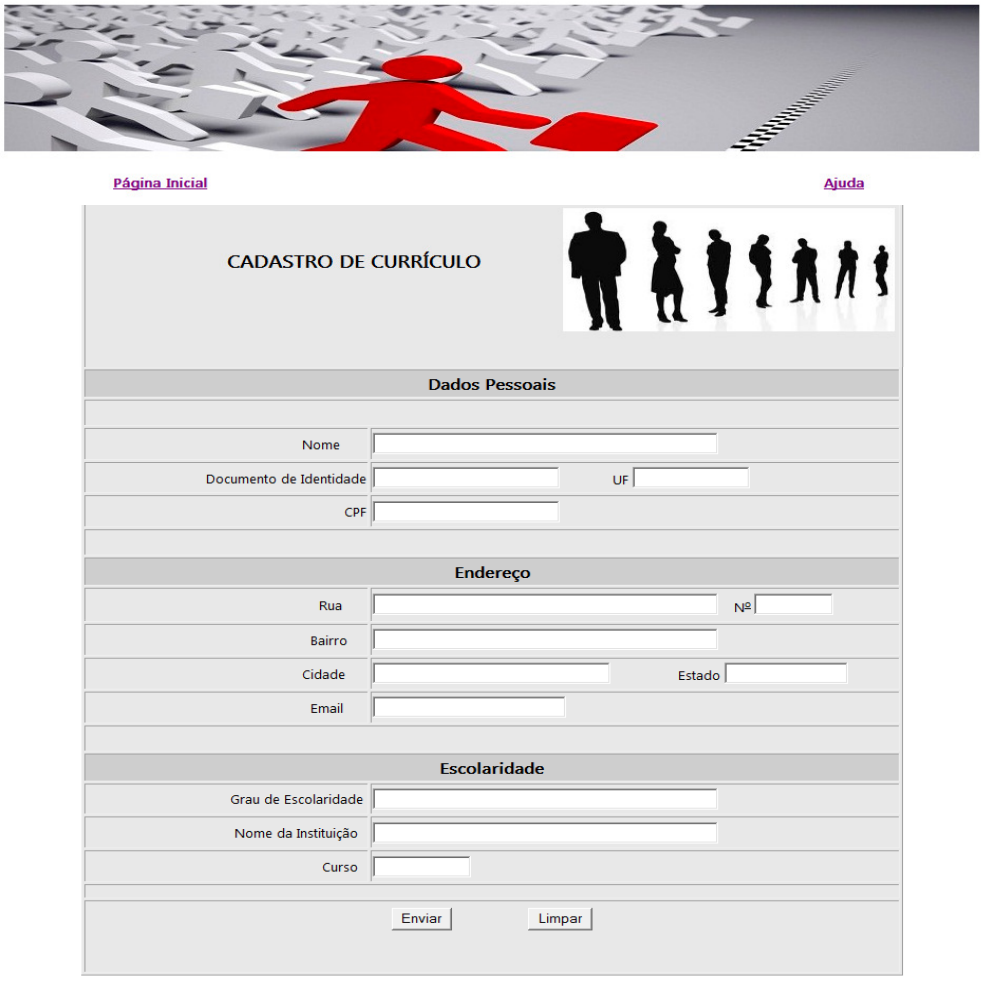

Figura 11 – Página de Cadastro (cadastro.php)
Fonte: Autoria própria.

O usuário, com o objetivo de cadastrar seu currículo no sistema web, tem como possível caminho o acesso aos arquivos: “index.php”, “cadastro.php” e finalmente “sucesso.html”. No entanto, ao defrontar-se com um problema na página “cadastro.php”, por exemplo, o usuário tem a opção de consultar a página de ajuda, apresentada na Figura 12.

Figura 12 – Página de Ajuda (ajuda.php)
Fonte: Autoria própria.

Se o problema persistir e o usuário insistir em se cadastrar, este será direcionado para a página de erro (erro.php) ilustrada na Figura 13.

Figura 13 – Página de Erro (erro.php)
Fonte: Autoria própria.

O usuário somente terá êxito em seu objetivo se conseguir atingir a página de sucesso (sucesso.html) ilustrada na Figura 14.

Figura 14 – Página de Sucesso (sucesso.html)
Fonte: Autoria própria.

A ferramenta WAUTT é empregada para a captura das interações do usuário e geração dos arquivos de *log* para posterior comparação entre a sequência realizada pelo usuário e a sequência de eventos desejada, isto é, a tarefa definida no MT. Assim, na coleta de dados para a definição de tarefas, além de considerar o caminho (URLs) percorrido pelo usuário, também são considerados os eventos que representam decisões do usuário durante a interação. Além disso, também é considerado o tempo total empregado na execução da tarefa.

Como exemplo, o MT com os eventos esperados para a realização da tarefa "Cadastrar currículo" está definido da seguinte maneira:

E = { load – index.php;
click (IMG) – index.php;
load – cadastro.php;
click (FORM) – cadastro.php;
click (INPUT) – cadastro.php;
load – ajuda.html;
load – erro.php;
load – sucesso.html }

p_o = { index.php }

p_f = { sucesso.html }

temp = em segundos

A Figura 15 ilustra o diagrama de Análise Hierárquica de Tarefas (AHT), no qual a tarefa “Cadastrar currículo” é decomposta em vários níveis, também chamados de subtarefas.

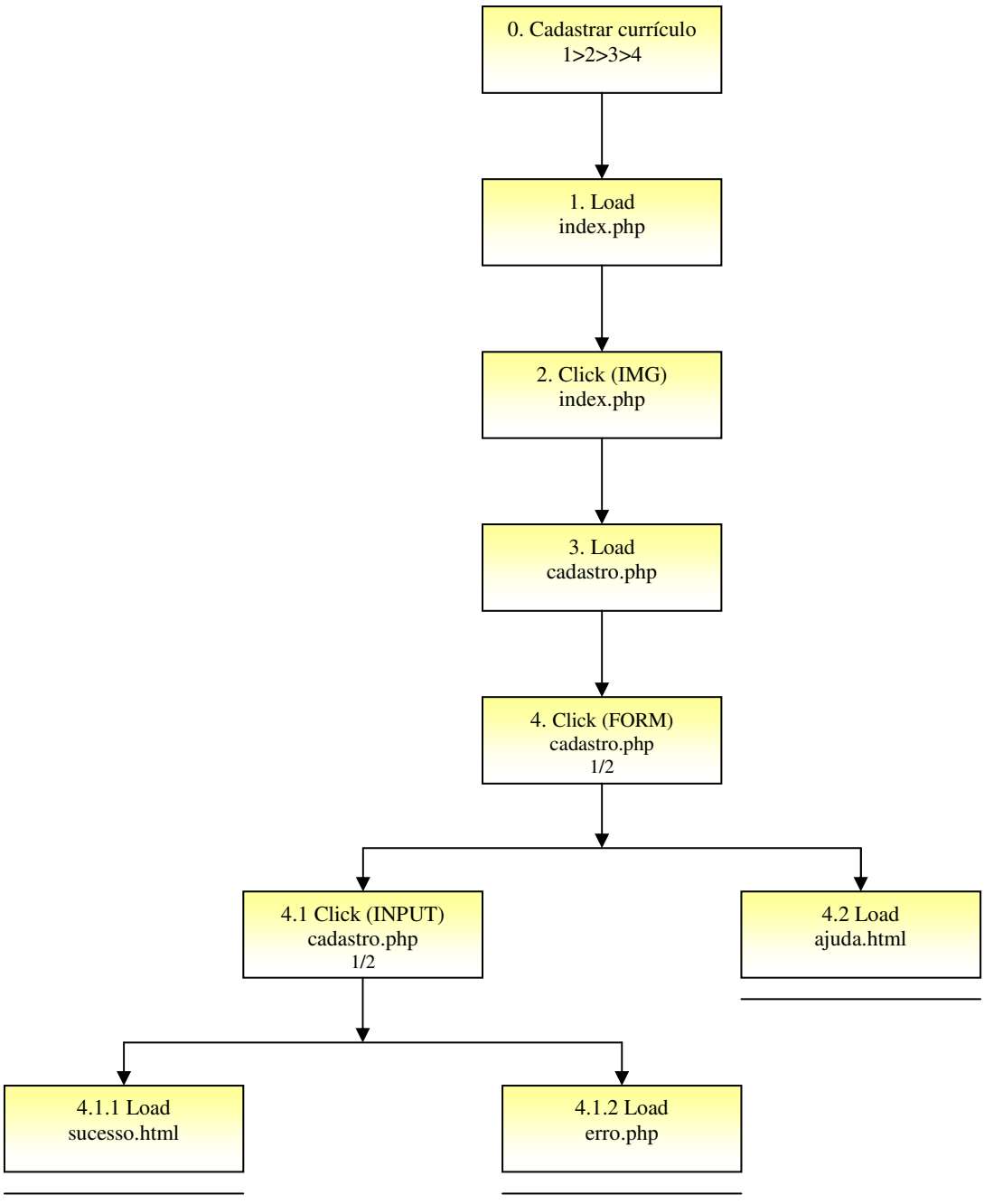

Figura 15 – Diagrama AHT (Cadastrar currículo)
Fonte: Autoria própria.

O evento “click (FORM)” está relacionado à quantidade de cliques em caixas de entrada. Assim, o número de vezes que este evento está presente no arquivo de *log* varia de acordo com o número de caixas de entrada implementado no formulário. Para o formulário da página “cadastro.php” foram implementados 13 campos para entrada de dados. Portanto, o evento “click (FORM)” consta em 13 entradas do arquivo de *log*, diferenciando-se apenas em relação ao atributo “valor”, que define o nome do campo.

Para cada combinação de eventos de uma tarefa (padrão) foi associada uma saída representando um tipo de problema de usabilidade e indicando que aquela combinação possivelmente não respeitou as recomendações específicas daquele padrão. Os padrões foram divididos de acordo com as recomendações para o projeto de formulários (Seção 3.2):

1) Navegação;
2) Frequência de interação;
3) Entrada de Dados;
4) Layout;
5) Funcionalidades; e
6) Ausência de problemas de usabilidade.

Com relação às classes, é importante ressaltar que a classe 6 (ausência de problemas de usabilidade) representa uma classe neutra, pois não se pode afirmar a total ausência de problemas de usabilidade, apenas que eles não foram identificados.

Na Tabela 3, é apresentada a associação de eventos às suas respectivas classes de problemas de usabilidade.

Tabela 3 – Relação: Problemas / Eventos

Nº da Classe	**Nome da Classe**	**Evento**
1	Navegação	*Focus*
2	Frequência de Iteração	*Tempo*
3	Entrada de dados	*Click – Form*
4	Layout	*Scroll*
5	Funcionalidade	*Click – Input*
6	Ausência de Problemas	--

Fonte: Autoria própria.

Após a etapa de registro do MT, torna-se necessário desenvolver um mecanismo que realize a filtragem, busca e a organização por tarefas no arquivo de *log* baseado nas tarefas registradas no MT. Na próxima subseção é apresentado o mecanismo de filtragem dos arquivos de *log* empregado, que tem por objetivo eliminar as entradas que não representam interações reais do usuário com o sistema web.

7.1.4 Filtragem dos Arquivos de *Log*

A ideia principal do processo de filtragem é analisar o Modelo de Tarefas, o arquivo de *log* e armazenar num arquivo de *log* refinado somente as linhas comuns ao MT e ao arquivo de *log* original. Portanto, o arquivo de *log* filtrado é constituído somente pelas entradas identificadas nos dois arquivos.

Neste trabalho, a filtragem dos arquivos de *log* foi realizada manualmente, levando em consideração somente as entradas significativas para a tarefa "Cadastrar currículo". No Quadro 3, é apresentado um trecho de um arquivo de *log* que armazena o acesso a uma única página do sistema web, "index.php". Esse Quadro apresenta o acesso a 9 (nove) recursos que constituem apenas uma única página (index.php). Para esse trecho é necessário retirar do arquivo de *log* todos os recursos embutidos na página web em questão, pois as entradas referentes aos carregamentos de figuras ou vídeos não eram significativas para a execução da tarefa "Cadastrar currículo".

*13:50:12,127.0.0., http://localhost/Php/index.php, event:**load**,width:1024, height:477*
13:51:00, 127.0.0.,http://localhost/Php/index.php, event:, "GET /css/styles.css
13:51:05, 127.0.0.,http://localhost/Php/index.php, event:, "GET /logo.ico
13:51:09, 127.0.0.,http://localhost/Php/index.php, event:, "GET /images/job.jpg
13:51:09, 127.0.0.,http://localhost/Php/index.php, event:, "GET /images/m1.jpg
13:51:09, 127.0.0.,http://localhost/Php/index.php, event:, "GET /images/m2.jpg
13:51:09, 127.0.0.,http://localhost/Php/index.php, event:, "GET /images/m3.jpg
13:51:10, 127.0.0.,http://localhost/Php/index.php, event:, "GET /images/m4.jpg
13:51:10, 127.0.0.,http://localhost/Php/index.php, event:, "GET /images/m5.jpg

Quadro 3 – Trecho de um arquivo de *log* (carregamento da página index.php)
Fonte: Autoria própria.

O Quadro 4 apresenta um trecho do arquivo de *log* filtrado que exemplifica a execução com sucesso da tarefa “Cadastrar currículo”.

13:50:12, 127.0.0., http://localhost/Php/index.php, event: ***load***

13:51:00, 127.0.0., http://localhost/Php/index.php, event: ***click****:, element: IMG, value/text:Cadastro*

13:51:55, 127.0.0., http://localhost/Php/cadastro.php, event: ***load***

13:52:12, 127.0.0., http://localhost/Php/cadastro.php, event: ***click****:, element: FORM:, value/text:Nome*

13:52:40, 127.0.0., http://localhost/Php/cadastro.php, event: ***click****:, element: FORM, value/text:RG*

13:53:30, 127.0.0., http://localhost/Php/cadastro.php, event: ***click****:, element: FORM, value/text:UF1*

13:53:52, 127.0.0., http://localhost/Php/cadastro.php, event: ***click****:, element: FORM, value/text:CPF*

13:54:20, 127.0.0., http://localhost/Php/cadastro.php, event: ***click****:, element: FORM, value/text:Rua*

13:55:22, 127.0.0., http://localhost/Php/cadastro.php, event: ***click****:, element: FORM, value/text:Num*

13:55:42, 127.0.0., http://localhost/Php/cadastro.php, event: ***click****:, element: FORM, value/text:Bairro*

13:56:34, 127.0.0., http://localhost/Php/cadastro.php, event: ***click****:, element: FORM, value/text:Cidade*

13:56:34, 127.0.0., http://localhost/Php/cadastro.php, event: ***click****:, element: FORM, value/text:UF2*

13:57:20, 127.0.0. http://localhost/Php/cadastro.php, event: ***click****:, element: FORM, value/text:Email*

13:57:50, 127.0.0., http://localhost/Php/cadastro.php, event: ***click****:, element: FORM, value/text:Grau*

13:58:32, 127.0.0., http://localhost/Php/cadastro.php, event: ***click****:, element: FORM, value/text:Escola*

13:59:57, 127.0.0., http://localhost/Php/cadastro.php, event: ***click****:, element: FORM, value/text:Curso*

14:00:12, 127.0.0., , http://localhost/Php/cadastro.php, event: ***click****:, element: INPUT, value/text:Enviar*

14:00:59, 127.0.0. http://localhost/Php/sucesso.html, event: ***load***

Quadro 4 – Arquivo de *log* filtrado
Fonte: Autoria própria.

Após a etapa de pré-processamento dos dados, os arquivos de *log* são transformados em um ou mais arquivos refinados, mantendo-se somente as informações necessárias para a próxima fase do RP.

7.1.5 Mensuração e Normalização dos Dados de Entrada

Nesta etapa do Reconhecimento de Padrões os dados são mensurados e normalizados para a montagem do vetor de entrada, necessário para as próximas etapas do RP. Foram

realizadas convenções para a normalização e mensuração dos dados (HAYKIN, 2001; SILVA et al., 2010). Após vários testes, verificou-se que para melhor desempenho de classificação dos padrões, a normalização dos eventos registrados nos arquivos de *log* deveria estar compreendida no intervalo entre 0 e 1. Assim, foram analisadas as 120 sequências de execução da tarefa "Cadastrar currículo", presentes em um arquivo de *log*.

A mensuração dos dados do arquivo de *log* foi realizada de acordo com algumas convenções. Em relação aos tipos de eventos que correspondem ao carregamento de página (*load*), rolagem de página (*scroll*) e mudança de foco (*focus*), são atribuídos o valor 1, no caso de ocorrência do evento, ou o valor 0, no caso de não ocorrência do evento, conforme ilustrado na Tabela 4.

Tabela 4 – Valores normalizados dos eventos (*load*, *scroll* e *focus*)

Evento	Ocorrência	Não ocorrência
Load	1	0
Scroll	1	0
Focus	1	0

Fonte: Autoria própria.

Para o tipo de evento "*click*", quando este ocorre em imagens ou *links* é atribuído o valor 1 para a ocorrência e 0 para não ocorrência, conforme Tabela 5.

Tabela 5 – Valores normalizados dos eventos (*click* – IMG)

Evento	Ocorrência	Não ocorrência
Click (IMG)	1	0

Fonte: Autoria própria.

Para os eventos de "*click*" realizados em botões, no caso específico de formulários, é atribuído o valor 1 para o botão "Enviar" e o valor 0,5 para o botão "Limpar", de acordo com a Tabela 6.

Tabela 6 – Valores normalizados dos eventos (*click* – INPUT)

Evento	Enviar	Limpar	Não ocorrência
Click (INPUT)	1	0,5	0

Fonte: Autoria própria.

Para os eventos de "*click*" em caixas de entrada (ocorridos dentro do formulário) é realizada a somatória de todos estes eventos e, para que a faixa de valores dos dados de entrada esteja entre o intervalo definido de 0 a 1, o número de eventos "*click*" é dividido por 100, conforme Tabela 7.

Tabela 7 – Valores normalizados dos eventos (*click* – FORM)

Evento *Click*	Valor Normalizado
Click (FORM)	Número de cliques / 100

Fonte: Autoria própria.

Além desses eventos, ainda são considerados o acesso (carregamento) às páginas de ajuda, erro e sucesso. Também é realizada a mensuração do tempo total de preenchimento do formulário, em segundos, calculado através da subtração do tempo final pelo tempo inicial e dividido por 1000, por questões de normalização.

Além dessas normalizações, também foi considerado o número total de ocorrência de cada evento para a realização da tarefa. Esse valor foi levado em consideração para que certa tolerância de erros do usuário fosse admitida no preenchimento do formulário. Por exemplo, o caso onde o usuário tem a intenção de clicar no botão "Limpar" ao invés do botão "Enviar"

não pode ser caracterizado como um problema de usabilidade, assim como a ocorrência de situações onde o usuário, por curiosidade, realiza ações indevidas.

Dessa forma, procurou-se desconsiderar questões relativas a ações intencionais do usuário, na tentativa de identificar apenas os problemas relacionados à interface do sistema web. Assim, é atribuída à maioria dos eventos certa margem aceitável de ocorrência. A Tabela 8 exemplifica a quantidade máxima aceitável de ocorrência para alguns eventos. Se em uma sequência analisada do arquivo de *log*, a quantidade de ocorrência dos eventos estiver dentro dessa faixa de valores, esta sequência é considerada sem problemas de usabilidade.

Tabela 8 – Quantidade máxima aceitável de ocorrência de eventos

Evento	Faixa de valores aceitável
Load – Index.php	1
Click – IMG	1
Load – Cadastro.php	1 – 2
Click – FORM	13 – 17
Load – Ajuda.html	0 – 1
Click – INPUT	1 – 2
Scroll	0 – 2
Focus	0
Load – Erro.php	0
Tempo	0,06 – 0,18
Load – Sucesso.html	1

Fonte: Autoria própria.

Na Tabela 9, é apresentada uma sequência normalizada para a tarefa “Cadastrar currículo”, obtida através da análise do arquivo de *log* filtrado apresentado no Quadro 4 e do modelo de tarefas apresentado na subseção 7.1.2. Essa sequência representa uma tarefa realizada com sucesso e que não apresenta problemas de usabilidade. Logo, uma sequência (padrão) com essas características encaixa-se na classe “Ausência de problemas de usabilidade”.

Tabela 9 – Padrão / Classe (Ausência de problemas de usabilidade)

Nº	Evento	Valores de Entrada	Nº de ocorrência do evento
1	Load – Index.php	1	1
2	Click – IMG	1	1
3	Load – Cadastro.php	1	1
4	Click – FORM	13/100 = 0,13	13
5	Load – Ajuda.html	0	0
6	Click – INPUT	1	1
7	Scroll	0	0
8	Focus	0	0
9	Load – Erro.php	0	0
10	Tempo	0,06	--
11	Load – Sucesso.html	1	1

Fonte: Autoria própria.

Na Tabela 10, é apresentada outra possível sequência normalizada para a tarefa "Cadastrar currículo", realizada através da análise do arquivo de *log* filtrado apresentado no Quadro 4, representando uma tarefa realizada com acesso à página de erro (erro.php).

Tabela 10 – Padrão / Classe (Entrada de dados)

(continua)

Nº	Evento	Valores de Entrada	Nº de ocorrência do evento
1	Load – Index.php	1	1
2	Click – IMG	1	1
3	Load – Cadastro.php	1	1
4	Click – FORM	20/100 = 0,2	20
5	Load – Ajuda.html	0	0
6	Click – INPUT	1	1
7	Scroll	0	0

Tabela 10 – Padrão / Classe (Entrada de dados)

(conclusão)

Nº	Evento	Valores de Entrada	Nº de ocorrência do evento
8	Focus	0	0
9	Load – Erro.php	1	1
10	Tempo	0,11	--
11	Load – Sucesso.html	1	1

Fonte: Autoria própria.

Percebe-se que o número de eventos "*click*" em caixas de entrada é maior, pois nesse caso o usuário tentou enviar o formulário com campos preenchidos inadequadamente, obtendo acesso à página de erro; consequentemente, o usuário teve a necessidade de digitar novamente alguns dados. Por esse mesmo motivo o tempo de execução da tarefa foi maior, porém o usuário alcançou o sucesso na execução da tarefa. Uma sequência com essa característica encaixa-se na classe específica de um possível problema de usabilidade, denominada "Entrada de Dados", e geralmente ocorre por ausência de recursos na implementação dos campos de entrada, como máscaras, mensagens explicativas etc.

Visando esclarecer a viabilidade das convenções adotadas para as normalizações dos eventos, pode ser citado como exemplo o evento "click – INPUT". Esse evento indica o clique de mouse em um elemento "botão". No caso do sistema web, nos botões "Enviar" ou "Limpar". Se o evento "click – INPUT" estiver presente no arquivo de *log* e possuir como valor o texto "Enviar", é atribuído o valor 1, indicando o sucesso no envio de dados. Analisando outra possibilidade, se este evento estiver presente no arquivo de *log* e possuir como valor o texto "Limpar", é atribuído o valor 0,5, indicando a intenção do usuário de excluir os dados digitados. Porém, se o evento "click – INPUT" não estiver presente no arquivo *log* é atribuído o valor 0.

Alguns usuários podem clicar "sem querer" no botão errado e precisar carregar novamente a página de cadastro. Essa situação também pode indicar que o botão de ação secundária "Limpar" não foi implementado da maneira recomendada, por exemplo, com menos peso visual que o botão de ação primária "Enviar". Assim, como o problema também pode estar relacionado com a utilização da palavra genérica "Enviar", pois recomenda-se que outras expressões sejam utilizadas.

Essa condição também pode ser identificada através da observação da frequência elevada de novas chamadas para a página de cadastro (evento “load”). Isso pode ser caracterizado como um problema de usabilidade, pois a implementação dos botões não segue recomendações para o bom projeto de formulários.

Logo, sequências que apresentem essas características (padrão) são classificadas em uma classe específica representando um possível problema de usabilidade, denominada “Funcionalidades”, conforme Tabela 11.

Tabela 11 – Padrão / Classe (Funcionalidades)

Nº	Evento	Valores de Entrada	Nº de ocorrência do evento
1	Load – Index.php	1	1
2	Click – IMG	1	1
3	Load – Cadastro.php	1	3
4	Click – FORM	42/100 = 0,42	42
5	Load – Ajuda.html	0	0
6	Click – INPUT	0,5	2
7	Scroll	0	0
8	Focus	0	0
9	Load – Erro.php	0	0
10	Tempo	0,16	--
11	Load – Sucesso.html	0	0

Fonte: Autoria própria.

Na Tabela 12, é apresentada uma sequência que exemplifica um problema de usabilidade relacionado à navegação. Esse tipo de problema é associado ao evento “focus” que indica a mudança do foco de entrada de dados ou quando um elemento recebe ou perde o foco. Esse evento pode ocorrer devido à ausência de navegação lógica entre os campos, ou seja, a ordem sequencial de digitação dos campos (de cima para baixo, da esquerda para direita) fornecida ao usuário não foi respeitada. Quando uma sequência desse tipo é identificada no arquivo de *log*, esta é associada à classe “Navegação”.

Tabela 12 – Padrão / Classe (Navegação)

Nº	Evento	Valores de Entrada	Nº de ocorrência do evento
1	Load – Index.php	1	1
2	Click – IMG	1	1
3	Load – Cadastro.php	1	1
4	Click – FORM	15/100 = 0,15	15
5	Load – Ajuda.html	0	0
6	Click – INPUT	1	1
7	Scroll	0	0
8	Focus	1	1
9	Load – Erro.php	0	0
10	Tempo	0,10	--
11	Load – Sucesso.html	1	1

Fonte: Autoria própria.

A Tabela 13 ilustra um exemplo de problema de usabilidade relacionado ao *layout* da interface. Nesse exemplo, percebe-se que o evento "scroll" ocorreu por diversas vezes, acima da faixa tolerável que foi definida. Isso pode indicar que o projeto desse formulário provavelmente não seguiu algumas recomendações, como a partição por páginas e/ou abas, ou foi implementado em mais de uma coluna, fatos estes que podem ocasionar a rolagem excessiva da página.

Tabela 13 – Padrão / Classe (*Layout*)

(continua)

Nº	Evento	Valores de Entrada	Nº de ocorrência do evento
1	Load – Index.php	1	1
2	Click – IMG	1	1
3	Load – Cadastro.php	1	1
4	Click – FORM	13/100 = 0,13	13

Tabela 13 – Padrão / Classe (*Layout*)

(conclusão)

Nº	Evento	Valores de Entrada	Nº de ocorrência do evento
5	Load – Ajuda.html	0	0
6	Click – INPUT	1	1
7	Scroll	1	3
8	Focus	0	0
9	Load – Erro.php	0	0
10	Tempo	0,12	--
11	Load – Sucesso.html	1	1

Fonte: Autoria própria.

Na Tabela 14, é ilustrada uma sequência que caracteriza um problema de usabilidade relacionado ao tempo de preenchimento do formulário. Nesse caso, o usuário utilizou um tempo de preenchimento considerado alto, o que pode indicar alguma dificuldade. Essa dificuldade pode estar relacionada com a falta de uso de rótulos e abreviações nos campos, como também pode indicar que o usuário teve dificuldades em acessar e/ou preencher os campos do formulário.

Tabela 14 – Padrão / Classe (Frequência de interação)

(continua)

Nº	Evento	Valores de Entrada	Nº de ocorrência do evento
1	Load – Index.php	1	1
2	Click – IMG	1	1
3	Load – Cadastro.php	1	1
4	Click – FORM	13/100 = 0,13	13
5	Load – Ajuda.html	1	1
6	Click – INPUT	1	1
7	Scroll	0	0
8	Focus	0	0
9	Load – Erro.php	0	0

Tabela 14 – Padrão / Classe (Frequência de interação)

(conclusão)

Nº	Evento	Valores de Entrada	Nº de ocorrência do evento
10	Tempo	0,4	--
11	Load – Sucesso.html	1	1

Fonte: Autoria própria.

Com os dados normalizados e mensurados, o vetor de entrada foi constituído inicialmente com 11 eventos, que são denominados formalmente de características na próxima subseção.

7.2 EXTRAÇÃO DE CARACTERÍSTICAS

Para a definição do conjunto de características mais importantes para este trabalho é utilizado o *Principal Component Analysis* (PCA). O objetivo da aplicação desse método de extração é reduzir a dimensionalidade dos dados de entrada. Assim, são extraídas apenas aquelas características que melhor discriminam os padrões, retirando do conjunto original de dados as informações redundantes. A próxima subseção apresenta os resultados obtidos e análise realizada a partir desses resultados.

7.2.1 Análise de Componentes Principais (PCA)

Os dados selecionados como entrada do extrator de características são os arquivos de *log* filtrados, de acordo com as técnicas e normalizações aplicadas na subseção anterior. Assim, o vetor inicial de dados ficou composto das seguintes características: 1) Load – Index.php; 2) Click – IMG; 3) Load – Cadastro.php; 4) Click – FORM; 5) Load – Ajuda.html; 6) Load – Erro.php; 7) Scroll; 8) Focus; 9) Click – INPUT; 10) Tempo; e 11) Load – Sucesso.html.

Para a análise dos componentes principais foi utilizado o ambiente *Waikato Environment for Knowledge Analysis* (WEKA) (HALL et al., 2009). O WEKA disponibiliza várias ferramentas de reconhecimento de padrões como classificadores, métodos de seleção de características, etc. O software WEKA trabalha com um formato de arquivo próprio, denominado ARFF (*Attribute Relation File Format*). Um arquivo ARFF é composto por uma estrutura definida em três partes: cabeçalho, declaração dos atributos e seção de dados. No cabeçalho tem-se a definição do nome da relação, a declaração dos atributos contém uma lista de todos os atributos com os nomes dos atributos e seus tipos. O arquivo ARFF desenvolvido está disponível no Apêndice A para complementar a descrição do seu formato.

A Figura 16 apresenta a tela inicial do *software* WEKA.

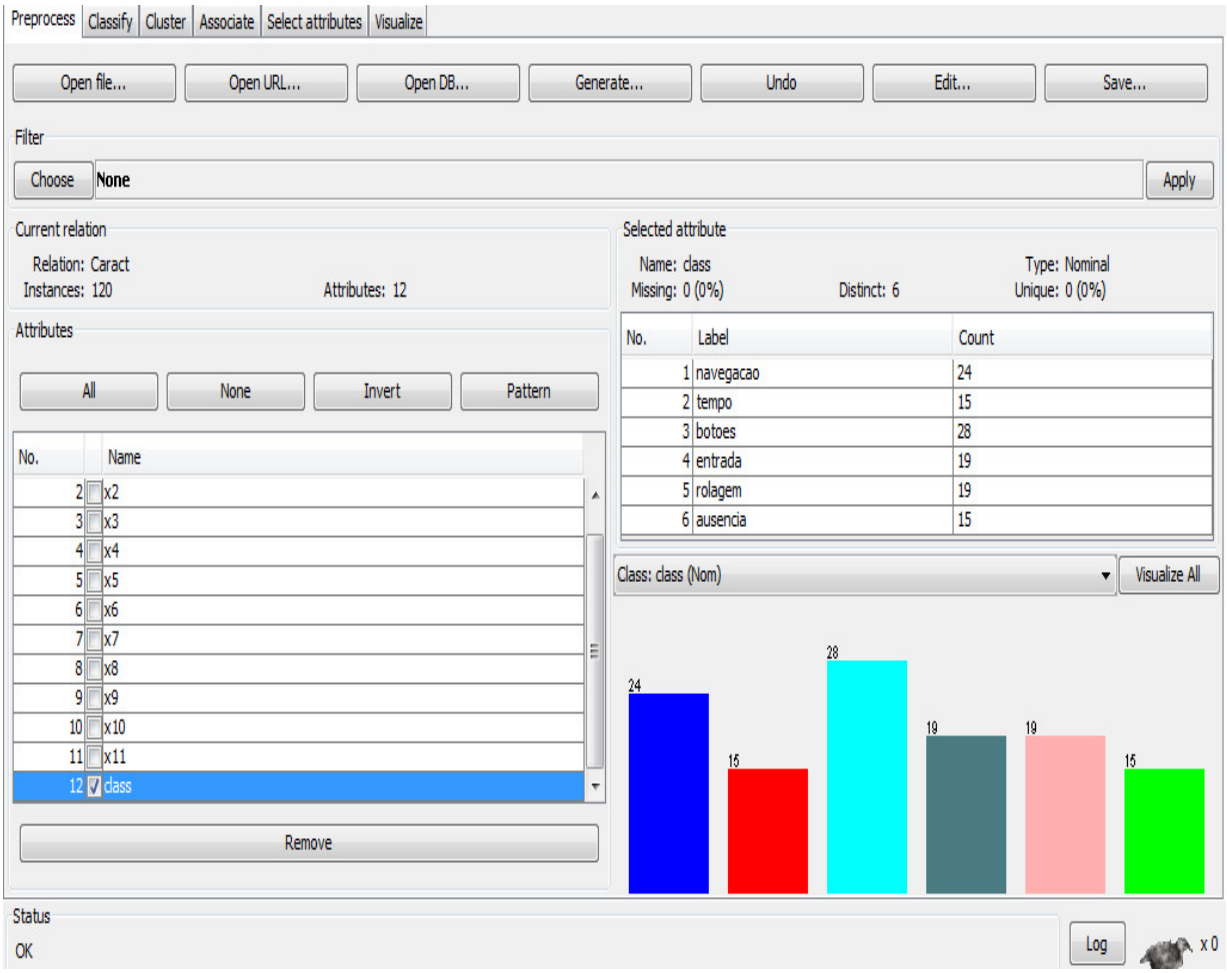

Figura 16 – Tela principal do *software* WEKA
Fonte: Autoria própria.

Na Figura 16, são ilustrados os elementos de pré-processamento do *software* WEKA e através do botão “Open File” é possível carregar um arquivo ARFF. Pode ser observado o total de 120 amostras e o total de características (*attributes*) utilizado na análise dos componentes principais. Nesse caso, são consideradas 12 características, pois o programa WEKA leva em consideração, além das 11 características originais, uma nova característica, denominada “*class*”, por padrão.

O gráfico de barras, visualizado no canto inferior direito da Figura 16, representa a relação atributos x classes, mostrando a separação dos padrões em 6 classes. As cores indicam as respectivas classes (da esquerda para a direita): azul (navegação), vermelho (tempo), azul celeste (botões), cinza (entrada), rosa (rolagem) e verde (ausência de problemas). Também pode ser observado o total de amostras pertencentes a cada classe.

Na Figura 17 é ilustrada a aba “*Select attributes*”, que consiste na área destinada para extração de características do conjunto original de dados. Nesse caso, foi selecionada a técnica de Análise de Componentes Principais conjuntamente com o algoritmo *Ranker*, que realiza a busca de componentes principais mais relevantes. É necessário ressaltar que esse algoritmo é definido por padrão pelo *software* WEKA para a aplicação do PCA.

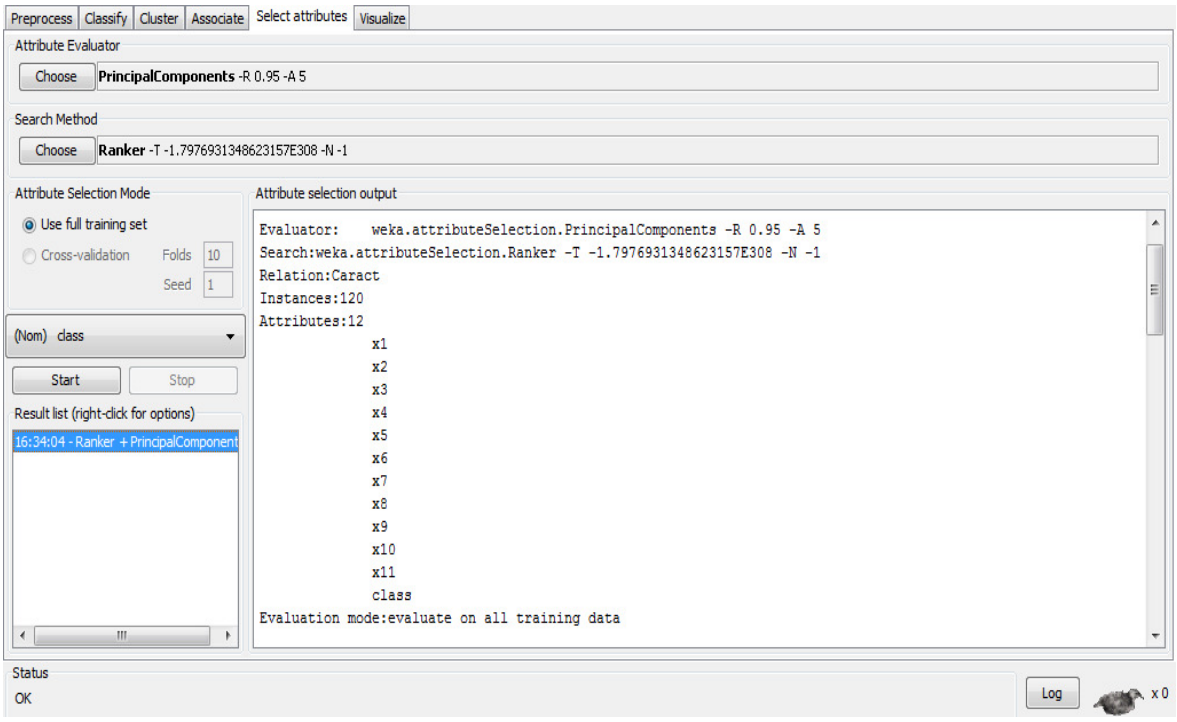

Figura 17 – Aba “*Select attributes*” do *software* WEKA
Fonte: Autoria própria.

Os resultados gerados pelo WEKA com a aplicação do PCA nos dados originais (120 amostras) deste trabalho estão no Apêndice B. A partir dos resultados gerados pelo WEKA, foi analisada primeiramente a matriz de correlação, apresentada na Figura 18.

$$R = \begin{pmatrix} 1 & 1 & 0.93 & 0.08 & 0.34 & 0.09 & 0.12 & -0.42 & 0.28 & 0.84 & 0.83 \\ 1 & 1 & 0.93 & 0.08 & 0.34 & 0.09 & 0.12 & -0.42 & 0.28 & 0.84 & 0.83 \\ 0.93 & 0.93 & 1 & 0.15 & 0.22 & 0.1 & 0.15 & -0.36 & 0.22 & 0.73 & 0.73 \\ 0.08 & 0.08 & 0.15 & 1 & -0.01 & 0.2 & -0.21 & 0.14 & 0.19 & 0.09 & 0.08 \\ 0.34 & 0.34 & 0.22 & -0.01 & 1 & 0.27 & 0.35 & 0.07 & 0.2 & 0.4 & 0.4 \\ 0.09 & 0.09 & 0.1 & 0.2 & 0.27 & 1 & -0.24 & -0.22 & 0.05 & 0.11 & 0.10 \\ 0.12 & 0.12 & 0.15 & -0.21 & 0.35 & -0.24 & 1 & 0.2 & -0.12 & 0.14 & 0.13 \\ -0.42 & -0.42 & -0.36 & 0.14 & 0.07 & -.022 & 0.2 & 1 & -0.15 & -0.32 & -0.40 \\ 0.28 & 0.28 & 0.22 & 0.19 & 0.2 & 0.05 & -0.12 & -0.15 & 1 & 0.22 & 0.27 \\ 0.84 & 0.84 & 0.73 & 0.09 & 0.4 & 0.11 & 0.14 & -0.32 & 0.22 & 1 & 0.26 \\ 0.11 & 0.11 & 0.15 & -0.20 & 0.34 & -0.24 & 0.10 & 0.20 & 0.27 & 0.26 & 1 \end{pmatrix}$$

Figura 18 – Matriz de correlação
Fonte: Autoria própria.

Percebe-se que à medida que aumenta a correlação entre as variáveis, o eixo maior torna-se cada vez maior, assim como o eixo menor torna-se cada vez menor, até se anular, quando as variáveis estão no caso limite de correlação perfeita igual a 1. Nessas condições, o eixo maior tem um comprimento igual a 11, assim como a soma dos autovalores, visto que o vetor original de características foi formado por 11 características.

Na Tabela 15, estão apresentados os autovalores e as porcentagens de variação explicada e acumulada em cada componente principal, mostrando que a porcentagem de variação explicada foi diminuindo a cada componente principal e que a variação acumulada em 7 componentes resultava em aproximadamente 96%. Percebe-se que o primeiro autovalor corresponde à maior porcentagem da variabilidade total presente e assim sucessivamente, sendo os componentes principais formados por combinações lineares das características.

Tabela 15 – Autovalores e porcentagens de variação explicada e acumulada em cada PC

Autovalor (λ_i)	Variância explicada (*Vi*) (%)	Variância acumulada (%)	Componentes principais (PCs)
4.06055	0.40605	0.40605	0.481x3+0.481x2+0.45x4+0.435x11+0.4901x1...
1.54743	0.15474	0.5608	0.68 x8-0.407x7-0.366x5+0.317x9+0.283x6...
1.21935	0.12193	0.68273	-0.548x6-0.488x5-0.472x9-0.37x7-0.245x10...
1.11718	0.11172	0.79445	-0.629x7+0.484x5-0.372x6+0.339x9+0.29x10...
0.86351	0.08635	0.8808	0.829x10-0.387x5+0.206x6-0.197x4-0.188x7...
0.51043	0.05104	0.93184	-0.675x8+0.505x9-0.352x5+0.23x11+0.208x6...
0.36618	0.03662	0.96846	0.462x9+0.446x7-0.438x6-0.331x5+0.0999x1...

Fonte: Autoria própria.

Outra forma utilizada para encontrar as características mais importantes na formação dos eixos foi a análise dos autovetores, apresentados na Tabela 16. Quanto maior o autovetor na combinação linear (ou componente principal), mais importante é a variável daquele autovetor na formação do eixo.

Tabela 16 – Coeficientes dos PCs e porcentagem de explicação acumulada

Características	Componentes principais						
	PC1	PC2	PC3	PC4	PC5	PC6	PC7
X1	0.4901	0.0512	0.0911	0.0900	-0.0701	0.1324	0.0999
X2	0.4811	0.0269	0.0911	0.0710	-0.0634	0.1024	0.0994
X3	0.4811	0.0269	0.0911	0.0710	-0.0634	0.1024	0.0994
X4	0.4503	0.0161	0.0950	0.1431	-0.1974	-0.0597	0.3419
X5	0.0585	-0.3659	-0.4881	0.4842	-0.3872	-0.3524	-0.3306
X6	0.2139	0.2831	-0.5483	-0.3723	0.2057	0.2084	-0.4378
X7	0.0920	-0.4065	-0.3704	-0.6290	-0.1877	-0.1414	0.4463
X8	0.0687	0.6804	-0.1242	0.0321	-0.0328	-0.6746	0.1425
X9	-0.2272	0.3175	-0.4720	0.3385	-0.1737	0.5053	0.4617
X10	0.1755	-0.2248	-0.2446	0.2918	0.8289	-0.1434	0.2256
X11	0.4353	0.0688	-0.0146	0.0100	-0.0847	0.2383	-0.2807

Fonte: Autoria própria.

Na Tabela 16, os autovetores correspondem aos componentes principais e representam o resultado do carregamento das variáveis originais em cada um deles. Esses carregamentos foram considerados como uma medida da relativa importância de cada variável em relação aos componentes principais e os respectivos sinais, se positivos ou negativos, indicam respectivamente relações diretamente e inversamente proporcionais.

Nas colunas da Tabela 16 estão os autovetores relativos aos componentes principais, cujos coeficientes representam os pesos de cada característica na obtenção das funções lineares. Percebe-se que a soma dos quadrados dos pesos das variáveis para cada componente (soma em coluna), corresponde ao valor próprio ou autovalor do componente. A soma dos quadrados dos pesos dos componentes para cada variável (soma em linha) corresponde à proporção da variância de cada variável, explicada pelas componentes retiradas e será igual a 1 se forem considerados todos os componentes.

A interpretação de cada componente foi feita com base nos pesos das variáveis e ficou simplificada, pois cada variável teve um peso relativamente mais elevado para apenas um dos componentes e pesos pequenos ou próximos de zero para todos os componentes restantes.

Com base na análise das tabelas e resultados obtidos verificou-se que para obter um bom valor de porcentagem acumulada nos padrões, apenas 5 componentes principais seriam suficientes, uma vez que estes representam 88,08% (linha 5 da Tabela 15) do conjunto original de dados, resultando em pouca perda de informação e discriminando adequadamente os padrões em suas respectivas classes. Assim, o vetor de dados para a fase de classificação foi gerado a partir desses 5 componentes principais.

A análise de componentes principais é uma técnica de transformação de variáveis ou características. Se cada variável for considerada como um eixo de variabilidade, estando usualmente correlacionada com outras variáveis. A análise de componentes principais transforma os dados de tal modo a descrever a mesma variabilidade total existente, mas sem correlação. Graficamente, as PCA pode ser descrita como a rotação de pontos existentes num espaço multidimensional originando eixos, ou componentes principais, que dispostos num espaço reduzido de dimensões, representam variabilidade suficiente que possa indicar algum padrão a ser interpretado.

7.3 CLASSIFICAÇÃO

Para a classificação de padrões proposta foi implementada uma rede neural artificial do tipo MLP (Subseção 6.3.1.2). Os padrões foram agrupados em 6 classes distintas que representam diferentes tipos de problemas de usabilidade. As configurações da RNA com melhor desempenho, assim como os resultados obtidos com os testes são descritos nas próximas subseções.

7.3.1 Projeto da Rede

Após alguns testes preliminares com outros tipos de RNAs definiu-se pela utilização de uma rede *MultiLayer Perceptron* (MLP), do tipo *feedforward* com algoritmo *Backpropagation* com gradiente descendente. Esse tipo de rede é muito indicado para problemas envolvendo reconhecimento de padrões (SILVA et al., 2010). Foi desenvolvido um programa utilizando a linguagem C++ para a implementação dessa rede e o código fonte está disponível no Apêndice C.

O ajuste dos pesos e do limiar de cada neurônio da rede foi efetuado utilizando o processo de treinamento supervisionado, considerando que as respectivas saídas desejadas estavam disponíveis para cada amostra dos sinais de entrada (SILVA et al., 2010).

Os parâmetros da rede foram alterados ao longo do treinamento com a finalidade de encontrar um modelo com resultado de generalização da classificação dos padrões satisfatório ao problema. Os vetores dos pesos iniciais dos neurônios foram inicializados com valores aleatórios pequenos, entre 0 e 1. Com o objetivo de buscar a melhor configuração para a taxa de aprendizado esta foi alterada ao longo do treinamento; a taxa de aprendizado utilizada que apresentou melhor desempenho foi de 0,1. A Tabela 17 resume os parâmetros selecionados para o projeto da RNA.

Tabela 17 – Parâmetros de projeto da RNA

Parâmetros	Valores / Tipos
Treinamento	Supervisionado
Vetor de pesos iniciais	Entre 0 e 1
Taxa de aprendizado	0,1
Máximo de épocas	2000
Erro quadrático médio	10^{-6}
Função de ativação	Logística sigmoidal
Número total de amostras	120
Número de neurônios de saída	3
Número de classes	6

Fonte: Autoria própria.

Uma rede MLP com apenas um neurônio na camada de saída é capaz de distinguir somente duas classes, ao passo que três neurônios podem diferenciar até oito classes no total (SILVA et al., 2010). Logo, a rede MLP adotada foi composta por três neurônios na camada de saída e as combinações dos valores de saída de cada neurônio representaram as 6 classes envolvidas no problema de classificação, conforme Tabela 18. Entenda-se por y_n os neurônios da camada de saída, onde n representa o número máximo de neurônios.

Tabela 18 – Codificação de saída dos neurônios

Nº da Classe	Nome da Classe	y_1	y_2	y_3
1	Navegação	1	0	0
2	Freqüência de Iteração	0	1	0
3	Entrada de dados	0	0	1
4	Layout	1	1	0
5	Funcionalidade	1	0	1
6	Ausência de Problemas	0	0	0

Fonte: Autoria própria.

7.3.2 Resultados dos Treinamentos

Após a normalização e extração de características, os dados foram divididos em dois conjuntos: treinamento e validação. Na fase de treinamento e validação, o processo somente foi interrompido por dois mecanismos pré-determinados: o primeiro diz respeito, na fase de retropropagação, pelo erro quadrático médio; o segundo foi referente ao número máximo de interações permitidas.

No tocante à arquitetura da rede, foi utilizada apenas uma camada oculta, considerando que esta topologia mais simples foi suficiente para possibilitar a classificação dos padrões de saída. Uma das técnicas estatísticas mais utilizadas para seleção das melhores topologias candidatas é a validação cruzada (*cross-validation*) (SILVA et al., 2010).

Neste trabalho é aplicado o método denominado validação cruzada por amostragem aleatória, no qual o conjunto total de dados, constituído de 120 amostras, foi aleatoriamente dividido em dois conjuntos: treinamento (80% do total dos dados) e validação (20% do total dos dados). O conjunto de treinamento foi utilizado para treinar todas as topologias candidatas e o conjunto de teste somente foi aplicado para selecionar a topologia que apresentou os melhores resultados de generalização.

A rede foi treinada com o conjunto inicial de características e posteriormente com o conjunto reduzido, calculado na etapa de Extração de Características (Seção 7.2). Assim, com o objetivo de encontrar a melhor arquitetura, partiu-se de 5 neurônios na camada oculta, aumentando este número sem limites iniciais pré-determinados, conforme Tabela 19.

Tabela 19 – Tipos de arquiteturas utilizadas

Rede	Nº Entradas	Nº Neurônios Camada Escondida	Nº Saídas
Rede 1	11	5	6
Rede 2	11	10	6
Rede 3	11	15	6
Rede 4	5	5	6
Rede 5	5	10	6
Rede 6	5	15	6

Fonte: Autoria própria.

Na Tabela 20 são apresentados os resultados obtidos referentes aos treinamentos utilizando as arquiteturas apresentadas na Tabela 19. Para cada rede foram registrados apenas os resultados referentes ao melhor treinamento.

A rede com 10 entradas e 15 neurônios na camada oculta (Rede 6) apresentou melhor desempenho, com menor erro quadrático médio, menor tempo de convergência e maior taxa de acertos. Isso ressalta a importância da etapa de extração de características, uma vez que a rede com o menor número de características, mas com relevância para a discriminação dos padrões, apresentou melhor resultado.

Tabela 20 – Resultados obtidos com a RNA

Rede	Tempo (s)	Erro quadrático médio (%)	Taxa de acerto (%)
Rede 1	16.33	0,0792	94%
Rede 2	15.30	0,0512	95%
Rede 3	13.63	0,0331	100%
Rede 4	14.32	0,0454	93%
Rede 5	13.60	0,0300	98%
Rede 6	12.13	0,0208	100%

Fonte: Autoria própria.

Após os treinamentos, a rede foi testada com os dados de teste, que representam 20% do conjunto total de amostras. Além disso, foi simulada a operação da rede em um ambiente real de operação. Um conjunto de 180 amostras não apresentadas na fase de treinamento e testes foi classificado pela rede. O desempenho da RNA foi satisfatório, considerando que a taxa de acerto da rede permaneceu na faixa de 90%. Esse resultado indica a viabilidade do emprego de técnicas de reconhecimento de padrões no processo de classificação automática de problemas de usabilidade. Por ser uma abordagem geral, esta pode ser utilizada para outros tipos de tarefas, desde que definidos e analisados previamente.

8 CONCLUSÕES

Neste trabalho foi apresentada a aplicação de técnicas de Reconhecimento de Padrões na identificação e classificação automática de problemas de usabilidade na interface de um sistema web. O foco inicial do trabalho foi a identificação de possíveis problemas de usabilidade em formulários web. Os potenciais problemas de usabilidade de formulários web foram definidos a partir das recomendações descritas por Wroblewski (2008) e Shneiderman e Plaisant (2010). As tarefas realizadas pelo usuário foram obtidas da interação do usuário armazenada em arquivos de *log*. A classificação das tarefas que são realizadas conforme o esperado e das que são consideradas potenciais problemas de usabilidade é realizada através de uma Rede Neural Artificial.

O tratamento dos dados dos arquivos de *log* e a mensuração e normalização são atividades que exigem tempo e esforço razoáveis. Apesar de existirem métricas disponíveis, o tratamento dos dados das entradas é um processo difícil e deve ser realizado com muita atenção, principalmente por representar a primeira etapa do Reconhecimento de Padrões (RP), a partir da qual as etapas subsequentes são dependentes. Logo, se os dados não forem tratados corretamente nesta etapa, o processo de RP pode ficar comprometido. Cabe ressaltar que é necessária a realização de uma filtragem nos arquivos de *log* para que somente as características mais significativas para o problema sejam consideradas nas etapas de extração e classificação dos padrões.

Um modelo de tarefas foi desenvolvido para representar padrões de execução de tarefas esperados. O modelo de tarefas proposto neste trabalho difere-se dos demais apresentados na literatura, pois ao invés de considerar somente as páginas (URLs) acessadas pelo usuário, representa as sequências necessárias para a realização de uma tarefa com base em eventos realizados pelo usuário em elementos de interação que estão contidos numa página web. O uso de eventos permitiu a modelagem mais precisa da interação do usuário para a execução de uma tarefa, aumentando a contribuição deste modelo de tarefas na avaliação e identificação de problemas de usabilidade do sistema web.

Adicionalmente às vantagens oferecidas pelo modelo de tarefas desenvolvido, os problemas de usabilidade são identificados e classificados automaticamente em padrões distintos, através de técnicas de reconhecimento de padrões. Os padrões são baseados em diretrizes de usabilidade para o projeto de interfaces. As demais abordagens descritas na literatura, apresentadas no Capítulo 2, oferecem medidas relacionadas à eficiência e eficácia

de um sistema web, porém não classificam de forma automática os potenciais problemas de usabilidade em interfaces de sistemas web.

A abordagem proposta pode ser aplicada em outros tipos de sistemas web, inclusive em sistemas de automação web, que é a motivação inicial para este trabalho, porém a definição do modelo de tarefas deve ser ajustada de acordo com a tarefa desejada. Além disso, o projeto da RNA deve ser analisado e seus parâmetros devem ser modificados com o objetivo de encontrar uma configuração que apresente bom desempenho para o contexto analisado.

O processo de extração de características mostrou-se eficaz, pois reduziu consideravelmente o número de entradas, transformando as características originais em novos componentes principais, mas preservando a correta discriminação entre os padrões. Esta etapa também contribuiu no processo de classificação, pois a rede neural implementada apresentou melhor desempenho com menos características.

A rede *MultiLayer Perceptron* mostrou-se adequada na classificação de problemas de usabilidade. Taxas de acerto acima de 90% foram obtidas para todas as configurações testadas. Para os cenários avaliados foi possível verificar a capacidade de generalização dessa rede, a qual permitiu a identificação de padrões de erros de usabilidade desconhecidos. A rede conseguiu classificar corretamente os dados de entrada nas saídas esperadas, que foram divididas em seis classes.

A abordagem apresentada neste trabalho é eficaz para o fim proposto, possibilitando a identificação e classificação automática de potenciais problemas de usabilidade através da análise de *log* e técnicas de reconhecimento de padrões. Além de ser uma técnica de baixo custo, uma vez que não necessita de muitos recursos envolvidos no processo, pode auxiliar no desenvolvimento e manutenção de sistemas web, possibilitando a adequação destes aos critérios de usabilidade. Esses critérios podem proporcionar uma melhor interação do usuário com o sistema, reduzindo o número de fracassos e o tempo de execução das tarefas.

Como mencionado anteriormente, os sistemas de automação atuais utilizam tecnologias web que proporcionam ao usuário o controle e monitoramento remoto dos processos de uma indústria. No ambiente industrial, a qualidade está diretamente relacionada com a operação segura e eficiente dos sistemas. No entanto, o volume de informações apresentado ao usuário através da interface desses sistemas impõe restrições de tempo e segurança, impondo uma carga cognitiva grande por parte do usuário. Assim, muitos incidentes na indústria podem ocorrer devido ao erro humano que é provocado, na maioria das vezes, por uma interface mal projetada. Dessa forma, é importante que a interface desses

sistemas, cuja segurança na operação é crítica, seja projetada de forma a oferecer ao usuário maior flexibilidade, ergonomia e eficiência.

8.1 CONTRIBUIÇÕES

As contribuições deste trabalho podem ser destacadas de acordo com os seguintes itens:

- Definição de um processo para a identificação e classificação automática de problemas de usabilidade na interface de sistemas web.
- Desenvolvimento de um modelo de tarefas baseado nos eventos rastreados da interação do usuário na realização de uma tarefa;
- Levantamento dos problemas de usabilidade relacionados a formulários web, assim como o agrupamento de recomendações similares em classes específicas;
- Emprego de uma RNA para a classificação automática de problemas de usabilidade.

8.2 PUBLICAÇÕES RELATIVAS AO TRABALHO

Durante o desenvolvimento dos trabalhos relacionados à dissertação, dois artigos foram publicados:

- SANTANA, G. A.; PANSANATO, L. T. E. Identifying usability problems in web applications through analysis of user interaction logs using pattern recognition. In: IADIS INTERNATIONAL CONFERENCE WWW/INTERNET, 15., 2011, Rio de Janeiro. **Proceedings...** Rio de Janeiro: ICWI , 2011. 587-590.

- SANTANA, G. A.; GOEDTEL, A.; PANSANATO, L. T. E. Identificação Automática de problemas de usabilidade através de redes neurais artificiais. In: Congresso Brasileiro de Automática, 19., 2012, Campina Grande. **Anais...** Campina Grande: CBA, 2012.

8.3 TRABALHOS FUTUROS

Alguns dos trabalhos futuros que podem dar continuidade às contribuições apresentadas são resumidos a seguir:

- As definições de sequências de eventos realizadas pelo usuário durante a interação com um sistema web são armazenadas em arquivos de *log* e filtradas manualmente. Esse processo representa uma atividade cansativa, difícil e propensa a erros em sua condução. Logo, em trabalhos futuros devem ser estudadas as possibilidades da realização da filtragem automática dos arquivos de *log*. Essa filtragem pode ser realizada automaticamente utilizando abordagens algorítmicas.
- Os valores referentes à normalização e mensuração dos arquivos de *log* foram definidos a partir dos melhores resultados obtidos do extrator de características PCA e da RNA. Em trabalhos futuros sugere-se empregar outros tipos de normalização e mensuração, utilizando outros intervalos de valores, visto que a montagem do vetor de entrada depende da técnica empregada para a classificação dos padrões. Assim, se outras técnicas de RP forem utilizadas, recomenda-se que o intervalo de valores seja modificado com o objetivo de verificar e comparar os resultados futuros com aqueles apresentados neste trabalho.
- Para a extração de características deste trabalho foi utilizado o PCA. Sugere-se aplicar outras técnicas de reconhecimento de padrões para extração de características, como o método Wrapper (MALDONADO; WEBER, 2009) que seleciona as melhores características do conjunto original de dados. Esse método é totalmente dependente do algoritmo de aprendizado, ou seja, as características selecionadas pelo Wrapper são diferentes dependendo do classificador utilizado. No entanto, o classificador empregado conjuntamente com o Wrapper tende a apresentar melhor desempenho na classificação dos padrões.
- Uma RNA do tipo MLP foi aplicada ao problema de classificação de padrões neste trabalho. Como trabalho futuro, outras técnicas para a classificação dos padrões podem ser utilizadas, como Lógica Fuzzy, classificador Bayesiano e outros tipos de RNAs com aprendizado não supervisionado.

- Os eventos resultantes da interação do usuário com o sistema web analisados foram capturados por meio da ferramenta WAUTT. Sugere-se explorar a disponibilidade de novos tipos de eventos que são capturados pela nova versão da ferramenta WAUTT, ainda em desenvolvimento. A disponibilidade de novos tipos de eventos traz novas possibilidades para a descrição de tarefas e, consequentemente, para a identificação automática de outros problemas de usabilidade.
- O modelo de tarefas deste trabalho foi baseado na análise da interação do usuário com uma página web com um formulário simples, contendo apenas elementos básicos de interação. Sugere-se a análise de páginas web mais complexas que possuem outros tipos de elementos interativos como menus, barras de navegação e multimídia. A partir dessa análise, um novo modelo de tarefas pode ser definido com base na interação do usuário nesses novos elementos. Além disso, outras classes de problemas de usabilidade relacionadas a formulários também podem ser definidas.
- A abordagem proposta foi testada em um sistema web simples desenvolvido apenas para fins de geração de dados e validação das técnicas utilizadas. Porém, recomenda-se o emprego desta abordagem em domínios específicos, como sistemas de automação web, sistemas web de notícias, comércio eletrônico e sistemas web para aprendizado eletrônico (*e-learning*), definindo novas tarefas pertinentes a contexto.

8.4 CONSIDERAÇÕES FINAIS

Atualmente, a Internet é o meio mais utilizado para acesso às informações. Vários tipos de sistemas industriais têm migrado para a plataforma web, pois esta pode proporcionar acesso ou controle remoto aos processos presentes em uma planta industrial. Porém, o projeto da interface desses sistemas é de grande importância para que a interação do usuário ocorra com sucesso. Em geral, a interface desses sistemas apresenta grande volume de informações e pode representar um forte impacto na ocorrência de erro humano, podendo levar o usuário/operador a cometer decisões equivocadas. Assim, a qualidade da interface pode reduzir a ocorrência de erro humano e aumentar a produtividade.

A pesquisa e os trabalhos realizados no contexto desta dissertação serviram de apoio para dar prosseguimento a novas pesquisas, assim como disponibilizaram uma base para o conhecimento do estado da arte relacionado com a identificação de problemas de usabilidade na interface de um sistema web. Dessa forma, os resultados e contribuições apresentadas representam progressos para o estado da arte relacionado com o tema deste trabalho.

REFERÊNCIAS

ABDI, H.; WILLIAMS, L. J. Principal Component Analysis. **WIREs Comp Stat**, [S.l.], v. 2, n. 4, p. 433-459, jul./aug. 2010.

AQUINO, N. et al. Conceptual modelling of interaction. In: INTERNATIONAL WORKSHOP ON USER INTERFACE EXTENSIBLE MARKUP LANGUAGE, 2., 2011, Lisbon. **Proceedings…** Lisbon: UsiXml, 2011. p. 335-358.

BALBO, S. et al. Choosing the right task modelling notation: a Taxonomy. In: DIAPER, D. (Ed.) **The handbook of task analysis for human-computer interaction**, London: Lawrence Erlbaum Associates, 2004. p. 445-166.

BALBO, S. et al. Leading web usability evaluation to WAUTER. In: AUSTRALIAN WORLD WIDE WEB CONFERENCE, 11., 2005, Sidney. **Proceedings…**Sidney: AusWeb, 2005.

BASUKI, T. A. et al. Model-checking user behaviour using interacting components. **Formal aspects of computing**, [S.l.], v. 21, n. 6, p. 571-588, dec. 2009.

BENYON, D. Task analysis and system design: the discipline of data. **Interacting with Computers**, London, v. 4, n. 2, p. 246-259, aug. 1992.

BISHOP, C. M. **Pattern recognition and machine learning**. 2nd ed. London: Springer, 2007.

BODART, F. et al. A model-based approach to presentation: a continuum from task analysis to prototype. **Interactive Systems: Design, Specification and Verification**, Dublin, v. 13, p. 77-94, nov. 1994.

BREEDVELT, I.; PATERNÒ, F.; SEVERIINS, C. A. Reusable structures in task models. **Interactive Systems: Design, Specification and Verification**, Granada, v. 21, n. 10, p. 225-239, mar. 1997.

CAFFIAU, S. et al. Increasing the expressive power of task analysis: Systematic comparison and empirical assessment of tool-supported task models. **Interacting with Computers**, London, v. 22, n. 6, p. 569-593, nov. 2010.

CARBONELL, J. G. **Paradigms for machine learning**. 2nd ed. New York: Elsevier North-Holland, 1990.

CARD, S.; MORAN, T.; NEWELL, A. **The psychology of human computer interaction**. 2nd ed. New Jersey: Lawrence Erlbaum Associates, 1983.

CUMMINGS, M. L. et al. **Human-system interface complexity and opacity part I:** literature review. Massachusetts: Institute of Technology Editors, 2010.

CYBIS, W. A.; BETIOL, A. H.; FAUST, R. **Ergonomia e usabilidade:** conhecimentos, métodos e aplicações. 2. ed. São Paulo: Novatec, 2010.

DIAPER, D. Defining and representing activity context for systems analysis. In: WORKSHOP ON ACTIVITY CONTEXT REPRESENTATION TECHNIQUES AND LANGUAGES, 11., 2011, San Francisco. **Proceedings…** San Francisco: AAAI'11, 2001. p. 8-13.

DIAPER, D.; STANTON, N. **The handbook of task analysis for human-computer interaction**. New Jersey: Lawrence Erlbaum Associates, 2004.

DIAS, C. **Usabilidade na web:** criando portais mais acessíveis. Rio de Janeiro: Alta Books, 2007.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern classification**. 2nd ed. Los Angeles: Wiley- Interscience, 2001.

FARENC, C.; LIBERATI, V.; BARTHET, M-F. Automatic ergonomic evaluation: what are the limits. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN OF USER INTERFACES, 3., 1999, Bucharest. **Proceedings**… Bucharest: CADUI, 1999. p. 159-170.

FERNANDEZ, R et al. Evaluation of estimates of effort to develop applications with supervisory system. In: SYMPOSIUM SERIES IN MECHATRONICS, 2012, São Paulo. **Proceedings…** São Paulo: ABCM, 2012. p. 115-128.

FIELDS, R. E.; WRIGHT, P. C.; HARRISON, M. D. A task centered approach to analysing human error tolerance requirements. In: IEEE INTERNATIONAL SYMPOSIUM ON REQUIREMENTS ENGINEERING, 2., 1995, York. **Proceedings…** York: RE'95, 1995. p. 18-26.

FREEMAN, J. A.; SKAPURA, D. M. **Neural networks**: algorithms, applications and programming techniques. New York: Addison-Wesley Publishing, 1991.

GREENBERG, S. Working through task-centered system design. In: DIAPER, D. (Ed.); STANTON, N. (Ed.). **The handbook of task analysis for human–computer Iinteraction**. London: Lawrence Erlbaum Assoc., 2004. p. 49-66.

GUERRERO, C. et al. A process for human centered modelling of incident scenarios. In: SYMPOSIUM OF THE WORKGROUP HUMAN-COMPUTER INTERACTION AND USABILITY, 4., 2008, Graz. **Proceedings…** Graz: USAB, 2008. p. 439-458.

_____. Modelling incident scenarios**.** In: HUMAN ERROR, SAFETY AND SYSTEMS DEVELOPMENT, 7., 2004, Toulouse. **Proceedings…** Toulouse: IFIP, 2004. p. 77-92.

HALL, M. et al. The Weka data mining software: an update**. ACM SIGKDD Explorations Newslwtter**, New York, v. 11, n.1, p. 10-18, jun. 2009.

HARTSON, H. R. et al. The User Action Notation: A user-oriented representation for direct manipulation interface designs. **ACM Transactions on Information Systems**, New York, v. 8, n. 3, p. 181-203, aug. 1990.

HAYKIN, S. **Neural networks**: a comprehensive foundation. 2nd ed. New York: Prentice-Hall, 2001.

HEBB, D. O. **The organization of behavior**. New York: John-Wiley & Sons Inc, 1949.

HUSSAK, W.; YAUG, S. H. Formal development of remote interfaces for large- scale real-time systems. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 9., 2004, Singapore. **Proceedings…** Singapore: IEEE, 2004. p. 124-129.

IDOUGHI, D.; KERKAR, M.; KOLSKI, C. Towards new web service based supervisory systems in complex industrial organizations: basic principles and case study. **Computers in Industry**, [S.l.], v. 61, n. 3, p. 235-249, oct. 2010.

INTERNATIONAL STANDART ORGANIZATION / INTERNATIONAL ELECTROTECHNICAL COMMISSION. **ISO/IEC 9126**: Information technology - Software product evaluation - Quality characteristics and guidelines for their use. Geneva, 1991.

_____. **ISO/IEC 9126-1**. Software engineering – Product quality - Part 1. Geneva, 2000.

_____. **ISO/IEC 9241-11**. Guidance on usability - Ergonomic requirements for office work with visual display terminals **–** Part 11. Geneva, 1998.

IVORY, M.; HEARST, M. The state of the art in automating usability evaluation of user interfaces. **ACM Computing Survey**, Berlin, v. 33, n. 4, p. 470-516, dec. 2001.

JAIN, A. et al. Statistical pattern recognition: A review. **IEEE Transactions Pattern Analysis and Machine Intelligence**. [S.l.], v. 22, n. 1, p. 4-37, jan. 2000.

JANSEN, B. J.; SPINK, A.; TAKSAI, I. **Handbook of research on web log analysis**. Information Science Reference, IGI Global, 2009.

JOLLIFFE, I. **Principal component analysis**. 2nd ed. New York: John Wiley & Sons, 2005.

JOHNSON, C. et al. Can prospective usability evaluation predict data errors?. In: AMERICAN MEDICAL INFORMATICS ASSOCIATION ANNUAL SYMPOSIUM, 2010, Washington. **Proceedings…** Washington: AMIA, 2010. p. 346.

KARAT, J. Cost-benefit and business case analysis of usability engineering. In: ACM SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 1993, Amsterdam. **Proceedings…** Amsterdam: CHI'93, 1993. p. 145-148.

KADLEC, J. Code Characterization for automated user interface creation. **Information Sciences and Technologies Bulletin**, Bratislava, v. 3, n. 1, p. 1-8, mar. 2011.

KOLB, J. et al. Using concurrent task trees for stakeholder-centered modeling and visualization of business processes. In: INTERNATIONAL CONFERENCE S-BPM ONE - EDUCATION AND INDUSTRIAL DEVELOPMENTS, 4., 2012, Vienna. **Proceedings…** Vienna: S-BPM ONE, 2012. p. 237-251.

LACEROF, A.; PATERNÒ, F. Automatic support for usability evaluation. **IEEE Transactions on Software Engineering**, [S.l.], v. 24, n. 10, p. 863-888, oct. 1998.

LAHTI, J. P.; KANKAANPAA, A. S. T. Web-based technologies in power plant automation and SCADA systems: A review and evaluation, In: IEEE INTERNATIONAL CONFERENCE ON CONTROL SYSTEM, COMPUTING AND ENGINEERING, 2011, Malaysia. **Proceedings…** Malaysia: ICCSCE, 2011. p. 279-284.

LECUN, Y. **Connectionism in perspective**. Amsterdam: North-Holland, 1989.

LIMBOURG, Q.; VANDERDONCKT, J. **The handbook of task analysis for human-computer interaction**. New York: PressI Llc, 2004.

MALDONADO, S.; WEBER, R. A wrapper method for feature selection using Support Vector Machines. **Information Sciences**, [S.l.], v. 179, n. 13, p. 2208-2217, jun. 2009.

MÁSSON, E.; WANG, Y.J. Introduction to computation and learning in artificial neural networks. **European Journal of Operational Research**, Paris, v. 47, n.1, p. 1-28, jul. 1990.

MINGOTI, S. A. **Análise de dados através de métodos de estatística multivariada:** uma abordagem aplicada. Belo Horizonte: UFMG, 2007.

MORANDINI, M. et al. A task model proposal for web sites usability evaluation for the Ergo-Monitor environment. **Human-Computer Interaction International – Interaction Design and Usability**, New York, v. 4550, n. 2, p. 1188-1197, jul. 2007.

NASCIMENTO NETO, J. A. et al. Proposing strategies to prevent the human error in qutomated industrial environments. In: FOUNDATIONS OF AUGMENTED COGNITION, NEUROERGONOMICS AND OPERATIONAL NEUROSCIENCE, 5., 2009, San Francisco. **Proceedings…** San Francisco: FAC'09, 2009. p. 279-288.

NAVARRE, D. et al. ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. **ACM Transactions on Computer-Human Interaction**, New York, v. 16, n. 4, p. 18, mar. 2009.

NIELSEN, J. **Designing web usability**. San Francisco: Morgan Kaufmann, 2000.

_____. **Usability engineering**. San Francisco: Morgan Kaufmann, 1993.

NIELSEN, J.; LORANGER, H. **Prioritizing web usability**. New York: Elsevier, New Riders Publishing, 2006.

NIELSEN, J; MACK, R.L. **Usability inspection methods**. New York: John Wiley & Sons, 1994.

OLIVEIRA NETTO, A. **IHC**: modelagem e gerência de interfaces com o usuário. Florianópolis: VisualBooks, 2004.

OYEWOLE, S. A.; HAIGHT, J. M. Determination of optimal paths to task goals using expert system based on GOMS model. **Computers in Human Behavior**, [S.l.], v. 27, n. 2, p. 823-833, nov. 2011.

PAGANELLI, L.; PATERNÒ, F. Intelligent analysis of user interactions with web applications. In: INTERNATIONAL CONFERENCE ON INTELLIGENT USER INTERFACES, 7., 2002, San Francisco. **Proceedings…** San Francisco: IUI, 2002. p. 111-118.

PALANQUE, P. et al. Model-based approach for supporting engineering usability evaluation of interaction techniques. In: ACM SIGCHI SYMPOSIUM ON ENGINEERING INTERACTIVE COMPUTING SYSTEMS, 3., 2011, Pisa. **Proceedings**… Pisa: EICS, 2011. p. 21-30.

_____. Task model – system model: towards an unifying formalism. In: INTERNATIONAL CONFERENCE ON HUMAN-COMPUTER INTERACTION, 6., 1995. **Proceedings…**Tokyo: HCI, 1995. p. 489-494.

PAQUETTE, D.; SCHNEIDER, K. A. Interaction templates for constructing user interfaces from task models. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN OF USER INTERFACES, 4., 2004, Funchal. **Proceedings…** Funchal: CADUI, 2004. p. 223–235.

PATERNÒ, F. ConcurTaskTrees: An engineering approach to model-based design of interactive systems. In: CHANG, S. K. (Ed.). **The handbook of software engineering and knowledge engineering**. Los Angeles: World Scientific Publishing Co., 2001. p. 483-500..

_____. Task models in interactive software systems. In: CHANG, S. K. (Ed.). **The handbook of software engineering and knowledge engineering**. Los Angeles: World Scientific Publishing Co., 2001, p. 817-835.

PATERNÒ, F.; PAGANELLI, L. Remote automatic evaluation of web sites based on task models and browser monitoring. In: EXTENDED ABSTRACTS ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2001, New York. **Proceedings…** New York: CHI, 2001. p. 283-284.

PISO, E. Task analysis for process control tasks. **Journal of Occupational Psychology**, [S.l.], v. 84, n. 4, p. 247-254, nov. 2011.

PREECE, I. et al. **Human-computer interaction**. London: Addison-Wesley, 1994.

PREECE, J.; ROGERS, Y.; SHARP, H. **Interaction design**: beyond human-computer interaction. 3th ed. Chichester: Wiley, 2012.

PRESSMAN, R. S. **Software engineering**: a practitioner's approach. 5th ed. New York: Mc Graw Hill, 2010.

RIVOLLI, A.; PANSANATO, L. T. E.; MARINHO, D. A. WAUTT: Uma Ferramenta para o Rastreamento da Interação do Usuário com Aplicações Interativas Web. In: BRAZILIAN SYMPOSIUM ON MULTIMEDIA AND THE WEB, 14., 2008. **Proceedings…** Florianópolis: WEBMEDIA, 2008. p. 179-181.

ROBLES, J. R.; KIM, T-H. Architecture of wireless supervisory control and data acquisition system. **Advances in Computational Intelligence, Man-Machine Systems and Cybernetics**, Venezuela, v. 2, n. 3, p. 241-244, dec. 2010.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Phisicological Review**, [S.l], v.11, , 5, p. 386-408, nov. 1958.

SANTANA, V. F.; BARANAUSKAS, M. C. C. Summarizing observational client-side data to reveal web usage patterns. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 2010, New York. **Proceedings…** New York: SAC'10, 2010. p. 1219-1223.

SCAPIN, D.; BASTIEN, C. Analyse des tâches et aide ergonomique à laconception: l'approche MAD*. In: PROCEEDINGS OF THE 14TH FRENCH-SPEAKING CONFERENCE ON HUMAN-COMPUTER, 14., 2001, New York. **Proceedings…** New York: IHM'01, 2001. p. 85-116.

SCAPIN, D.; PIERRET-GOLBREICH, C. Towards a method for task description: MAD. In: INTERNATIONAL SCIENTIFIC CONFERENCE ON WORK WITH DISPLAY UNITS, 2., 1989, Amsterdam. **Proceedings…**Amsterdam: WWDU, 1989. p. 371-380.

SHNEIDERMAN, B.; PLAISANT, C. **Designing the user interface**: strategies for effective human-computer interaction. 5th ed. New York: Addison-Wesley, 2010.

SILVA, I. N.; SPATTI, D. H.; FLAUZINO, R. A. **Redes neurais artificiais para engenharia e ciências aplicadas**. São Paulo: Artliber, 2010.

SILVA, L. E. et al. A utilização de ferramentas web para supervisão e controle de equipamentos do sistema elétrico de potência. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 17., 2008, Juiz de Fora. **Anais...** Juiz de Fora: CBA, 2008.

STORRS, G. The notion of task in human-computer interaction. In: CONFERENCE OF THE BRITISH HUMAN-COMPUTER INTERACTION, 10., 1995, London. **Proceedings…** London: BSCHCI, 1995. p. 357-357.

STOUFFER, K.; FALCO, J.; KENT, K. Guide to supervisory control and data acquisition (SCADA) and industrial control systems security. **Recommendations of the National Institute of Standards and Technology**, [S.l], v. 2, n. 4, p. 800-820, oct. 2006.

SURESH, K. D. et al. Research of Internet Based Supervisory Control and Information system. In: IEEE INTERNATIONAL CONFERENCE ON RECENT TRENDS IN INFORMATION TECHNOLOGY, 2011, Chennai. **Proceedings…** Chennai: ICRTIT, 2011. p. 1180-1185.

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern recognition**. 4th ed. New York: Academic Press, 2008.

TIEDTKE, T.; MARTIN, C.; GERTH, N. AWUSA a tool for automated website usability analysis. In: INTERNATIONAL WORKSHOP ON THE DESIGN, SPECIFICATION AND VERIFICATION OF INTERACTIVE SYSTEMS, 9., 2002, Rostock**. Proceedings…** Rostock: DSV-IS, 2002. p. 251-266.

TURNELL, M. F. Q. V.; FARIAS, G. F. The use of Supervisory Software in the Industrial Automation Process Control from the User Interface Perspective. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 1996, China. **Proceedings…** China: IEEE, 1996. p. 83-91.

TURNELL, M. F. et al. Análise de Incidentes Industriais Baseada em Modelos. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 15., 2004, Gramado. **Anais...** Gramado: CBA, 2004.

VAN DER VEER, G. et al. GTA: groupware task analysis: modelling complexity. **Acta Psychologica**, [S.l.], v. 91, n. 3, p. 297-322, apr. 1996.

VARGAS, A. et al. A method for remote and semi-automatic usability evaluation of web-based applications through users behavior analysis. In: INTERNATIONAL CONFERENCE ON METHODS AND TECHNIQUES IN BEHAVIORAL RESEARCH, 7., 2010, Netherlands**. Proceedings…** Netherlands: Measuring Behavior '10, 2010. p. 19:1-19:5.

_____. Analyzing user interaction logs to evaluate the usability of Web applications. In: IEEE SYMPOSIUM WEB SOCIETY**,** 3., 2011, London. **Proceedings…** London: IEEE-SWS, 2011. p. 61-67.

VIEIRA, M. F. Q. et al. Improving system safety through agent-supported user/system interfaces: effects of operator behavior model. In: AGENT-DIRECTED SIMULATION SYMPOSIUM, 2., 2005, San Diego. **Proceedings…** San Diego: ADS'05, 2005. p. 86-93.

WASSERMAN, P. D. **Neural computing**: theory and practice. New York: Van Nostrand Co., 1989.

WELIE, M. et al. An ontology for task world models. In: INTERNATIONAL EUROGRAPHICS WORKSHOP ON DESIGN SPECIFICATION AND VERIFICATION OF INTERACTIVE SYSTEMS, 5., 1998, London. **Proceedings…** London: EWDSVIS'98, 1998. p. 3-5.

WINCKLER, M. A. A.; PIMENTA, M. S. Análise e Modelagem de Tarefas - Tutorial IHC. In: SIMPÓSIO DE INTERAÇÃO HUMANO-COMPUTADOR, 4., 2004. **Anais…** Rio de Janeiro: IHC, 2004.

WROBLEWSKI, L. **Web form design**: filling in the blanks. New York: Rosenfeld Media, 2008.

YANG, S. H. et al. Development of an internet-based process control system. In: SYMPOSIUM OF THE WORKING PARTY ON COMPUTER AIDED PROCESS ENGINEERING, 35., 2002, Lappeenranta. **Proceedings…** Lappeenranta: CADUI, 2002. p. 601-606.

YANG, S. H.; YANG, L. Guidance on design of internet-based process control systems. **Automatica Sínica**, [S.l.], v. 31, n.1, p. 123-139, mar. 2005.

YASMEEN, A.; GUNTER, E. L. Automated framework for formal operator task analysis. In: INTERNATIONAL SYMPOSIUM ON SOFTWARE TESTING ANDANALYSIS, 2001, Toronto. **Proceedings…** Toronto: ISSTA, 2011. p. 78-88.

APÊNDICE A – Arquivo ARFF (Caract.arff)

```
@relation 'Caract'
@attribute 'x1' real
@attribute 'x2' real
@attribute 'x3' real
@attribute 'x4' real
@attribute 'x5' real
@attribute 'x6' real
@attribute 'x7' real
@attribute 'x8' real
@attribute 'x9' real
@attribute 'x10' real
@attribute 'x11' real
@attribute 'class' {navegacao,tempo,botoes,entrada,rolagem,ausencia}
@data

1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,0.0000,0.5000,0.0600,1.0000,botoes
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.1000,1.0000,navegacao
1.0000,1.0000,1.0000,0.1600,0.0000,0.0000,0.0000,0.0000,0.5000,0.0800,0.0000,botoes
1.0000,1.0000,1.0000,0.2100,0.0000,1.0000,0.0000,0.0000,1.0000,0.1200,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.1800,1.0000,navegacao
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,0.5000,0.1100,1.0000,botoes
1.0000,1.0000,1.0000,0.2400,0.0000,0.0000,1.0000,0.0000,1.0000,0.1000,1.0000,rolagem
1.0000,1.0000,1.0000,0.2200,0.0000,1.0000,0.0000,0.0000,1.0000,0.1800,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.0600,1.0000,navegacao
1.0000,1.0000,1.0000,0.1500,0.0000,0.0000,0.0000,0.0000,0.5000,0.1000,1.0000,botoes
1.0000,0.0000,0.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.1100,0.0000,navegacao
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.0600,1.0000,ausencia
1.0000,1.0000,1.0000,0.1600,0.0000,0.0000,1.0000,0.0000,1.0000,0.0800,1.0000,rolagem
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,0.5000,0.1700,1.0000,botoes
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,1.0000,1.0000,0.1300,1.0000,navegacao
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.2000,1.0000,tempo
1.0000,1.0000,1.0000,0.2600,1.0000,1.0000,0.0000,0.0000,1.0000,0.0700,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.0600,1.0000,ausencia
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,1.0000,0.0000,1.0000,0.0800,1.0000,rolagem
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.3200,1.0000,tempo
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,0.5000,0.0700,1.0000,botoes
1.0000,1.0000,1.0000,0.2900,1.0000,1.0000,0.0000,0.0000,1.0000,0.0800,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.0700,1.0000,ausencia
1.0000,1.0000,1.0000,0.1600,0.0000,0.0000,0.0000,0.0000,1.0000,0.4000,1.0000,tempo
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,1.0000,0.0000,1.0000,0.0700,1.0000,rolagem
1.0000,1.0000,1.0000,0.1900,1.0000,0.0000,0.0000,0.0000,0.5000,0.0800,1.0000,botoes
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.0700,1.0000,navegacao
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,0.0000,0.5000,0.0900,1.0000,botoes
1.0000,1.0000,1.0000,0.4000,0.0000,1.0000,0.0000,0.0000,1.0000,0.0900,1.0000,entrada
1.0000,1.0000,1.0000,0.1800,0.0000,0.0000,0.0000,1.0000,1.0000,0.0700,1.0000,navegacao
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,0.5000,0.1000,1.0000,botoes
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,1.0000,0.0000,1.0000,0.1000,1.0000,rolagem
```

1.0000,1.0000,1.0000,0.3200,0.0000,1.0000,0.0000,0.0000,1.0000,0.1100,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,1.0000,1.0000,0.1100,1.0000,navegacao
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,0.5000,0.1000,1.0000,botoes
1.0000,0.0000,0.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.0900,0.0000,navegacao
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.0600,1.0000,ausencia
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,1.0000,0.0000,1.0000,0.0600,1.0000,rolagem
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,0.5000,0.1000,1.0000,botoes
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.1000,1.0000,navegacao
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,0.0000,1.0000,0.5000,1.0000,tempo
1.0000,1.0000,1.0000,0.3100,1.0000,1.0000,0.0000,0.0000,1.0000,0.1000,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.1000,1.0000,ausencia
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,1.0000,0.0000,1.0000,0.0900,1.0000,rolagem
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,0.0000,1.0000,0.3200,1.0000,tempo
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,0.5000,0.1000,1.0000,botoes
1.0000,1.0000,1.0000,0.2200,0.0000,1.0000,0.0000,0.0000,1.0000,0.1100,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.0700,1.0000,ausencia
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,0.0000,1.0000,0.22000,1.000,tempo
1.0000,1.0000,1.0000,0.1600,0.0000,0.0000,1.0000,0.0000,1.0000,0.0700,1.0000,rolagem
1.0000,1.0000,1.0000,0.1500,1.0000,0.0000,0.0000,0.0000,0.5000,0.0900,1.0000,botoes
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.0700,1.0000,navegacao
1.0000,1.0000,1.0000,0.2000,1.0000,0.0000,0.0000,0.0000,0.5000,0.0800,0.0000,botoes
1.0000,1.0000,1.0000,0.2300,0.0000,1.0000,0.0000,0.0000,1.0000,0.0900,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,1.0000,1.0000,0.0800,1.0000,navegacao
1.0000,1.0000,1.0000,0.1200,0.0000,0.0000,0.0000,0.0000,0.5000,0.0800,1.0000,botoes
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,1.0000,0.0000,1.0000,0.0700,1.0000,rolagem
1.0000,1.0000,1.0000,0.2500,0.0000,1.0000,0.0000,0.0000,1.0000,0.0800,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.0600,1.0000,navegacao
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,0.0000,0.5000,0.0700,1.0000,botoes
1.0000,0.0000,0.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.0600,0.0000,navegacao
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.0600,1.0000,ausencia
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,1.0000,0.0000,1.0000,0.0700,1.0000,rolagem
1.0000,1.0000,1.0000,0.1600,0.0000,0.0000,0.0000,0.0000,0.5000,0.0900,1.0000,botoes
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,1.0000,1.0000,0.1100,1.0000,navegacao
1.0000,1.0000,1.0000,0.1700,1.0000,0.0000,0.0000,0.0000,1.0000,0.2900,1.0000,tempo
1.0000,1.0000,1.0000,0.2500,1.0000,1.0000,0.0000,0.0000,1.0000,0.1000,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.1000,1.0000,ausencia
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,1.0000,0.0000,1.0000,0.0900,1.0000,rolagem
1.0000,1.0000,1.0000,0.1900,1.0000,0.0000,0.0000,0.0000,1.0000,0.3900,1.0000,tempo
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,0.5000,0.0900,1.0000,botoes
1.0000,1.0000,1.0000,0.2800,0.0000,1.0000,0.0000,0.0000,1.0000,0.1000,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.0600,1.0000,ausencia
1.0000,1.0000,1.0000,0.1700,1.0000,0.0000,0.0000,0.0000,1.0000,0.3300,1.0000,tempo
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,1.0000,0.0000,1.0000,0.0900,1.0000,rolagem
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,0.0000,0.5000,0.1200,1.0000,botoes
1.0000,1.0000,1.0000,0.1800,0.0000,0.0000,0.0000,1.0000,1.0000,0.1300,1.0000,navegacao
1.0000,1.0000,1.0000,0.1600,0.0000,0.0000,0.0000,0.0000,0.5000,0.1500,1.0000,botoes
1.0000,1.0000,1.0000,0.2900,0.0000,1.0000,0.0000,0.0000,1.0000,0.1200,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.0900,1.0000,navegacao
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,0.5000,0.1200,1.0000,botoes
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,1.0000,0.0000,1.0000,0.1100,1.0000,rolagem

1.0000,1.0000,1.0000,0.3300,0.0000,1.0000,0.0000,0.0000,1.0000,0.1600,1.0000,entrada
1.0000,1.0000,1.0000,0.1700,1.0000,0.0000,0.0000,1.0000,1.0000,0.1000,1.0000,navegacao
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,0.0000,0.5000,0.1200,1.0000,botoes
1.0000,0.0000,0.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.0900,0.0000,navegacao
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.0600,1.0000,ausencia
1.0000,1.0000,1.0000,0.1900,0.0000,0.0000,1.0000,0.0000,1.0000,0.0900,1.0000,rolagem
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,0.0000,0.5000,0.1300,1.0000,botoes
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.1100,1.0000,navegacao
1.0000,1.0000,1.0000,0.1600,0.0000,0.0000,0.0000,0.0000,1.0000,0.2800,1.0000,tempo
1.0000,1.0000,1.0000,0.3000,1.0000,1.0000,0.0000,0.0000,1.0000,0.1100,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.0900,1.0000,ausencia
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,1.0000,0.0000,1.0000,0.0800,1.0000,rolagem
1.0000,1.0000,1.0000,0.1500,0.0000,0.0000,0.0000,0.0000,1.0000,0.2000,1.0000,tempo
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,0.5000,0.1200,1.0000,botoes
1.0000,1.0000,1.0000,0.3200,0.0000,1.0000,0.0000,0.0000,1.0000,0.1100,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.1000,1.0000,ausencia
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,0.0000,1.0000,0.3500,1.0000,tempo
1.0000,1.0000,1.0000,0.2000,0.0000,0.0000,1.0000,0.0000,1.0000,0.1200,1.0000,rolagem
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.0900,1.0000,navegacao
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,0.0000,0.5000,0.1000,1.0000,botoes
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,1.0000,0.0000,1.0000,0.1200,1.0000,rolagem
1.0000,1.0000,1.0000,0.3100,0.0000,1.0000,0.0000,0.0000,1.0000,0.1300,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.0900,1.0000,navegacao
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,0.0000,0.0000,0.5000,0.1000,1.0000,botoes
1.0000,0.0000,0.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.1000,0.0000,navegacao
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.0900,1.0000,ausencia
1.0000,1.0000,1.0000,0.1900,0.0000,0.0000,1.0000,0.0000,1.0000,0.0700,1.0000,rolagem
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,0.5000,0.0900,1.0000,botoes
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,1.0000,1.0000,0.1000,1.0000,navegacao
1.0000,1.0000,1.0000,0.1800,0.0000,0.0000,0.0000,0.0000,1.0000,0.2100,1.0000,tempo
1.0000,1.0000,1.0000,0.3300,1.0000,1.0000,0.0000,0.0000,1.0000,0.1300,1.0000,entrada
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.0600,1.0000,ausencia
1.0000,1.0000,1.0000,0.1300,1.0000,0.0000,1.0000,0.0000,1.0000,0.0700,1.0000,rolagem
1.0000,1.0000,1.0000,0.1900,1.0000,0.0000,0.0000,0.0000,1.0000,0.3700,1.0000,tempo
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,0.5000,0.1100,1.0000,botoes
1.0000,1.0000,1.0000,0.3800,0.0000,1.0000,0.0000,0.0000,1.0000,0.1100,1.0000,entrada
1.0000,1.0000,1.0000,0.1900,1.0000,0.0000,0.0000,0.0000,1.0000,0.2900,1.0000,tempo
1.0000,1.0000,1.0000,0.1300,0.0000,0.0000,0.0000,0.0000,1.0000,0.0600,1.0000,ausencia

APÊNDICE B – Resultados do PCA (Software WEKA)

= Run information ===

Evaluator: weka.attributeSelection.PrincipalComponents -R 0.95 -A 5
Search:weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation:Caract
Instances:120
Attributes:12

x1
x2
x3
x4
x5
x6
x7
x8
x9
x10
x11
class

Evaluation mode:evaluate on all training data

=== Attribute Selection on all input data ===

Search Method:

Attribute ranking.

Attribute Evaluator (unsupervised):

Principal Components Attribute Transformer

Correlation matrix

1 1 0.93 0.08 0.34 0.09 0.12 -0.42 0.28 0.84 0.83
1 1 0.93 0.08 0.34 0.09 0.12 -0.42 0.28 0.84 0.83

0.93	0.93	1	0.15	0.22	0.1	0.15	-0.36	0.22	0.73	0.73	
0.08	0.08	0.15	1	-0.01	0.2	-0.21	0.14	0.19	0.09	0.08	
0.34	0.34	0.22	-0.01	1	0.27	0.35	0.07	0.2	0.4	0.4	
0.09	0.09	0.1	0.2	0.27	1	-0.24	-0.22	0.05	0.11	0.10	
0.12	0.12	0.15	-0.21	0.35	-0.24	1	0.2	-0.12	0.14	0.13	
-0.42	-0.42	-0.36	0.14	0.07	-0.22	0.2	1	-0.15	-0.32	-0.40	
0.28	0.28	0.22	0.19	0.2	0.05	-0.12	-0.15	1	0.22	0.27	
0.84	0.84	0.73	0.09	0.4	0.11	0.14	-0.32	0.22	1	0.26	
0.11	0.11	0.15	-0.20	0.34	-0.24	0.10	0.20	-0.12	0.13	1	

eigenvalue	proportion	cumulative	
4.11965	0.41197	0.41197	0.481x3+0.481x2+0.45 x4+0.435x11+0.4901x1...
1.56383	0.15638	0.56835	0.68 x8-0.407x7-0.366x5+0.317x9+0.283x6...
1.26464	0.12646	0.69481	-0.548x6-0.488x5-0.472x9-0.37x7-0.245x10...
1.04363	0.10436	0.79918	-0.629x7+0.484x5-0.372x6+0.339x9+0.292x10...
0.87029	0.08703	0.8862	0.829x10-0.387x5+0.206x6-0.197x4-0.188x7...
0.48313	0.04831	0.93452	-0.675x8+0.505x9-0.352x5+0.238x11+0.208x6...
0.35915	0.03592	0.97043	0.462x9+0.446x7+0.342x4-0.331x5+0.0999x1...

Eigenvectors

V1	V2	V3	V4	V5	V6	V7	
0.4901	0.0512	0.0911	0.090	-0.0701	0.1324	0.0999	x1
0.4811	0.0269	0.0911	0.071	-0.0634	0.1024	0.0994	x2
0.4811	0.0269	0.0911	0.071	-0.0634	0.1024	0.0994	x3
0.4503	0.0161	0.095	0.1431	-0.1974	-0.0597	0.3419	x4
0.0585	-0.3659	-0.4881	0.4842	-0.3872	-0.3524	-0.3306	x5
0.2139	0.2831	-0.5483	-0.3723	0.2057	0.2084	-0.4378	x6
0.092	-0.4065	-0.3704	-0.629	-0.1877	-0.1414	0.4463	x7
0.0687	0.6804	-0.1242	-0.0321	-0.0328	-0.6746	0.1425	x8
-0.2272	0.3175	-0.472	0.3385	-0.1737	0.5053	0.4617	x9
0.1755	-0.2248	-0.2446	0.2918	0.8289	-0.1434	0.2256	x10
0.4353	0.0688	-0.0146	0.01	-0.0847	0.2383	-0.2807	x11

Ranked attributes:

0.588 1 0.481x3+0.481x2+0.45 x4+0.435x11-0.227x9...

0.4317 2 0.68 x8-0.407x7-0.366x5+0.317x9+0.283x6...

0.3052 3 -0.548x6-0.488x5-0.472x9-0.37x7-0.245x10...

0.2008 4 -0.629x7+0.484x5-0.372x6+0.339x9+0.292x10...

0.1138 5 0.829x10-0.387x5+0.206x6-0.197x4-0.188x7...

0.0655 6 -0.675x8+0.505x9-0.352x5+0.238x11+0.208x6...

0.0296 7 0.462x9+0.446x7-0.438x6+0.342x4-0.331x5...

Selected attributes: 1,2,3,4,5,6,7 : 7

APÊNDICE C – Código do MLP (MLP.cpp)

```
/* Este programa implementa um Perceptron Multicamadas */

// Autora: Gisele Alves Santana
#include <cstring>
#include <stdio.h>
#include <stdlib.h>
#include<iostream.h>
#include<conio.h>
#include <time.h>
#include<math.h>

void treinar(float X1T[50],float X2T[50],float X3T[50]);
void vizualizar(float X1[50],float X2[50],float X3[50]);

int main() {
   int op;
   float x1[50]={1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,
00000, 1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000, 1.0000,1.0000, 1.0000,1.0000,
1.0000, 1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000, 1.0000,1.0000,
1.0000, 00000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000, 1.0000,1.0000, 1.0000, 1.0000,
1.0000,  1.0000,1.0000,1.0000};
    float x2[50]={1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,
0.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000, 1.0000,1.0000,1.0000, 1.0000, 1.0000,
1.0000,1.0000,1.0000,1.0000, 1.0000, 1.0000,1.0000,1.0000, 1.0000, 1.0000, 1.0000, 1.0000,
1.0000,0.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000, 1.0000,1.0000, 1.0000, 1.0000,
1.0000,1.0000,1.0000,0.0000};
    float x3[50]={0.9900,0.7400,0.8900,0.8000,0.8000,0.7400,0.7400,0.8000, 0.8800,0.7400,
0.0000,0.7400,0.8000,0.8400,0.7400,0.7400,0.7400, 0.7400,0.7400,0.7400, 0.7400, 0.8000,
0.7400,0.7400,0.9000,0.9900,0.7400,0.8900,0.8000, 0.8000, 0.7400,0.7400, 0.8000,0.8800,
0.7400,0.0000,0.7400,0.8000,0.8400,0.7400,0.7400,0.7400, 0.7400,0.7400, 0.7400, 0.7400,
0.8000,0.7400,0.7400,1.0000};
    float x4[50]={1.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000, 0.0000,0.0000,
0.0000,0.0000,0.0000, 0.0000,0.0000,0.0000,0.0000,1.0000, 0.0000,1.0000,0.0000,
0.0000,0.0000,0.0000,0.0000,0.0000,1.0000,0.0000,0.0000, 0.0000,0.0000, 0.0000,
0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000, 0.0000, 0.0000, 1.0000,
0.0000,1.0000,0.0000,0.0000,0.0000,0.0000,0.0000,1.0000};
     float x5[50]={0.0000,1.0000,0.0000,1.0000,1.0000,0.0000, 1.0000,1.0000, 1.0000,0.0000,
0.0000,1.0000,1.0000,0.0000,1.0000,1.0000, 1.0000,1.0000,1.0000,1.0000, 0.0000,1.0000,
1.0000,1.0000,1.0000,0.0000,1.0000,0.0000, 1.0000,1.0000,0.0000, 1.0000,1.0000, 1.0000,
0.0000,0.0000,1.0000,1.0000,0.0000,1.0000, 1.0000,1.0000,1.0000, 1.0000, 1.0000, 0.0000,
1.0000,1.0000,1.0000,1.0000};
     float x6[50]={0.5000,0.5000,0.5000,0.5000,0.3000,0.6000,0.5000, 0.5000,0.2000,
0.5400,0.1000,0.5000,0.4000,0.5000,0.6000,0.7500,0.5000,0.3000, 0.6000,0.8000,0.5000,
0.6000,0.2000,0.9000,0.3000,0.5200,0.5300,0.4000,0.3000,0.2000,0.4500, 0.1200, 0.3300,
0.5100,0.1100,0.1300,0.2200,0.3600,0.1900,0.1800,0.7500,0.4600,0.1800, 0.5670, 0.8000,
0.1190,0.2900,0.1700,0.9000,0.3900};
```

```
        float x7[50]={0.0000,1.0000,0.0000,0.0000,1.0000,0.0000,0.0000,1.0000,
1.0000,0.0000,0.0000,0.0000,0.0000,0.0000,1.0000,0.0000,0.0000, 0.0000,0.0000,
0.0000,0.0000,1.0000,0.0000,0.0000,0.0000,0.0000,  1.0000,0.0000,0.0000,1.0000,
0.0000,0.0000,1.0000,1.0000,0.0000,0.0000,0.0000,0.0000,0.0000,1.0000, 0.0000,
0.0000,0.0000,0.0000,0.0000,0.0000,1.0000,0.0000,0.0000,0.0000};
        float x8[50]={0.0000,0.0000,0.0000,1.0000,0.0000,0.0000,0.0000, 1.0000,
0.0000,0.0000,1.0000,0.0000,0.0000,0.0000,0.0000,0.0000,1.0000, 0.0000,0.0000,
0.0000,0.0000,1.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,1.0000,0.0000,
0.0000,0.0000,1.0000,0.0000,0.0000,1.0000,0.0000,0.0000,0.0000,0.0000, 0.0000,
1.0000,0.0000,0.0000,0.0000,0.0000,1.0000,0.0000,0.0000,0.0000};
        float x9[50]={0.5000,0.5000,0.5000,0.5000,0.3000,0.6000,0.5000, 0.5000,
0.2000,0.5400,0.1000,0.5000,0.4000,0.5000,0.6000,0.7500, 0.5000,0.3000, 0.6000,0.8000,
0.5000,0.6000,0.2000,0.9000,0.3000,0.5200,0.5300,0.4000,0.3000, 0.2000,0.4500, 0.1200,
0.3300,0.5100,0.1100,0.1300,0.2200,0.3600,0.1900,0.1800,0.7500,0.4600, 0.1800, 0.5670,
0.8000,0.1190,0.2900,0.1700,0.9000,0.3900};
       float x10[50]={1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000, 1.0000,
1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000, 1.0000,1.0000, 1.0000,
1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000, 1.0000,1.0000, 1.0000, 1.0000,
1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000, 1.0000,1.0000, 1.0000,1.0000,
1.0000,1.0000,1.0000,1.0000,1.0000};

do {
   cout<<'\n'<<'\n';
   cout<<"--------------MENU PRINCIPAL---------------"<<'\n'<<'\n';
   cout<<"********PERCEPTRON MULTICAMADAS***********"<<'\n'<<'\n'<<'\n';
   cout<<"||          1. Treinar                ||"<<'\n';
   cout<<"||          2. Vizualizar Entradas    ||"<<'\n';
   cout<<"||          3. Sair                   ||"<<'\n';
   cout<<"\nDigite sua opcao --> ";
   cin>>op;
   switch (op) {
    case 1: treinar(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10);
          break;
    case 2: visualizar(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10);
          break; }
   }while (op!=3);

   system("PAUSE");
   return 0;
} // Fim de main()

void vizualizar(float X1[50],float X2[50],float X3[50],float X4[50],float X5[50],float
X6[50],float X7[50],float X8[50],float X9[50],float X9[50], float X10[50])
{
  cout<<"\n-----------Valores da Entrada 1 -------------------"<<'\n';
  for(int i = 0; i < 50; i++)
  cout<<X1[i]<<'\t';
  cout<<'\n'<<'\n';
  cout<<"-----------Valores da Entrada 2 -------------------"<<'\n';
  for (int j=0; j<50; j++)
```

```
    cout<<X2[j]<<'\t';
    cout<<'\n'<<'\n';
    cout<<"-----------Valores da Entrada 3 --------------------"<<'\n';
    for (int k=0; k<50; k++)
    cout<<X3[k]<<'\t';
    cout<<"\n-----------Valores da Entrada 4 --------------------"<<'\n';
   for(int i = 0; i < 50; i++)
   cout<<X4[i]<<'\t';
   cout<<'\n'<<'\n';
   cout<<"\n-----------Valores da Entrada 5 --------------------"<<'\n';
   for(int i = 0; i < 50; i++)
   cout<<X5[i]<<'\t';
   cout<<'\n'<<'\n';
   cout<<"\n-----------Valores da Entrada 6--------------------"<<'\n';
   for(int i = 0; i < 50; i++)
   cout<<X6[i]<<'\t';
   cout<<'\n'<<'\n';
   cout<<"\n-----------Valores da Entrada 7 --------------------"<<'\n';
   for(int i = 0; i < 50; i++)
   cout<<X7[i]<<'\t';
   cout<<'\n'<<'\n';
   cout<<"\n-----------Valores da Entrada 8 --------------------"<<'\n';
   for(int i = 0; i < 50; i++)
   cout<<X8[i]<<'\t';
   cout<<'\n'<<'\n';
   cout<<"\n-----------Valores da Entrada 9 --------------------"<<'\n';
   for(int i = 0; i < 50; i++)
   cout<<X9[i]<<'\t';
   cout<<'\n'<<'\n';
   cout<<"\n-----------Valores da Entrada 10 --------------------"<<'\n';
   for(int i = 0; i < 50; i++)
   cout<<X10[i]<<'\t';
   cout<<'\n'<<'\n';
}

void treinar(float X1T[50],float X2T[50],float X3T[50],float X4T[50],float X5T[50],float
X6T[50],float X7T[50],float X8T[50],float X9T[50],float X10T[50]) {
    float saida_d1[50]={1,1,1,0,1,1,1,0,1,1,1,0,1,1,1,0,0,0,1,0,1,1,0,0,1,1,1,1,0,1,1,1,0,1,1,1,0,
                                  1,1,1,0,0,0,1,0,1,0,0,0,1};
    float saida_d2[50]={0,0,0,0,1,0,1,0,0,0,0,0,1,0,0,1,0,0,1,1,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,
                                 1,0, 0,1,0,0,1,1,0,0,0,1,1};
          float saida_d3[50]={1,0,1,1,0,1,0,1,0,1,0,0,0,1,0,0,1,0,0,0,1,0,0,0,0,1,0,1,1,0,1,0,1,0,
                                       1,0,0,0,1,0,0,1,0,0,0,1,1,0,0,0};
    float x0[50]= {-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-
1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000, -
1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-
1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000, -1.0000,-
1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000,-1.0000};
    float I1[10];
    float Y1[11];
```

```
float Y21=0;          float Y22=0;   float Y23=0;
float Y2_der1;        float Y2_der2;          float Y2_der3;
float I21=0;          float I22=0;    float I23=0;
char resp;
int epoca=0;    int max_epoca=1000;    float aprendizado=0.1;
float e=0.000001;    float eqm=0;    float eqm1=0;    float eqm2=0;
float eqm3=0;    float eqm_ant=0;    float eqm_atual=0;    float diferenca=100;
float p=120;
float aux_w0[10],aux_w1[10],aux_w2[10],aux_w3[10];aux_w4[10];aux_w5[10];
float aux_w6[10]; aux_w7[10];aux_w8[10];aux_w9[10];aux_w10[10];
float delta_21=0;    float delta_22=0;    float delta_23=0;
float delta_1[11];    float soma_delta;    float Y_sai1[50];
float Y_sai2[50];    float Y_sai3[50];    float Y1_der;
float w0[10],w1[10],w2[10],w3[10],w4[10];w5[10]; w6[10];w7[10]; w8[10];w9[10];
w10[10];
float w_saida1[11];w_saida2[11];w_saida3[11];      float vetor_eqm[200];
int ve=0;
srand(time(NULL));//inicializando a semente do rand
for(int i=0;i<10;++i)//um rand para cada elemento
{
  w0[i]=((rand()%100)+0.1)/10;      w1[i]=((rand()%100)+0.1)/10;
  w2[i]=((rand()%100)+0.1)/10;      w3[i]=((rand()%100)+0.1)/10;
  w4[i]=((rand()%100)+0.1)/10;      w5[i]=((rand()%100)+0.1)/10;
  w6[i]=((rand()%100)+0.1)/10;      w7[i]=((rand()%100)+0.1)/10;
  w8[i]=((rand()%100)+0.1)/10;      w9[i]=((rand()%100)+0.1)/10;
  w10[i]=((rand()%100)+0.1)/10;
}

for (int g=0;g<11;g++) {
        w_saida1[g]=((rand()%100)+0.1)/100;
         w_saida2[g]=((rand()%100)+0.1)/100;
         w_saida3[g]=((rand()%100)+0.1)/100;
}

for (int k=0;k<10;k++) {
    aux_w0[k]=w0[k];       aux_w1[k]=w1[k];       aux_w2[k]=w2[k];
    aux_w3[k]=w3[k];       aux_w4[k]=w4[k];       aux_w5[k]=w5[k];
    aux_w6[k]=w6[k];       aux_w7[k]=w7[k];       aux_w8[k]=w8[k];
    aux_w9[k]=w9[k];       aux_w10[k]=w10[k];
}

       cout<<"\nTreinando..."<<'\n';

 for (int a=0;a<10;a++) //Zerando os vetores I e Y.
  {
    I1[a]=0;
  }

  for (int i=0;i<50;i++) {  //Zerando os vetores I e Y.
         Y_sai1[i]=0;        Y_sai2[i]=0;        Y_sai3[i]=0;      }
```

```
          for (int l=0;l<11;l++) {
        Y1[l]=0;
        delta_1[l]=0;
       }

    while(epoca<=max_epoca) {
      eqm=0; eqm1=0; eqm2=0; eqm3=0;
      I21=0; I22=0; I23=0;
      Y21=0;  Y22=0;  Y23=0;

       for(int j=0;j<50;j++) //Calculo de I1
         {
         for (int k=0; k<10;k++) {
           I1[k]=(w0[k]*x0[j])+(w1[k]*X1T[j])+(w2[k]*X2T[j])+(w3[k]*X3T[j])+
(w4[k]*X4T[j])+(w5[k]*X5T[j])+(w6[k]*X6T[j])+(w7[k]*X7T[j])+(w8[k]*X8T[j])+
(w9[k]*X9T[j])+ (w10[k]*X11T[j]);
          }

        Y1[0]=-1;
        for (int b=0;b<11;b++) //Calculo de Y1
          {
           Y1[b+1]= 1/(1+exp(-I1[b]));  //Função de Ativação
          }

       I21=0;   I22=0;  I23=0;
       for(int m=0;m<11;m++) //Calculo de I2
       {
          I21=I21+(Y1[m]*w_saida1[m]);
          I22=I22+(Y1[m]*w_saida2[m]);
          I23=I23+(Y1[m]*w_saida3[m]);
       }

      Y21=1/(1+(exp(-I21))); //Calculo de Y2
      Y22=1/(1+(exp(-I22)));
      Y23=1/(1+(exp(-I23)));

       //Valor do Eqm antes do Treino
       eqm1=eqm1+(0.5*((saida_d1[j]-Y21)*(saida_d1[j]-Y21)));
       eqm2=eqm2+(0.5*((saida_d2[j]-Y22)*(saida_d2[j]-Y22)));
       eqm3=eqm3+(0.5*((saida_d3[j]-Y23)*(saida_d3[j]-Y23)));
       eqm=(eqm1+eqm2+eqm3)/3;

  } //Fim do for j (fim de todas as amostras).

       //Calculo do Eqm Anterior
        eqm=eqm/p;         eqm_ant=eqm;

        //*************Camada Escondida***********
       I21=0;   I22=0;   I23=0;
       Y21=0;   Y22=0;   Y23=0;
```

```
      eqm1=0;  eqm2=0;  eqm3=0; eqm=0;
      for(int j=0;j<40;j++) {  //Calculo de I1
        for (int k=0; k<10;k++)
         {  I1[k]=(w0[k]*x0[j])+(w1[k]*X1T[j])+(w2[k]*X2T[j])+ (w3[k]*X3T[j])+
(w4[k]*X4T[j])+(w5[k]*X5T[j])+(w6[k]*X6T[j])+ (w7[k]*X7T[j])+ (w8[k]*X8T[j])+
(w9[k]*X9T[j])+(w10[k]*X10T[j]);
         }
      Y1[0]=-1;
      for (int b=0;b<11;b++) {
          Y1[b+1]= 1/(1+exp(-I1[b]));  //Calculo de Y1
        }
      I2=0;
      for(int n=0;n<11;n++) //Calculo de I2
      {
         I21=I21+(Y1[n]*w_saida1[n]);
         I22=I22(Y1[n]*w_saida2[n]);
         I23=I23+(Y1[n]*w_saida3[n]);
      }
      Y21=1/(1+(exp(-I21))); //Calculo de Y2
      Y22=1/(1+(exp(-I22)));
      Y23=1/(1+(exp(-I23)));
      Y2_der1=(Y21*(1-Y21)); //Derivada
      Y2_der2=(Y22*(1-Y22));
      Y2_der3=(Y23*(1-Y23));
      delta_21=(saida_d1[j]-Y21)*Y2_der1;  //Delta 2
      delta_22=(saida_d2[j]-Y22)*Y2_der2;
      delta_23=(saida_d3[j]-Y23)*Y2_der3;
      for (int r=0;r<11;r++) //Correção de Peso Saída
      {
        w_saida1[r] = w_saida1[r] + (aprendizado * delta_21 * Y1[r]);
        w_saida2[r] = w_saida2[r] + (aprendizado * delta_22 * Y1[r]);
        w_saida3[r] = w_saida3[r] + (aprendizado * delta_23 * Y1[r]);
      }
      soma_delta=0;
      for (int t=0;t<11;t++) {
         soma_delta=soma_delta + delta_21 * w_saida1[t];
         soma_delta=soma_delta + delta_22 * w_saida2[t];
          soma_delta=soma_delta + delta_23 * w_saida3[t];
      }
      for (int y=0; y<11;y++) {
          Y1_der=(Y1[y]*(1-Y1[y]));
          delta_1[y]=soma_delta*Y1_der;     //Delta 1
      }

     //Ajuste de Peso
     for (int a=0; a<10; a++) {
       w0[a] = w0[a] + (aprendizado * delta_1[a+1] * x0[j]);
       w1[a] = w1[a] + (aprendizado * delta_1[a+1] * X1T[j]);
       w2[a] = w2[a] + (aprendizado * delta_1[a+1] * X2T[j]);
       w3[a] = w3[a] + (aprendizado * delta_1[a+1] * X3T[j]);
```

```
        w4[a] = w4[a] + (aprendizado * delta_1[a+1] * X4T[j]);
        w5[a] = w5[a] + (aprendizado * delta_1[a+1] * X5T[j]);
        w6[a] = w6[a] + (aprendizado * delta_1[a+1] * X6T[j]);
        w7[a] = w7[a] + (aprendizado * delta_1[a+1] * X7T[j]);
        w8[a] = w8[a] + (aprendizado * delta_1[a+1] * X8T[j]);
        w9[a] = w9[a] + (aprendizado * delta_1[a+1] * X9T[j]);
        w10[a] = w10[a] + (aprendizado * delta_1[a+1] * X10T[j]);
       }
} // Fim do For j
      //Calculo do Eqm Atual
       I21=0;  I22=0;  I23=0;
       Y21=0;  Y22=0;  Y23=0;
       eqm=0;  eqm1=0;  eqm2=0;  eqm3=0;

       for(int j=0;j<40;j++) //Calculo de I1
       {
         for (int k=0; k<10;k++) {

                  I1[k]=(w0[k]*x0[j])+(w1[k]*X1T[j])+(w2[k]*X2T[j])+(w3[k]*X3T[j])+
(w4[k]*X4T[j])+(w5[k]*X5T[j])+(w6[k]*X6T[j])+ (w7[k]*X7T[j])+(w8[k]*X8T[j])+
(w9[k]*X9T[j])+(w10[k]*X10T[j]);
          }

      Y1[0]=-1;
      for (int b=0;b<11;b++) {  //Calculo de Y1
           Y1[b+1]= 1/(1+exp(-I1[b]));  }

      I21=0;       I22=0;       I23=0;
       for(int me=0;me<11;me++) //Calculo de I2
       {
          I21=I21+(Y1[me]*w_saida1[me]);
          I22=I22+(Y1[me]*w_saida2[me]);
          I23=I23+(Y1[me]*w_saida3[me]);
       }

       Y21=1/(1+(exp(-I21))); //Calculo de Y2
       Y_sai1[j]=Y21;
       Y22=1/(1+(exp(-I22)));
       Y_sai2[j]=Y22;
       Y23=1/(1+(exp(-I23)));
       Y_sai3[j]=Y23;
      eqm1=eqm1+0.5*((saida_d1[j]-Y21)*(saida_d1[j]-Y21));
       eqm2=eqm2+0.5*((saida_d2[j]-Y22)*(saida_d2[j]-Y22));
       eqm3=eqm3+0.5*((saida_d3[j]-Y23)*(saida_d3[j]-Y23));
       eqm=(eqm1+eqm2+eqm3)/3;

} //Fim do For J.
       eqm=eqm/p;
       eqm_atual=eqm;
       vetor_eqm[ve]=eqm;
```

```
        ve=ve+1;
        diferenca=(eqm_atual-eqm_ant);  //Criterio de Parada
        if(diferenca<0)
        diferenca=diferenca*(-1); //Modulo
       epoca=epoca+1;
        if (epoca == max_epoca) {
         cout<<'\n'<<"\n###### Nao Converge! #######"<<'\n';
         cout<<"\nEpoca = "<<epoca;
         diferenca=0;
        }
}//Fim do While Geral
if(diferenca<=e)
cout<<"\n **** Convergiu ***"<<'\n'<<'\n';
cout<<"\n***** FIM DOS CALCULOS ****"<<'\n'<<'\n';
cout<<"\n*** VETORES DE PESOS INICIAIS *** "<<'\n'<<'\n';
cout<<"\n||      Peso do Bias         ||"<<'\n'<<'\n';
for(int y=0;y<10;y++)
cout<<aux_w0[y]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X1              ||"<<'\n'<<'\n';
for(int x=0;x<10;x++)
cout<<aux_w1[x]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X2              ||"<<'\n'<<'\n';
for(int z=0;z<10;z++)
cout<<aux_w2[z]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X3              ||"<<'\n'<<'\n';
for(int t=0;t<10;t++)
cout<<aux_w3[t]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X4              ||"<<'\n'<<'\n';
for(int z=0;z<10;z++)
cout<<aux_w4[z]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X5              ||"<<'\n'<<'\n';
for(int z=0;z<10;z++)
cout<<aux_w5[z]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X6              ||"<<'\n'<<'\n';
for(int z=0;z<10;z++)
cout<<aux_w6[z]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X7              ||"<<'\n'<<'\n';
for(int z=0;z<10;z++)
cout<<aux_w7[z]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X8              ||"<<'\n'<<'\n';
for(int z=0;z<10;z++)
cout<<aux_w8[z]<<'\t';
```

```
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X9             ||"<<'\n'<<'\n';
for(int z=0;z<10;z++)
cout<<aux_w9[z]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X10            ||"<<'\n'<<'\n';
for(int z=0;z<10;z++)
cout<<aux_w10[z]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n*** VETORES DE PESOS FINAIS *** "<<'\n'<<'\n';
cout<<"\n||      Peso do Bias        ||"<<'\n'<<'\n';
for(int p=0;p<10;p++)
cout<<w0[p]<<'\t'; cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X1             ||"<<'\n'<<'\n';
for(int x=0;x<10;x++)
cout<<w1[x]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X2             ||"<<'\n'<<'\n';
for(int z=0;z<10;z++)
cout<<w2[z]<<'\t'; cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X3             ||"<<'\n'<<'\n';
for(int t=0;t<10;t++)
cout<<w3[t]<<'\t'; cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X4             ||"<<'\n'<<'\n';
for(int t=0;t<10;t++)
cout<<w4[t]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X5             ||"<<'\n'<<'\n';
for(int t=0;t<10;t++)
cout<<w5[t]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X6             ||"<<'\n'<<'\n';
for(int t=0;t<10;t++)
cout<<w6[t]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X7             ||"<<'\n'<<'\n';
for(int t=0;t<10;t++)
cout<<w7[t]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X8             ||"<<'\n'<<'\n';
for(int t=0;t<10;t++)
cout<<w8[t]<<'\t'; cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X9             ||"<<'\n'<<'\n';
for(int t=0;t<10;t++)
cout<<w9[t]<<'\t'; cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Peso X10            ||"<<'\n'<<'\n';
for(int t=0;t<10;t++)
cout<<w10[t]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Pesos de Saida          ||"<<'\n'<<'\n';
```

```
for(int s=0;s<11;s++)
cout<<w_saida1[s]<<'\t';
cout<<'\n'<<'\n'<<'\n';
for(int s=0;s<11;s++)
cout<<w_saida2[s]<<'\t';
cout<<'\n'<<'\n'<<'\n';
for(int s=0;s<11;s++)
cout<<w_saida3[s]<<'\t';
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Saidas              ||"<<'\n'<<'\n';
for(int k=0;k<50;k++)
cout<<'\t'<<Y_sai1[k];
cout<<'\n'<<'\n'<<'\n';
for(int k=0;k<50;k++)
cout<<'\t'<<Y_sai2[k];
cout<<'\n'<<'\n'<<'\n';
for(int k=0;k<50;k++)
cout<<'\t'<<Y_sai3[k];
cout<<'\n'<<'\n'<<'\n';
cout<<"\n||      Vetor EQM           ||"<<'\n'<<'\n';
for(int o=0;o<ve;o++)
cout<<'\t'<<vetor_eqm[o];
cout<<"\nNumero de Epocas Total = "<<epoca;
cout<<"\nErro Final = "<<eqm_atual;

//**Etapa de Teste da Rede**
   float y0[15]={-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};
   float y1[15]={1.0000,1.0000,1.0000,1.0000,1.0000, 1.0000,1.0000, 1.0000,1.0000,1.0000,
00000, 1.0000,1.0000,1.0000,1.0000};
   float y2[15]={1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,
0.0000,1.0000,1.0000,1.0000,1.0000};
   float y3[15]={0.9900,0.7400,0.8900,0.8000,0.8000,0.7400,0.7400,0.8000,0.8800,0.7400,
0.0000,0.7400,0.8000,0.8400,0.7400};
   float y4[15]={1.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000, 0.0000,
0.0000,0.0000,0.0000,0.0000,0.0000};
        float y5[15]={0.0000,1.0000,0.0000,1.0000,1.0000,0.0000,1.0000,1.0000,1.0000,
0.0000,0.0000,1.0000,1.0000,0.0000,1.0000};
        float y6[15]={0.5000,0.5000,0.5000,0.5000,0.3000,0.6000,0.5000,0.5000,0.2000,
0.5400,0.1000,0.5000,0.4000,0.5000,0.6000,0.7500};
        float y7[15]={0.0000,1.0000,0.0000,0.0000,1.0000,0.0000,0.0000,1.0000,1.0000,
0.0000,0.0000,0.0000,0.0000,0.0000,1.0000,0.0000};
        float y8[50]={0.0000,0.0000,0.0000,1.0000,0.0000,0.0000,0.0000,1.0000,0.0000,
0.0000,1.0000,0.0000,0.0000,0.0000,0.0000,0.0000};
        float y9[15]={0.5000,0.5000,0.5000,0.5000,0.3000,0.6000,0.5000,0.5000,0.2000,
0.5400,0.1000,0.5000,0.4000,0.5000,0.6000,0.7500};
        float y10[15]={1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,
1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000};
   float yd1[15]={1,1,1,0,1,1,1,0,1,1,1,0,1,1,1};
   float yd2[15]={0,0,0,0,1,0,1,0,0,0,0,0,1,0,0};
   float yd3[15]={1,0,1,1,0,1,0,1,0,1,0,0,0,1,0};
```

```
   float y_saida1[15];   float y_saida2[15];   float y_saida3[15];
  char resp1;
   cout<<"\n Deseja testar a rede? --> ";
   cin>>resp1;
   if ((resp1=='S')||(resp1=='s')) {
          cout<<"****************** TESTE **********************"<<'\n'<<'\n';
    I21=0; I22=0; I23=0;
    Y21=0; Y22=0; Y23=0;
    eqm=0;
    for(int j=0;j<20;j++) {    //Calculo de I1
       for (int k=0; k<10;k++) {
  I1[k]=(w0[k]*y0[j])+(w1[k]*y1[j])+(w2[k]*y2[j])+(w3[k]*y3[j])+(w4[k]*y4[j])+(w5[k]*
y5[j])+(w6[k]*y6[j])+(w7[k]*y7[j])+(w8[k]*y8[j])+(w9[k]*y9[j])+(w10[k]*y10[j]);
          }
      Y1[0]=-1;
      for (int b=0;b<11;b++)  {
           Y1[b+1]= 1/(1+exp(-I1[b]));          }  //Calculo de Y1
      I21=0; I22=0; I23=0;
       for(int me=0;me<11;me++)        {    //Calculo de I2
          I21=I21+(Y1[me]*w_saida1[me]);    I22=I22+(Y1[me]*w_saida2[me]);
          I23=I23+(Y1[me]*w_saida3[me]);          }
      Y21=1/(1+(exp(-I21))); //Calculo de Y2
      Y22=1/(1+(exp(-I22)));
      Y23=1/(1+(exp(-I23)));    y_saida1[j]=Y21;      y_saida2[j]=Y22; y_saida3[j]=Y23;
} //Fim do For J.
system("cls");
cout<<"\n**************TESTE*****************"<<'\n'<<'\n';
cout<<"\n  ##Amostra##"<<'\t'<<"   ##Saida da Rede##     "<<'\n'<<'\n';
for(int a=0;a<20;a++) {
     cout<<'\t'<<a+1;
    cout<<'\t'<<'\t'<<y_saida1[a]<<'\t';
    cout<<'\t'<<'\t'<<y_saida2[a]<<'\t';
    cout<<'\t'<<'\t'<<y_saida3[a]<<'\t'; cout<<'\n';  }
    cout<<"\n  ##Amostra##"<<'\t'<<"   ##Saida Desejada##      "<<'\n'<<'\n';
for(int a=0;a<20;a++)   {
   cout<<'\t'<<a+1;
   cout<<'\t'<<'\t'<<yd1[a]<<'\t';
   cout<<'\t'<<'\t'<<yd2[a]<<'\t';
   cout<<'\t'<<'\t'<<yd3[a]<<'\t'; cout<<'\n';  }
   }//Fim do if
} //Fim da função Treinar
```