MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2017/07.24.15.06-TDI

UMA ABORDAGEM PARA MINERAÇÃO DE LOGS PARA APOIAR A CONSTRUÇÃO DE APLICAÇÕES WEB ADAPTATIVAS

Leandro Guarino de Vasconcelos

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Rafael Duarte Coelho dos Santos, e Laércio Augusto Baldochi Júnior, aprovada em 24 de maio de 2017.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3PB998B>

INPE
São José dos Campos
2017

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
E-mail: pubtc@inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2017/07.24.15.06-TDI

UMA ABORDAGEM PARA MINERAÇÃO DE LOGS PARA APOIAR A CONSTRUÇÃO DE APLICAÇÕES WEB ADAPTATIVAS

Leandro Guarino de Vasconcelos

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Rafael Duarte Coelho dos Santos, e Laércio Augusto Baldochi Júnior, aprovada em 24 de maio de 2017.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3PB998B>

INPE
São José dos Campos
2017

Dados Internacionais de Catalogação na Publicação (CIP)

Vasconcelos, Leandro Guarino de.

V441a Uma abordagem para mineração de logs para apoiar a construção de aplicações web adaptativas / Leandro Guarino de Vasconcelos. – São José dos Campos : INPE, 2017.

xxiv + 120 p. ; (sid.inpe.br/mtc-m21b/2017/07.24.15.06-TDI)

Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2017.

Orientadores : Drs. Rafael Duarte Coelho dos Santos, e Laércio Augusto Baldochi Júnior.

1. Análise do comportamento do usuário. 2. Mineração de dados da web. 3. Mineração de uso da web. 4. Web adaptativa. I.Título.

CDU 004.6/.7

Aluno (a): *Leandro Guarino de Vasconcelos*

"UMA ABORDAGEM PARA MINERAÇÃO DE LOGS PARA APOIAR A CONSTRUÇÃO DE APLICAÇÕES WEB ADAPTATIVAS"

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

Computação Aplicada

Dr. Nandamudi Lankalapalli Vijaykumar

Presidente / INPE / SJCampos - SP

Dr. Rafael Duarte Coelho dos Santos

Orientador(a) / INPE / SJCampos - SP

Dr. Laércio Augusto Baldochi Júnior

Orientador(a) / UNIFEI / Itajubá - MG

Dra. Karine Reis Ferreira

Membro da Banca / INPE / São José dos Campos - SP

Dra. Maria da Graça Campos Pimentel

Convidado(a) / USP / São Carlos - SP

Dr. Tiago Silva da Silva

Convidado(a) / UNIFESP / São José dos Campos - SP

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 24 de maio de 2017

"A sabedoria é a coisa principal; adquire, pois, a sabedoria; sim, com tudo o que possuis, adquire o conhecimento.".

Rei Salomão
Provérbio bíblico

A Deus, à minha esposa ***Tálita****, a meus pais* ***Luiz Antonio*** *e* ***Margaret****, à minha tia* ***Margarida****, ao meu irmão* ***Luiz Eduardo****, à minha cunhada* ***Aline*** *e ao mais novo membro da família, meu sobrinho* ***João Pedro****.*

AGRADECIMENTOS

Agradeço a Deus pela motivação para desenvolver este trabalho, pelas oportunidades alcançadas e, principalmente, pelas pessoas mencionadas abaixo.

À minha esposa, Tálita, pelo amor, pelo carinho e cuidado, pela paciência e tolerância, por acreditar e por ser maravilhosa.

Aos meus orientadores, Dr. Rafael Duarte Coelho dos Santos, por aceitar iniciar esta jornada, pelas orientações, por confiar no meu trabalho e por se dedicar para a realização desta tese; e ao Dr. Laércio Augusto Baldochi Júnior, pela parceria desde o mestrado, por confiar na ideia, pelo empenho e pelas críticas que sempre motivam a busca pela excelência.

Aos meus pais, Luiz Antonio e Margaret, que semearam anos de ensino e exemplo de trabalho, de garra e de perseverança. Ao meu irmão, Luiz Eduardo, por confiar na ideia, por ser motivador e pela dedicação em viabilizar a execução deste trabalho. À minha cunhada, Aline, por ouvir pacientemente as ideias mirabolantes quando nos reunimos em família.

Aos amigos e irmãos da Igreja Cristã da Família de Aparecida por fazerem parte da minha vida neste intenso período.

À minha ex-chefe, Regina Célia Souza, da FEG/Guaratinguetá, por apoiar e incentivar o início do doutorado. Ao Dr. Marcelo dos Santos Pereira, diretor da FEG na época, pelo apoio e pelo incentivo. Aos professores Dr. Samuel Lucena e Dr. José Manoel Neves por recomendarem o meu ingresso no doutorado. Aos coordenadores da FATEC Guaratinguetá, Deborah Murgel, Manuela Vasconcelos e André Amarante, por serem pacientes e por apoiarem este trabalho.

Aos meus chefes, Ana Cláudia e Dr. Geilson Loureiro, pela paciência e pela disposição em colaborar. Aos meus colegas e amigos, Francisco Iakimoff, João Albertino, Maiara Flausino e Anderson Alvim, por colaborarem neste trabalho e pela compreensão.

Às pessoas que auxiliaram o desenvolvimento e os experimentos desta tese, Jonas Willian, Vladimir Geraseev Jr., Glauco Silva, Silvio Luís, Helton Franco. Em especial, ao CPTEC/INPE de Cachoeira Paulista, Dr. Antonio Ocimar Manzi, Dr. Gilvan Sampaio de Oliveira, Alexandre Oliveira, Marcos Araújo, Bianca Antunes e José Alberto.

RESUMO

Atualmente, há mais de 1 bilhão de *web sites* disponíveis. Neste enorme hiperespaço, há muitos *web sites* que fornecem o mesmo conteúdo ou serviço. Portanto, quando um usuário não encontra o que está procurando ou enfrenta dificuldades na interação, ele tende a procurar em outro *web site*. Para suprir as necessidades dos usuários atuais da Web, *web sites* adaptativos têm sido propostos. As abordagens de adaptação existentes geralmente adaptam o conteúdo das páginas de acordo com o interesse do usuário. Entretanto, a adaptação da estrutura da interface para atender às necessidades do usuário ainda necessita ser explorada. Nesta tese, uma abordagem é proposta para analisar o comportamento do usuário de aplicações Web durante a navegação, explorando a mineração de *logs* de cliente, chamada RUM (em inglês, *Real-time Usage Mining*). Nesta abordagem, as ações do usuário são coletadas na interface da aplicação e processadas de forma síncrona. Assim, a RUM é capaz de detectar problemas de usabilidade e padrões de comportamento para o usuário ativo, enquanto ele navega na aplicação. A fim de facilitar a implantação, a RUM fornece um *toolkit* que permite à aplicação consumir informações sobre o comportamento do usuário. Usando o *toolkit*, os desenvolvedores podem codificar adaptações que são automaticamente disparadas em resposta aos dados fornecidos pelo *toolkit*. Experimentos foram realizados em diferentes *web sites* para demonstrar a eficiência da abordagem em apoiar adaptações na interface que aprimoram a experiência do usuário.

Palavras-chave: Análise do comportamento do usuário. Mineração de dados da Web. Mineração de uso da Web. Web adaptativa.

AN APPROACH FOR MINING CLIENT LOGS TO SUPPORT THE CONSTRUCTION OF ADAPTIVE WEB APPLICATIONS

ABSTRACT

Currently, there are more than 1 billion websites available. In this huge hyperspace, there are many websites that provide exactly the same content or service. Therefore, when the user does not find what she is looking for easily or she faces difficulties during the interaction, she tends to search for another website. In order to fullfil the needs and preferences of today's web users, adaptive websites have been proposed. Existing adaptation approaches usually adapt the content of pages according to the user interest. However, the adaptation of the interface structure in order to meet user needs and preferences is still incipient. In this thesis, an approach is proposed to analyze the user behavior of Web applications during navigation, exploring the mining of client logs, called RUM (Real-time Usage Mining). In this approach, user actions are collected in the application's interface and processed synchronously. Thus, RUM is able to detect usability problems and behavioral patterns for the current application user, while she is browsing the application. In order to facilitate its deployment, RUM provides a toolkit which allows the application to consume information about the user behavior. By using this toolkit, developers are able to code adaptations that are automatically triggered in response to the data provided by the toolkit. Experiments were conducted on different websites to demonstrate the efficiency of the approach in order to support interface adaptations that improve the user experience.

Keywords: User behavior analysis. Web mining. Web usage mining. Web adaptive.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

AJAX	–	*Asynchronous Javascript and XML*
COP	–	*Container, Object and Page*
CPTEC	–	Centro de Previsão do Tempo e Estudos Climáticos
CSS	–	*Cascading Style Sheets*
FTP	–	*File Transfer Protocol*
HTML	–	*HyperText Markup Language*
HTTP	–	*HyperText Transfer Protocol*
INPE	–	Instituto Nacional de Pesquisas Espaciais
IP	–	*Internet Protocol*
JSON	–	*JavaScript Object Notation*
JSP	–	*Java Server Pages*
KDD	–	*Knowledge Discovery in Databases*
NoSQL	–	*Not-only SQL*
PHP	–	*PHP Hypertext Preprocessor*
RUM	–	*Real-time Usage Mining*
SQL	–	*Structured Query Language*
TAWS	–	*Toolkit for Adaptive WebSites*
URL	–	*Uniform Resource Locator*
WUM	–	*Web Usage Mining*

SUMÁRIO

Pág.

1 INTRODUÇÃO

A Web tem revolucionado a comunicação para instituições governamentais, empresas e pessoas. Com o crescimento do uso da Internet, a Web tem se tornado o meio predominante pelo qual as pessoas obtêm informações, seja, por exemplo, para realizar transações, entreter-se ou estudar. Devido ao rápido crescimento dos recursos disponíveis na Web, há um grande volume de informação e, consequentemente, muitos usuários se sentem perdidos durante a navegação. Hoje, portanto, os usuários priorizam encontrar informações de forma rápida e eficiente de acordo com seus interesses e suas necessidades.

O comportamento dos usuários é parcialmente influenciado pelas mudanças no cotidiano, que tornaram o tempo ainda mais precioso. As pessoas buscam economizar tempo através do uso de serviços *on-line* oferecidos pelo Governo ou por empresas privadas, além dos diversos aplicativos que podem ser usados em atividades simples, como comprar um lanche no final do dia. Além disso, o comportamento dos usuários na Web também é influenciado pelas tendências de personalização nos produtos e serviços que eles consomem. Por exemplo, hoje há empresas bancárias que oferecem serviços personalizados desde o tipo de conta até a disponibilidade do gerente responsável para atender os clientes. No cenário de móveis, há maior procura por móveis planejados sob medida, diferentemente de anos atrás em que havia apenas os modelos prontos disponíveis nas lojas. Nesse contexto, as mesmas pessoas que buscam produtos e serviços personalizados no cotidiano extrapolam esse comportamento para a navegação na Web.

Para atender às necessidades desses usuários, as aplicações Web [1] deveriam ser capazes de fornecer o conteúdo certo no momento certo para o usuário (VELASQUEZ; PALADE, 2008). Para isso, é necessário conhecer o usuário, pois, segundo um raciocínio do *marketing*, "mais dados sobre o cliente, mais conhecimento sobre seu comportamento" (KOTLER; ARMSTRONG, 2000). Sendo assim, analisar o comportamento do usuário em aplicações Web tem se tornado mais relevante.

O comportamento do usuário tem sido estudado para diferentes propósitos, tais como: avaliar a usabilidade (PAGANELLI; PATERNò, 2002; SANTANA; BARANAUSKAS, 2010; CARTA et al., 2011; VASCONCELOS; BALDOCHI, 2011; GONCALVES et al., 2016); analisar o conteúdo de interesse do usuário (GUNDUZ; OZSU, 2003; CHOI; LEE,

[1] Aplicação Web é qualquer *software* baseado na plataforma Web que seja acessado por um aplicativo navegador. Assim, o termo Aplicação Web abrange *web sites* estáticos ou dinâmicos, e sistemas de informação executados em Intranet ou Internet.

2009; GEEL et al., 2012; LIU et al., 2012); realizar recomendações através de sistemas recomendadores (AGHABOZORGI; WAH, 2009; LI; TAKANO, 2011; BOTHOREL; CHEVALIER, 2003; BURKE, 2003); realizar a predição do comportamento do usuário (KHALIL, 2008); personalizar a interação ou a interface (ANAND; MOBASHER, 2005); e construir aplicações Web adaptativas (VELASQUEZ; PALADE, 2008).

Para Velasquez e Palade (VELASQUEZ; PALADE, 2008), as aplicações Web adaptativas, que se adaptam às necessidades dos usuários, seriam a próxima geração do desenvolvimento Web. Porém, essa afirmação completará uma década e esse ainda não é o cenário global da Web. Há aplicações Web que se adaptam aos interesses do usuário, principalmente quanto ao conteúdo. As indicações de amigos e sugestões de páginas nas mídias sociais, as buscas geolocalizadas dos mecanismos de busca e as recomendações de produtos em *e-commerce* são exemplos que caracterizam aplicações Web adaptativas, cujas adaptações ocorrem durante a navegação do usuário. Quanto à interface, o único recurso comum nas aplicações Web é a adaptação ao tamanho da tela do dispositivo do usuário (computador, *tablet*, *smartphone*, relógio, etc.). Portanto, percebe-se que as aplicações Web apresentam limitações no que diz respeito à adaptação da estrutura e da interface durante a navegação.

Uma das dificuldades para transformar a Web em uma Web adaptativa é a diversidade de conhecimentos necessários para a construção de aplicações adaptativas, tais como: usabilidade, acessibilidade, experiência do usuário, *design* de interface, mineração de dados, entre outros. A principal dificuldade não é aprendê-los, mas sim o investimento necessário para manter um time de desenvolvimento Web capaz nesses conceitos, que custa mais caro que um time com desenvolvedores capazes apenas nas linguagens-padrão da Web e banco de dados. Além disso, o tempo para o desenvolvimento de uma aplicação Web adaptativa é maior que o de uma aplicação Web comum, pois há o esforço para entender o comportamento dos usuários e implementar as mudanças pertinentes.

Ghorab et.al. (GHORAB et al., 2013) indicam que há uma necessidade crescente de sistemas que oferecem serviços personalizados aos usuários em termos de conteúdo e apresentação. O processo de desenvolvimento de aplicações Web adaptativas abrange tarefas como a análise do comportamento do usuário, a identificação de seu interesse e/ou suas necessidades e a recomendação de conteúdo personalizado ou a adaptação da interface. No início da década passada, Huang et.al. (HUANG et al., 2002) já apontava a demanda de sistemas que identificam as necessidades do usuário e recomendam informações relevantes.

Para analisar o comportamento do usuário, é necessário conhecer as ações que ele realizou, portanto é essencial registrar a sua interação com a aplicação Web. O registro das ações pode ser feito através de *logs*. *Logs* são arquivos que armazenam o registro de ações de usuários ou de sistemas para uma análise posterior, que pode ser realizada por motivos de segurança ou para compreender o comportamento sobre os dados registrados. Na Web, há dois tipos de *logs* explorados para análise de comportamento do usuário: os *logs* de servidor e os *logs* de cliente. Os *logs* de servidor, coletados pelos servidores que hospedam as aplicações, armazenam dados sobre os recursos da aplicação acessados pelo usuário como páginas, imagens e outros arquivos. Já os *logs* de cliente, coletados no navegador do usuário, fornecem informações mais detalhadas sobre a interação, como movimentos do *mouse*, uso da barra de rolagem, ações de toque (em dispositivos com telas sensíveis ao toque), uso do teclado, entre outras. Portanto, os *logs* da Web contêm informações valiosas sobre o comportamento do usuário e o conteúdo que ele acessa. Os *logs* de cliente, possuem um volume de dados significativamente maior que os *logs* de servidor.

Os trabalhos reportados na literatura comumente utilizam *logs* de servidor. No entanto, esses *logs* contêm apenas dados sobre as decisões do usuário quando escolhem visitar uma página; já os *logs* de cliente contêm também ações que antecedem as decisões do usuário e, portanto, podem ser explorados para extrair mais conhecimento do que há em *logs* de servidor.

O estudo da extração de conhecimento de dados da Web foi nomeado na literatura por Cooley et. al. (COOLEY et al., 1997) como Mineração de Dados da Web (em inglês, *Web Mining*). Quando se trata da análise de comportamento do usuário, há a Mineração de Dados de Uso da Web (em inglês, *Web Usage Mining* - WUM). Técnicas de WUM têm sido amplamente usadas para extrair conhecimento de *logs*, devido à capacidade de fornecer informações úteis sobre o uso da aplicação (PIERRAKOS et al., 2003). A análise do comportamento do usuário através de *logs* pode fornecer percepções que levam à customização e personalização da experiência do usuário. Por isso, a WUM é de grande interesse para profissionais de *e-marketing* e *e-commerce* (AGHABOZORGI; WAH, 2009).

A partir do levantamento dos trabalhos reportados na literatura para exploração de *logs* da Web, observou-se que existem diversas motivações para pesquisas nesse campo, como melhorar a experiência de busca (SERDYUKOV, 2014), atender tipos específicos de aplicações Web (SARUKKAI, 2013; KUO et al., 2005), aprimorar a acessibilidade (APAOLAZA et al., 2013) ou ainda fazer a predição de páginas futuras

enquanto o usuário navega (MOBASHER et al., 1999; KHONSHA; SADREDDINI, 2011; SEN et al., 2016).

Nesta tese, é apresentada uma abordagem [2] para coleta, processamento e análise de *logs* de cliente durante a navegação do usuário para apoiar a construção de aplicações Web adaptativas, chamada RUM (*Real-time Usage Mining*). A abordagem proposta coleta os *logs* na aplicação Web e realiza dois tipos de análise durante a navegação: (i) a avaliação remota e automática de usabilidade, e (ii) a detecção de padrões de comportamento do usuário. Os padrões são detectados a partir de um processo automatizado de descoberta de conhecimento nos *logs*. Em seguida, o especialista [3] avalia os padrões detectados e associa-os aos perfis de usuários [4] conhecidos da aplicação. Enquanto o usuário navega, a aplicação Web pode consumir o resultado da avaliação de usabilidade e os padrões detectados, de forma que o desenvolvedor possa pré-programar ações de adaptação da interface de acordo com as necessidades do usuário.

Os estudos de caso realizados demonstram a capacidade da abordagem RUM para o processamento de *logs* de cliente durante a navegação do usuário, a fim de auxiliar as aplicações Web a se tornarem adaptativas. Um dos estudos de caso foi realizado no *web site* do Tempo, pertencente ao Instituto Nacional de Pesquisas Espaciais (INPE). No INPE, que preza em seus valores pela "interação permanente com a sociedade para atendimento de suas necessidades e divulgação dos resultados do Instituto, facilitando o acesso à informação, produtos e serviços gerados" (INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS, 2017), a personalização das aplicações Web disponíveis ao público pode auxiliar os cidadão no acesso à informação e, consequentemente, colaborar com o Instituto a cumprir o seu compromisso com a sociedade.

1.1 Desafios

A partir desta breve revisão, os dois maiores desafios que se tornam evidentes referem-se (i) ao processamento e à análise de *logs* durante a navegação do usuário e (ii) a uma abordagem que colabore com os desenvolvedores para transformarem as aplicações Web comuns em aplicações adaptativas.

[2] Uma abordagem é uma maneira de enfocar ou interpretar algo. Nesta tese, o assunto a ser interpretado é a construção de aplicações Web adaptativas.

[3] Especialista da aplicação é o profissional responsável por administrar a aplicação Web, podendo ser o próprio desenvolvedor. Em alguns contextos, o especialista é chamado de *webmaster*.

[4] Nesta tese, um perfil de usuário é o comportamento de um tipo de usuário sobre o qual o especialista tem conhecimento explícito ou tácito.

Processar *logs* não é trivial devido à heterogeneidade e ao volume das informações que eles contêm. O volume de dados dos *logs* de cliente é um fator que desencoraja a análise desses *logs* nos trabalhos reportados na literatura. Se o conhecimento sobre o usuário é diretamente proporcional à quantidade de informações sobre o seu comportamento, então há a motivação para buscar soluções que explorem esses *logs*. Ao custo computacional para processar *logs* de cliente, soma-se o desafio de capturar e analisar as ações do usuário em diferentes dispositivos como computadores, *tablets* e *smartphones*, nos quais a entrada das ações ocorre por teclado, *mouse* ou telas sensíveis ao toque.

Quanto à análise de *logs*, há o desafio de compreender o comportamento do usuário enquanto ele interage com a interface, pois, uma vez que esse desafio é superado, é possível tomar decisões para reter o visitante na aplicação Web. A tomada de decisões pode resultar em ações que auxiliem a navegação de um usuário que esteja enfrentando dificuldades ou em ações que levem o usuário a aprofundar seu conhecimento sobre a aplicação Web, como, por exemplo, recomendações de páginas para navegação. No intuito de auxiliar o usuário, há o desafio de avaliar a usabilidade da interface especificamente para o comportamento dele; já para aprofundar a experiência do usuário, o desafio é detectar padrões de comportamento durante a interação baseando-se no comportamento histórico dos usuários da aplicação.

Para auxiliar os desenvolvedores a construírem aplicações Web adaptativas ou transformarem as aplicações comuns em adaptativas, é essencial reduzir o esforço de implementação. Portanto, um desafio é tornar invisíveis a coleta e a análise de *logs*, o processo de descoberta de conhecimento em bases de dados e, essencialmente, a detecção de padrões de comportamento dos usuários.

Resumidamente, há quatro tópicos de pesquisa no centro desta tese. O primeiro refere-se à capacidade de processar *logs* durante a navegação do usuário, a fim de compreender o seu comportamento; o segundo relaciona-se com a capacidade de avaliar a usabilidade da interface para cada usuário enquanto a interação ocorre; o terceiro diz respeito à capacidade de detectar padrões de comportamento dos usuários durante a navegação; e o quarto refere-se à capacidade de auxiliar o desenvolvedor a transformar sua aplicação Web em uma aplicação Web adaptativa, contribuindo para evoluir a Web atual para uma Web adaptativa.

1.2 Contribuições desta tese

As contribuições desta tese estão relacionadas à abordagem RUM para análise do comportamento do usuário durante a navegação, a fim de apoiar a construção de aplicações Web adaptativas.

Como primeira contribuição, investigou-se a capacidade de processar *logs* de cliente durante a navegação do usuário com o objetivo de obter informações sobre o seu comportamento e retorná-las em tempo hábil para a aplicação Web. Esta contribuição é uma evolução da pesquisa iniciada com o desenvolvimento dos sistemas USABILICS (VASCONCELOS; BALDOCHI, 2011) e MOBILICS (GONCALVES et al., 2016), cujo objetivo era avaliar remota e automaticamente a usabilidade de aplicações Web após as interações de um grupo de usuários. Portanto, a primeira contribuição desta pesquisa é um mecanismo para coletar *logs* da aplicações Web em diversos dispositivos e processá-los de forma eficiente. Para isso, houve uma mudança na forma de armazenar e recuperar os *logs*, que passaram a ser estruturados como grafos.

Como segunda contribuição, esta tese propõe a avaliação remota e automática de usabilidade de aplicações Web durante a navegação do usuário. O método de avaliação de usabilidade proposto no sistema USABILICS é baseado na análise de tarefas reconhecidas pelo especialista da aplicação. Esse método calcula um índice de eficácia e eficiência da interface para cada interação, mas naquele sistema a avaliação era realizada somente após os usuários encerrarem a interação. No entanto, detectar dificuldades dos usuários durante a navegação é essencial para ajudá-los a cumprir os objetivos desejados na aplicação Web. Portanto, a segunda contribuição desta tese é um algoritmo proposto (VASCONCELOS et al., 2016) como evolução do método do sistema USABILICS para realizar a avaliação de usabilidade durante a navegação do usuário, a fim de prover informações que permitam aprimorar a usabilidade durante a interação do usuário. Assim, o índice de eficácia e eficiência é calculado durante a interação do usuário. Na literatura, não há pesquisas reportadas que avaliem a usabilidade da interface durante a interação.

Como terceira contribuição desta tese, procurou-se identificar padrões de comportamento dos usuários a fim de caracterizá-los. Para isso, explorou-se o processo de descoberta de conhecimento em bases de dados (em inglês, KDD - *Knowledge Discovery in Databases*), desde a seleção dos atributos até a interpretação e avaliação por um especialista. Um processo automatizado de KDD foi elaborado para explorar os *logs*. Portanto, conjuntos de atributos foram definidos e são extraídos diariamente dos *logs* para serem pré-processados, transformados e processados por algoritmos de

mineração de dados. Os padrões resultantes da mineração de dados são avaliados pelo especialista da aplicação Web, que define a relevância dos padrões detectados e pode selecionar um ou mais padrões e categorizá-los como perfis de usuários ou ações de adaptação a ser disparadas durante a navegação do usuário.

A quarta contribuição desta tese é a implementação de um *toolkit*, chamado TAWS (em inglês, *Toolkit for Adaptive Web Sites*), que reduz o esforço de implementação do desenvolvedor, pois o *toolkit* executa de forma invisível para a aplicação a coleta e a análise de *logs*, a avaliação de usabilidade e o processo automatizado de KDD. Portanto, o desenvolvedor concentra o esforço de implementação apenas nas adaptações que o especialista deseja disparar a partir das informações providas pelo *toolkit*.

1.3 Visão geral da tese

No Capítulo 2, são caracterizados os dois tipos de *logs* coletados em aplicações Web: os *logs* de servidor e de cliente. O capítulo também descreve as diferenças entre os dados coletados nesses tipos de *logs*, bem como suas vantagens e desvantagens. Pesquisas de coleta e análise de *logs* para diferentes fins são brevemente apresentadas, incluindo trabalhos relacionados à avaliação de usabilidade, análise de conteúdo, sistemas recomendadores, personalização Web e predição do comportamento do usuário. Além disso, são recuperados da literatura dois conceitos fundamentais para a tese no que se relaciona à análise de tarefas, os conceitos de *objetivo* e *tarefa*.

O Capítulo 3 discorre sobre o processo de descoberta de conhecimento em bases de dados, destacando as iterações que compõem esse processo. Dentre as etapas do processo de KDD, a etapa de mineração de dados é destacada devido à sua relação com a Web, que nomeia-se na literatura como Mineração de Dados da Web (em inglês, *Web Mining*). Sendo assim, são apresentadas as principais técnicas utilizadas nessa etapa do processo de KDD. Por fim, são listados exemplos de pesquisas em Mineração de Dados da Web reportadas na literatura.

O Capítulo 4 apresenta a abordagem RUM, detalhando sua arquitetura e seu funcionamento. Inicialmente, a função de cada módulo da arquitetura é brevemente descrita. Em seguida, de acordo com a ordem lógica do funcionamento da abordagem, são apresentados os métodos e conceitos concernentes à operação de cada módulo. Na segunda parte do capítulo, o *toolkit* TAWS é descrito, no intuito de detalhar como a abordagem RUM foi implementada para apoiar a construção de aplicações Web adaptativas. O capítulo apresenta detalhes da coleta de *logs*, o algoritmo para

análise de tarefas em tempo real e o funcionamento do processo automatizado de KDD. Finalizando a apresentação do TAWS, são apresentados os diferentes tipos de requisições que as aplicações podem fazer ao *toolkit* enquanto o usuário navega. Por fim, são considerados os diversos aspectos de contribuição desta tese em relação aos trabalhos reportados na literatura.

No Capítulo 5, são apresentados quatro estudos de caso realizados para demonstração da abordagem RUM no apoio à construção de aplicações Web adaptativas. Em todos os estudos de caso foram coletados *logs* em aplicações Web reais, isto é, aplicações Web já publicadas na Web e que são acessadas por milhares de usuários mensalmente. Cada estudo de caso enfatiza uma contribuição desta tese. O primeiro objetivou verificar a hipótese de analisar *logs* durante a navegação do usuário devido ao desafio de processar eficientemente um grande volume de dados. Uma vez que se verificou a validade dessa hipótese, o segundo estudo de caso objetivou avaliar o apoio a adaptações a partir da avaliação remota e automática de usabilidade durante a navegação. No terceiro, foi demonstrado o apoio à construção de adaptações a partir do processo automatizado de KDD, com a participação dos especialistas da aplicação Web. O último estudo de caso, de caráter foi realizado para avaliar a capacidade e a flexibilidade da biblioteca *jUsabilics* do ponto de vista de um desenvolvedor externo. Para isso, o desenvolvedor recebeu orientações mínimas para a implantação do *toolkit* em sua aplicação Web e avaliou as características da abordagem RUM e a facilidade de implantação do TAWS para adaptar a aplicação. Um questionário foi respondido pelo desenvolvedor e as respostas destacam a viabilidade de uso do *toolkit* para reduz o esforço de implementação.

Finalmente, o Capítulo 6 inicia com uma rápida revisão das contribuições desta tese antes de discutir a abordagem RUM em relação ao apoio à construção de aplicações Web adaptativas. Além disso, são apresentados os pontos de melhoria da pesquisa e a tese encerra com as sugestões de pesquisas relevantes para trabalhos futuros.

2 COLETA E ANÁLISE DE *LOGS* DA WEB

Este capítulo apresenta os conceitos, a relevância e os principais temas de pesquisa reportados na literatura quanto à coleta e análise de *logs* para caracterização do comportamento do usuário de aplicações Web.

2.1 Coleta de logs para análise do comportamento do usuário

A cada dia, uma enorme quantidade de informação é produzida por sistemas computacionais na Web. Grande parte dessa informação é armazenada em logs, mantendo o registro histórico sobre o uso de sistemas. Um *web site* popular, por exemplo, pode gerar *gigabytes* de *logs* em um curto período de tempo (VELASQUEZ; PALADE, 2008). Apesar de serem pouco explorados, esses dados são armazenados por motivos de segurança, de registro histórico de alertas de sistema, de registro de acesso a sistemas, entre outros. Outros tipos de aplicações podem manter seus próprios tipos de *logs*, como, por exemplo, FTP, sistemas gerenciadores de bancos de dados, sistemas de auditoria, entre outros - estes estão fora do escopo desta pesquisa.

Especificamente quando um usuário interage com uma aplicação Web, independentemente do dispositivo conectado à Internet, algumas informações são registradas pelos servidores que hospedam as aplicações, tais como: a data/hora de acesso a um recurso [1], o recurso que foi acessado, o endereço IP do dispositivo usado e o tipo de requisição do protocolo HTTP (*GET*, *POST*, *PUT* ou *DELETE*). Além dessas informações, alguns servidores Web (APACHE.ORG, 2016) armazenam em seus *logs* de acesso o código de resposta do recurso requisitado (por exemplo, o código 200 indica que o recurso foi acessado corretamente) e o tamanho em bytes do recurso acessado. Esses *logs* são chamados de *logs* de servidor (do inglês, *server logs*).

Na literatura, muitos trabalhos têm sido desenvolvidos sobre a análise de *logs* de servidor (Seção 2.2), pois uma vez que esses *logs* são processados é possível obter informações como o tempo de permanência de um usuário em uma página, os tópicos de interesse do usuário a partir das URL visitadas, as páginas visitadas pelo usuário até encontrar um conteúdo específico, as páginas que influenciam a saída dos usuários de um *web site*, entre outras. Segundo Géry e Haddad (2003), a maneira mais fácil de encontrar informações sobre a navegação dos usuários é usar *logs* de servidor.

Com a evolução das tecnologias da Web, a forma de se construir aplicações Web

[1] São arquivos hospedados no servidor Web, tais como: páginas Web, imagens, animações e vídeos.

mudou. Atualmente, com o aumento do uso de dispositivos móveis para acessar a Internet, há a necessidade de reduzir o tráfego entre o cliente e o servidor, considerando a limitação das conexões desses dispositivos (taxa de transmissão e volume de tráfego).

Sendo assim, com as tecnologias modernas de desenvolvimento Web, a forma de construção de aplicações não se dá apenas pela criação de uma página para cada conteúdo, mas principalmente pelo carregamento assíncrono de partes menores de uma página e pela criação de interfaces que não necessitam da comunicação contínua com o servidor.

Essa forma de construção de aplicações Web prejudica as abordagens que utilizam *logs* de servidor, que só armazenam aquilo que é requisitado pelo lado cliente. Portanto, se uma aplicação contém uma página cuja interação ocorre apenas no lado cliente sem fazer requisições ao servidor Web, a interação do usuário com essa página não é registrada nos *logs* de servidor. Se há menos requisições, há menos dados nos *logs* de servidor, comprometendo as abordagens que usam apenas esses *logs* para analisar o comportamento do usuário.

Neste contexto, há a necessidade de coletar as ações do usuário executadas no lado cliente, que podem ser detectadas no navegador Web. Os *logs* que registram os dados coletados no navegador Web são chamados *logs* de cliente (em inglês, *client logs*).

A principal diferença entre *logs* de cliente e *logs* de servidor Web é a granularidade dos dados sobre a interação do usuário. *Logs* de servidor não capturam alguns aspectos da interatividade do usuário, como ações de teclado ou movimento de *mouse* ou toques na tela. Em *logs* de cliente, essas ações, que antecedem o carregamento de um novo recurso do servidor, são registradas e, portanto, o volume de dados em *logs* de cliente é significativamente maior.

A coleta de *logs* de servidor é executada de forma nativa pelos servidores Web que hospedam as aplicações. Já para a coleta de *logs* de cliente, há a necessidade de implantar um software no *web site* para captura das ações do usuário. Os *logs* de servidor registram informações de todos os usuários simultaneamente e, portanto, separar as sessões dos usuários torna-se complexo, enquanto os *logs* de cliente já distinguem as ações de cada usuário, que é uma tarefa importante para a análise de usuários.

Para a coleta de *logs* de cliente, há diferentes abordagens: (a) instalação de softwares

no dispositivo do usuário; (b) softwares implantados como extensões e *plugins* no navegador (CHOI; LEE, 2009; GEEL et al., 2012; BORDINO et al., 2012); (c) softwares que necessitam de configurações específicas no navegador do usuário (HONG et al., 2001); e (d) *scripts* inseridos nas páginas da aplicação Web (RIVOLLI et al., 2008; SANTANA; BARANAUSKAS, 2010; VARGAS et al., 2010; VASCONCELOS; BALDOCHI, 2011; GOOGLE, 2011; AZZOPARDI et al., 2012; GONCALVES et al., 2016).

Dentre as diferentes abordagens para coleta de *logs* de cliente, as abordagens *a*, *b* e *c* são recomendadas para testes realizados com um número limitados de usuários, pois requerem o esforço do usuário no processo de instalação e/ou configuração. Em contrapartida, a abordagem *d* coleta as ações do usuário de maneira invisível ao usuário, podendo ser usada para capturar todas as interações de uma aplicação Web.

O objetivo de coletar *logs* da Web é analisá-los a fim de encontrar informações relevantes sobre o comportamento do usuário ou sobre o conteúdo das aplicações Web. Velasquez et al. (2003) considera que, de todos os dados disponíveis da Web, os mais relevantes para a análise do comportamento e das preferências do usuário durante a navegação são os *logs* e as páginas Web.

Na Seção 2.2, são apresentadas as diferentes abordagens e os principais objetivos reportados na literatura para a análise de *logs* da Web.

2.2 Análise de logs

Segundo Géry e Haddad (2003), analisar *logs* é um dos desafios mais importantes para fornecer serviços Web inteligentes. Santos et al. (2012) apresenta alguns objetivos para a análise de logs, tais como: otimização e monitoramento de recursos, análise de segurança e perfil do usuário, bem como a identificação de problemas de segurança na infraestrutura de servidores (geralmente relacionados a software) e, potencialmente, a identificação de atividades suspeitos de usuários.

Além desses objetivos, analisar *logs* de interação pode fornecer informações úteis que ajudam um engenheiro Web no aprimoramento da estrutura da aplicação Web, de forma que ele pode tornar a aplicação mais fácil de usar e mais rápida no futuro (KHASAWNEH; CHAN, 2006).

Entretanto, explorar *logs* não é uma tarefa trivial, pois a análise de grandes volumes de dados é computacionalmente custosa. Apesar da complexidade existente para a análise de logs, um dos fatos que tornam necessárias as pesquisas neste assunto é

encontrar informações sobre o usuário para conhecer seus interesses e suas necessidades. O ideal é que o próprio usuário forneça todas as informações, no entanto os usuários não se sentem confortáveis em fornecer seus dados pessoais, o que torna os métodos interativos de coleta de dados inviáveis (BOTHOREL; CHEVALIER, 2003) e, consequentemente, não há um volume relevante de informações pessoais dos usuários para analisar.

Na computação, a análise de *logs* tem sido um objeto de estudo importante devido à variedade de tópicos de interesse relacionados ao uso e à manipulação de *logs*, tais como: mineração de dados, inteligência artificial, armazenamento de dados, processamento de alto de desempenho, tráfego de dados. Zulkernine et al. (2013) afirma, por exemplo, que a análise de *logs* é crucial para recuperar o conhecimento sobre o estado do sistema na solução de problemas de desempenho.

Especificamente quanto à análise do comportamento do usuário, a literatura reporta trabalhos para diferentes propósitos, tais como: avaliação de usabilidade (HONG et al., 2001; PAGANELLI; PATERNò, 2002; RIVOLLI et al., 2008; SANTANA; BARANAUSKAS, 2010; VARGAS et al., 2010; CARTA et al., 2011; VASCONCELOS; BALDOCHI, 2011; GONCALVES et al., 2016), análise do conteúdo de interesse do usuário (GUNDUZ; OZSU, 2003; CHOI; LEE, 2009; GEEL et al., 2012; LIU et al., 2012), sistemas recomendadores (AGHABOZORGI; WAH, 2009; LI; TAKANO, 2011), personalização Web (ANAND; MOBASHER, 2005; BOTHOREL; CHEVALIER, 2003; BURKE, 2003) e predição do comportamento do usuário (KHALIL, 2008).

WAUTT (RIVOLLI et al., 2008), WELFIT (SANTANA; BARANAUSKAS, 2010) e o trabalho de Carta et al. (2011) são abordagens para avaliação remota e semi-automática de usabilidade, em que há a identificação manual de problemas de usabilidade a partir de ferramentas de visualização.

WebRemUSINE (PAGANELLI; PATERNò, 2002), WebHint (VARGAS et al., 2010), USABILICS (VASCONCELOS; BALDOCHI, 2011) e MOBILICS (GONCALVES et al., 2016) são abordagens para avaliação remota e automática de usabilidade baseada em análise de tarefas.

A análise de tarefas baseia-se na premissa de que os usuários acessam aplicações Web para cumprirem objetivos, tais como: realizar uma compra, comentar uma notícia ou fazer o *download* de um material de estudo. Esses objetivos geralmente são divididos em uma ou mais tarefas. Por isso, as definições dos conceitos de tarefa e objetivo são importantes:

- Tarefas (em inglês, *tasks*) são atividades realizadas para se alcançar um objetivo. Elas podem ser atividades lógicas como "Acessar as informações sobre os filmes da programação de hoje à noite", ou atividades físicas como "Selecionar o botão no canto superior esquerdo da página" (PATERNò, 2002).

- Um objetivo (em inglês, *goal*) pode ser uma modificação do estado de uma aplicação ou uma tentativa de recuperar uma informação de uma aplicação. Por exemplo, "Acessar os vôos disponíveis" é um objetivo que não requer a modificação do estado da aplicação, porém "Acessar os vôos disponíveis para adicionar uma reserva" requer uma modificação do estado da aplicação (PATERNò, 2002). É notório que tarefas e objetivos estão diretamente ligados, sendo que uma tarefa pode corresponder a um objetivo, mas também um objetivo pode ser composto de múltiplas tarefas.

Uma abordagem comum para a análise de tarefas é a definição de um caminho ótimo para cada tarefa. O termo "caminho ótimo" foi definido por Paternò (PATERNò; PAGANELLI, 2001), que é a sequência de ações essenciais para que o usuário complete uma determinada tarefa. A análise de tarefas compara o caminho percorrido pelo usuário na interface e as ações que compõem o caminho ótimo da tarefa.

WebRemUSINE e WAUTER utilizam uma notação para definição de tarefas, já as demais abordagens permitem a definição de tarefas através da navegação do usuário na interface. USABILICS e MOBILICS propõem um modelo de interface para generalização de caminhos de uma tarefa, de forma que o avaliador não necessita gravar todos os diversos possíveis caminhos que um usuário pode percorrer na interface.

Gunduz e Ozsu (2003) propõem um método para identificação do interesse do usuário baseado no tempo de navegação em cada página. Choi e Lee (2009) analisam *logs* de cliente para criar perfis de usuários analisando a interação entre os diferentes sites que cada um visita. Geel et al. (2012) utiliza técnicas de *web scraping* [2] e fornecem uma ferramenta para o usuário organizar o conteúdo de interesse enquanto navega em um *web site*. Liu et al. (2012) utiliza os textos usados pelos usuários em caixas de busca para identificar o conteúdo de interesse.

No que diz respeito a sistemas recomendadores, Aghabozorgi e Wah (2009) propõem uma abordagem híbrida que usa os dados da interação do usuário e o conteúdo do

[2] *Web scraping* é o uso de técnicas de raspagem de dados para extração de dados de páginas Web.

web site para construir um modelo de recomendação. A abordagem de Li e Takano (2011) consiste em monitorar a interação do usuário, atribuir pesos às páginas do *web site* a partir do conteúdo e criar as preferências do usuário para recomendações posteriores.

Quanto à personalização Web, Bothorel e Chevalier (2003) realizam a recomendação de links ao usuário usando técnicas de predição. Burke (2003) realiza personalização Web a partir de recomendações colaborativas, baseadas na comparação dos históricos dos usuários. Khalil (2008) combina técnicas de mineração de dados para realizar a predição de páginas Web a partir da navegação do usuário.

2.3 Considerações finais

Neste capítulo, foram apresentadas as motivações que levam à análise de logs, tendo em vista que os usuários geralmente não são cooperativos quanto a fornecer seus dados pessoais. Foram apresentados os principais trabalhos reportados na literatura relacionados à coleta e análise de *logs* de cliente que objetivam analisar o comportamento do usuário.

Logs contêm dados importantes que, se forem analisados adequadamente, podem fornecer informações valiosas para as empresas e os negócios. Especificamente os *logs* da Web registram dados importantes sobre o comportamento do usuário. Portanto, analisar *logs* é essencial para a extração de conhecimento a partir dos dados coletados, a fim de suportar decisões estratégicas. Neste contexto, a mineração de dados destaca-se na literatura devido à capacidade de processar o grande volume de dados gerados pelas aplicações Web, motivando o desenvolvimento de diferentes técnicas e algoritmos.

3 MINERAÇÃO DE DADOS

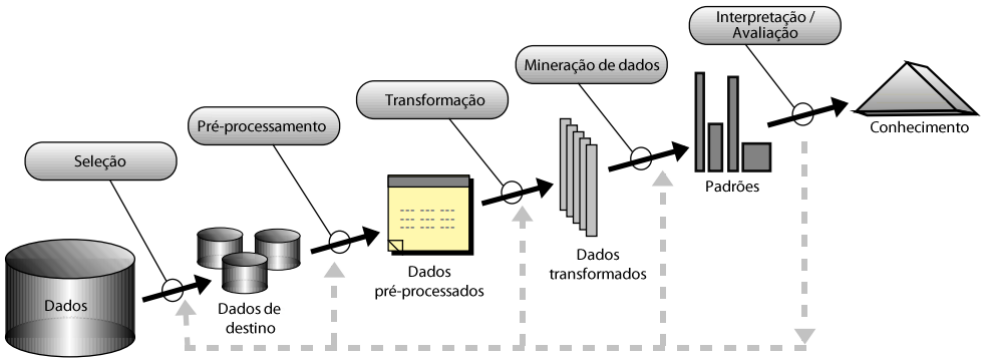
Este capítulo apresenta os conceitos de Descoberta de Conhecimento em Bases de Dados, destacando a Mineração de Dados da Web (em inglês, *Web Mining*), que explora a análise de *logs* para detecção de padrões do comportamento do usuário. Inicialmente, são definidos conceitos relevantes quanto à Descoberta de Conhecimento em Bases de Dados, assunto no qual a mineração de dados está inserida.

3.1 Descoberta de Conhecimento em Bases de Dados

KDD (*Knowledge Discovery in Databases*) é definido por Frawley et al. (1992) como "o processo de extração não-trivial de informações dos dados, informações que estão implicitamente presentes nos dados, previamente desconhecidas e potencialmente úteis para o usuário", e é o processo genérico para encontrar informação em grandes conjuntos de dados.

Segundo Fayyad et al. (1996a), o processo de KDD é composto por cinco passos: Seleção, Pré-processamento, Transformação, Mineração de Dados e Interpretação e avaliação, que são mostrados na Figura 3.1.

Figura 3.1 - Fases do processo de Descoberta do Conhecimento em Bases de Dados.



O processo de KDD é iterativo e os passos anteriores podem ser retomados quando necessário. A seguir, há uma breve descrição do que ocorre em cada passo do processo.

a) **Seleção**

- **Compreensão do domínio da aplicação:** inclui conhecimento relevante e os objetivos da aplicação;

- **Criação de conjunto de dados para descoberta:** inclui selecionar um conjunto de dados ou focar em um subconjunto dos dados disponíveis em que a descoberta será realizada;

b) **Pré-processamento**

- **Limpeza e pré-processamento dos dados:** inclui tarefas básicas como remover ruídos ou desvios, se necessário;

c) **Transformação**

- **Redução e reprojeção:** inclui encontrar características úteis para representar os dados, dependendo do objetivo da tarefa, e usar método de redução ou transformação de dimensionalidade para reduzir o número efetivo de variáveis que devem ser analisadas;

d) **Mineração de Dados**

- **Escolha da técnica de mineração de dados** inclui definir a técnica de mineração adequada aos dados, tais como: agrupamento, classificação, regressão, descoberta regras de associação, entre outras. Essas técnicas são apresentadas na Seção 3.2;
- **Escolha dos algoritmos de mineração e de seus parâmetros:** inclui selecionar um algoritmo referente à técnica escolhida e definir seus parâmetros;
- **Mineração de dados:** inclui a busca pelos padrões no conjunto de dados;

e) **Interpretação e avaliação**

- **Interpretação:** inclui interpretar os padrões detectados e, se necessário, retornar a alguns dos passos anteriores;
- **Consolidação e avaliação:** inclui incorporar os resultados em um sistema ou simplesmente documentá-los.

Como é possível observar na definição de Fayyad et al. (1996a), a mineração de dados é apenas uma etapa do processo de KDD, isto significa que o sucesso da descoberta de conhecimento através da mineração depende da correta execução dos outros passos.

A fim de detectar padrões de informação ocultos, a mineração de dados pode ser descritiva - identifica propriedades gerais dos dados que estão sendo analisados - ou preditiva - constrói modelos sobre os dados disponíveis. Para a execução da mineração de dados, há diversas técnicas que podem ser aplicadas, e estas são apresentadas na Seção 3.2.

3.2 Técnicas de Mineração de Dados

As técnicas mais conhecidas de mineração de dados incluem: classificação, regressão, agrupamento e descoberta de regras de associação. Essas técnicas são descritas nas seções seguintes.

3.2.1 Classificação

Dado um conjunto de objetos, técnicas de classificação ordenam objeto em categorias ou classes distintas (DUDA; STORK, 2000). Isso envolve dois passos: aprendizagem e classificação. Durante a aprendizagem (treinamento), uma amostra aleatória de dados é usada para testar um modelo proposto, comparando as classes encontradas com as classes reais a que os objetos pertencem.

Nesta técnica, é comum o uso de redes neurais artificiais. Pelo treinamento de uma rede neural artificial, a saída pode ser usada como um classificador de vetor característico. Por exemplo, para a classificação de clientes de um banco quanto ao risco, a camada de entrada recebe um vetor n-dimensional com as circunstâncias econômicas de uma pessoa e, na camada de saída, ela pode ser classificada como “risco” ou “não-risco”, dependendo do neurônio resultante ser fechado para “0” ou “1”.

Além das redes neurais artificais, há também o uso de árvores de decisão. Uma árvore de decisão (BREIMAN et al., 1984) é uma estrutura de dados para a classificação de um conjunto de dados, sendo que cada nível da árvore usa um critério de seleção. Por exemplo, em um conjunto de dados com a idade, o salário e o estado civil de algumas pessoas, a árvore de decisão teria três níveis, um para cada atributo. Além disso, os ramos da árvore em cada nível seriam separados por critérios referentes aos valores de cada atributo. No nível da árvore referente à idade, pode ocorrer a separação entre pessoas com menos de sessenta anos e pessoas com sessenta anos ou mais.

Exemplos de algoritmos desta técnica são: J48 (QUINLAN, 1993), Id3 (QUINLAN, 1986) e NaiveBayes (JOHN; LANGLEY, 1995).

3.2.1.1 Regressão

Segundo Fayyad et al. (1996b), regressão é aprender uma função que mapeia um item de dados para uma variável de predição de um valor real. Alguns exemplos de aplicação dessa técnica são: predição da quantidade de biomassa presente em uma floresta dada medições de microondas de sensoriamento remoto, predição da demanda de um novo produto como resultado das despesas publicitárias e previsão de séries temporais em que as variáveis de entrada podem ser versões temporizadas da variável de previsão.

Assim como para a técnica de classificação, na técnica de regressão também podem ser usados algoritmos de redes neurais artificiais. Dado um conjunto de exemplos de treinamento, uma rede neural artificial pode ser treinada para representar a função de aproximação em que modela uma situação. Quando surgem novos exemplos de entrada, a rede neural fornece a melhor predição baseada no que foi aprendido usando o conjunto de treinamento de exemplo.

3.2.2 Agrupamento

Agrupamento (ou clusterização) é o processo de agrupar objetos com característica similares, geralmente como um processo de aprendizagem não-supervisionada. Em aprendizagem supervisionada, o rótulo de uma classe é conhecido antecipadamente. A principal ideia de agrupamento é identificar classes de objetos (*clusters*) que são homogêneos dentro de cada classe e heterogêneos entre diferentes classes (RASMUSSEN, 1992). Para ocorrer o agrupamento, é necessária uma medida de similaridade que permita comparar cada par de itens da base de dados (JAIN et al., 1999).

Os principais algoritmos desta técnica são: DBScan (ESTER et al., 1996), SimpleKMeans (ARTHUR; VASSILVITSKII, 2007) e XMeans (PELLEG; MOORE, 2000).

3.2.3 Descoberta de regras de associação

Algoritmos de regras de associação (AGRAWAL et al., 1993) encontram padrões interessantes em bases de dados transacionais. Em um loja, por exemplo, uma transação é um conjunto de itens comprados por um cliente. Quando são analisadas diversas transações, podem ser detectadas regras de associação entre os itens, como, por exemplo, "a compra de cerveja está associada à compra de fralda". Regras de associação são tipicamente da forma $X \rightarrow Y$, onde X e Y são subconjuntos disjuntos de todos os itens disponíveis. Geralmente, dois parâmetros são usados pelos algoritmos que identificam regras de associação: o suporte e a confiança, que serão definidos a

seguir.

Definição 1: Seja $I = I_1, I_2, ..., I_m$ um conjunto de m atributos distintos, também chamado literais. Seja uma base de dados D um conjunto de transações, em que cada transação T é um conjunto de items tal que $T \subseteq I$. Um regra de associação é uma implicação na forma $X \rightarrow Y$, onde X, $Y \subset I$, são conjuntos de itens chamados *itemsets*, e $X \cap Y = \emptyset$. Aqui, X é chamado antecedente e Y, consequente.

Definição 2: O suporte (s) de uma regra de associação é a taxa (em percentual) entre os registros que contêm $X \cup Y$ e o número total de registros da base de dados. Portanto, se dissermos que o suporte de uma regra de associação é 5%, então isso significa que 5% do total de registro contêm $X \cup Y$.

Definição 3: Para um determinado número de registros, a confiança (γ) é a taxa (em percentual) entre o número de registros que contêm $X \cup Y$ e o número de registros que contêm X. Então, se dissermos que uma regra de associação tem confiança de 85%, isso significa que 85% dos registros que contêm X também contêm Y. A confiança indica o grau de correlação entre X e Y no conjunto de dados. A confiança é medida de força da associação. Geralmente, um valor alto de confiança é requerido para regras de associação. Um dos algoritmos mais usados para identificação de regras de associação é o algoritmo Apriori (AGRAWAL et al., 1993), que usa as medidas de suporte e confiança. Há diversos reportados na literatura que estendem ou adaptam esse algoritmo para a aplicação em contextos específicos, tais como: Improved Apriori (SHIRGAONKAR et al., 2010), R-Apriori (RATHEE et al., 2015), pcApriori (SCHLEGEL et al., 2013) e *Predictive Apriori* (SCHEFFER, 2001).

Regras de associação comuns não levam em conta a ordem em que os *itemsets* aparecem no registro da base de dados. No entanto, quando são analisadas bases de dados transacionais em que os *itemsets* de um registro possuem uma relação de precedência, como os *logs* de cliente e de servidor de aplicações Web, os algoritmos de regras de associação precisam ser adaptados para considerar a dimensão temporal dos dados. Assim, os algoritmos que detectam regras de associação em conjuntos de dados sequenciais são divididos em algoritmos para detecção de padrões sequenciais (do inglês, *sequential patterns*) e padrões sequenciais contíguos (do inglês, *contiguous sequential patterns*).

O problema de mineração de padrões sequenciais foi introduzido por Agrawal e Srikant (1994). Dado um conjunto de sequências, onde cada sequência consiste de uma lista de elementos e cada elemento consiste de um conjunto de itens, dado um

limite mínimo de suporte, a mineração de padrões sequenciais é usada para encontrar todas as subsequências frequentes cuja frequência de ocorrência é maior que o limite mínimo de suporte.

Padrões sequenciais contíguos são uma variação de padrões sequenciais em que os itens de uma sequência que contém o padrão devem ser adjacentes, considerando a ordenação dos dados. Para Chen e Cook (2007), padrões sequenciais contíguos são mais efetivos quando comparados a padrões sequenciais para aplicações tal como recomendação Web. No entanto, padrões sequenciais contíguos geralmente são mais raros no comportamento dos usuários, pois são detectados quando usuários realizam as ações na mesma ordem.

Com as possibilidades de descoberta do conhecimento criadas pelas técnicas de mineração de dados, os dados da Web tornaram-se atrativos e as técnicas de Mineração de Dados da Web emergiram como um resultado da aplicação da teoria de mineração de dados, principalmente para a mineração de *logs* de aplicações Web (CHANG et al., 2003; SPILIOPOULOU, 1999; LINOFF; BERRY, 2001).

A Seção 3.3 apresenta os conceitos de Mineração de Dados da Web, sua classificação e destaca a Mineração de Dados de Uso da Web.

3.3 Mineração de Dados da Web

Cooley et al. (1997) definiram Mineração de Dados da Web (em inglês, *Web Mining*) como "a descoberta e análise de informações úteis da *World Wide Web*". Esse termo é também definido como "a aplicação de técnicas de mineração de dados para grandes repositórios de dados Web" (COOLEY et al., 1999).

Velasquez e Palade (2008) classifica as técnicas de Mineração de Dados da Web em três categorias:

- **Mineração de Conteúdo da Web (do inglês, *Web Content Mining*):** o conteúdo da página Web é usado como entrada dos algoritmos;
- **Mineração de Estrutura da Web (do inglês, *Web Structure Mining*):** os algoritmos usam como entrada a estrutura das páginas da aplicação Web e os *links* entre elas;
- **Mineração de Uso da Web (do inglês, *Web Usage Mining*):** utiliza os *logs* de aplicações Web em conjunto com informações externas, tais como: idade, sexo, preferências de assuntos, etc.

Mineração de Dados da Web não é uma tarefa trivial, considerando que a Web é uma enorme coleção de dados heterogêneos, não-rotulados, distribuídos, variantes no tempo e semi-estruturados (VELASQUEZ; PALADE, 2008). Sendo assim, o processo de Mineração de Dados da Web deve considerar os passos do KDD definidos por Fayyad et al. (1996a), principalmente o pré-processamento, a descoberta de padrões e a análise desses padrões (SRIVASTAVA et al., 2000).

Para a análise do comportamento do usuário, assunto em que está inserida esta pesquisa, as técnicas de Mineração de Uso da Web têm sido mais comumente utilizadas, destacando-se a geração de regras de associação e a detecção de padrões sequenciais. As categorias citadas anteriormente são detalhadas nas Seções 3.3.1 a 3.3.3 a fim de especificar o escopo pertencente a cada categoria.

3.3.1 Mineração de Conteúdo da Web

A Mineração de Conteúdo da Web abrange a descoberta de informação útil e extração de conhecimento a partir do conteúdo da Web (KLEFTODIMOS; EVANGELIDIS, 2013). Essencialmente, os dados para Mineração de Conteúdo da Web são textos livres dentro de uma página Web. Este processo é semelhante ao de mineração de texto e os algoritmos utilizados são semelhantes aos utilizados na área de Recuperação de Informação. Os principais desafios de pesquisa nesta categoria são (SRIVASTAVA et al., 2005):

- Descoberta de tópicos, que usa principalmente as técnicas de agrupamento e regras de associação;
- Extração de padrões associativos;
- Agrupamento de páginas Web; e
- Classificação de páginas Web.

3.3.2 Mineração de Estrutura da Web

A Mineração de Estrutura da Web explora a topologia dos *hyperlinks* do *web site*. É usada para classificar páginas Web em páginas relevantes e páginas de transição para gerar informações sobre a similaridade entre diferentes *web sites* (PATIL; PATIL, 2012). Além disso, esta categoria também abrange a classificação de páginas em relação a um determinado assunto para fornecer informações sobre o interesse do usuário.

Alguns exemplos de aplicação da Mineração de Estrutura da Web abrangem:

- Correlação da estrutura do *web site* com problemas de usabilidade (LI; KIT, 2005). Os autores propuseram o algoritmo AWA (em inglês, *Adaptive Window Algorithm*), que extrai automaticamente estruturas de navegação em um *web site* sem realizar uma análise do hipertexto;

- Extração da arquitetura da informação de *web sites* (KELLER; NUSSBAUMER, 2012), em que os autores desenvolveram o algoritmo MenuMiner, que permite recuperar a estrutura principal de conteúdo de *web sites* de larga escala; e

- Análise da navegabilidade em *web sites* (ALQURASHI; WANG, 2014), em que os autores desenvolveram uma metodologia de análise baseada em grafos, cujos nós representam as páginas do *web site* e as arestas representam os *links* entre as páginas.

3.3.3 Mineração de Uso da Web

O termo *Web Usage Mining* foi introduzido por Cooley et al. (1997) quando uma primeira taxonomia de *Web Mining* foi feita. Os autores definiram o termo como "descoberta automática de padrões de acesso do usuário a partir de servidores Web" Cooley et al. (1997).

Velasquez e Palade (2008) afirmam que o objetivo da Mineração de Uso da Web é descobrir padrões usando diferentes técnicas de mineração de dados, tais como: análise estatística, regras de associação, agrupamento, classificação e detecção de padrões sequenciais. Cooley et al. (1999) afirmam que as técnicas mais usadas para Mineração de Uso da Web são regras de associação, padrões sequenciais e agrupamento.

As principais motivações que aumentam o interesse da comunidade científica em Mineração de Uso da Web são o aumento da complexidade dos *web sites* e a possibilidade de personalização de conteúdo Web. Para Sudhamathy (2010), os *logs* da Web precisam ser analisados para identificar as tendências de uso e de acesso e para fornecer informações úteis para os desenvolvedores Web e administradores, a fim de criar *web sites* adaptativos.

Algumas aplicações de Mineração de Uso da Web são:

- Agrupamento de usuários baseado em padrões de interesse (CHEN et al., 2006). Neste trabalho, foi implementado um algoritmo chamado COWES, que agrupa os usuários a partir de padrões detectados no histórico de navegação dos usuários do *web site*;

- Construção de sequências de acesso frequentes (LU; EZEIFE, 2003), em que os autores propuseram um algoritmo eficiente para detecção de sequências de acesso frequentes em *logs* de servidor;

- Aprimoramento do design de *web sites* (MASSEGLIA et al., 1999). Neste trabalho, foi usada a técnica de detecção de regras de associação para detectar padrões sequenciais relevantes. Com os padrões detectados, o desenvolvedor pode melhorar a estrutura de navegação do *web site* dinamicamente;

- Predição da navegação do usuário, usando cadeias de Markov (LIU et al., 2007) ou regras de associação (KHALIL, 2008);

- Construção de *web sites* adaptativos (VORA; BOJEWAR, 2011; RAMYA; SAJEEV, 2015).

3.4 Considerações finais

Neste capítulo, foram apresentados os conceitos e as definições relacionadas à mineração de dados, iniciando pelos passos do processo de KDD proposto por Fayyad et al. (1996a) e descrevendo as principais técnicas de mineração de dados. Em seguida, foram discutidos os conceitos relacionados à Mineração de Dados da Web, enfatizando a Mineração de Uso da Web, que é o tema da mineração de dados pertinente a este trabalho.

O Capítulo 4 apresenta a abordagem RUM (do inglês, *Real-time Usage Mining*), detalhando os conceitos de análise de *logs* e mineração de dados da Web usados, a arquitetura proposta, bem como o arcabouço tecnológico para a implementação de um *toolkit* como apoio para a construção de aplicações Web adaptativas.

4 ABORDAGEM RUM

Este capítulo detalha a principal contribuição deste trabalho, que é a abordagem RUM, cujos objetivos são realizar a mineração de *logs* de cliente coletados e avaliar a usabilidade em aplicações Web a fim de apoiar adaptações em durante a navegação do usuário. O capítulo está dividido em três partes: a primeira parte apresenta a arquitetura e o funcionamento da abordagem proposta nesta tese; a segunda parte aponta os detalhes da implementação de um *toolkit*[1] que foi usado para validar e demonstrar a abordagem; por fim, a terceira parte relata os trabalhos reportados na literatura mais relacionados à abordagem RUM.

4.1 Arquitetura e funcionamento

A RUM foi projetada para coletar, armazenar, processar e analisar *logs* de cliente gerados em quaisquer aplicações Web, sejam estáticas ou dinâmicas, e fornecer informações sobre o comportamento do usuário durante a navegação. Para isso, a arquitetura da RUM está estruturada em cinco módulos, destacados na Figura 4.1, que são:

a) **Coleta de *logs*:** coleta e armazena as ações dos usuários detectadas na aplicação Web;

b) **Análise de tarefas:** realiza a avaliação remota e automática de usabilidade durante a navegação baseada em tarefas previamente definidas pelo especialista da aplicação Web;

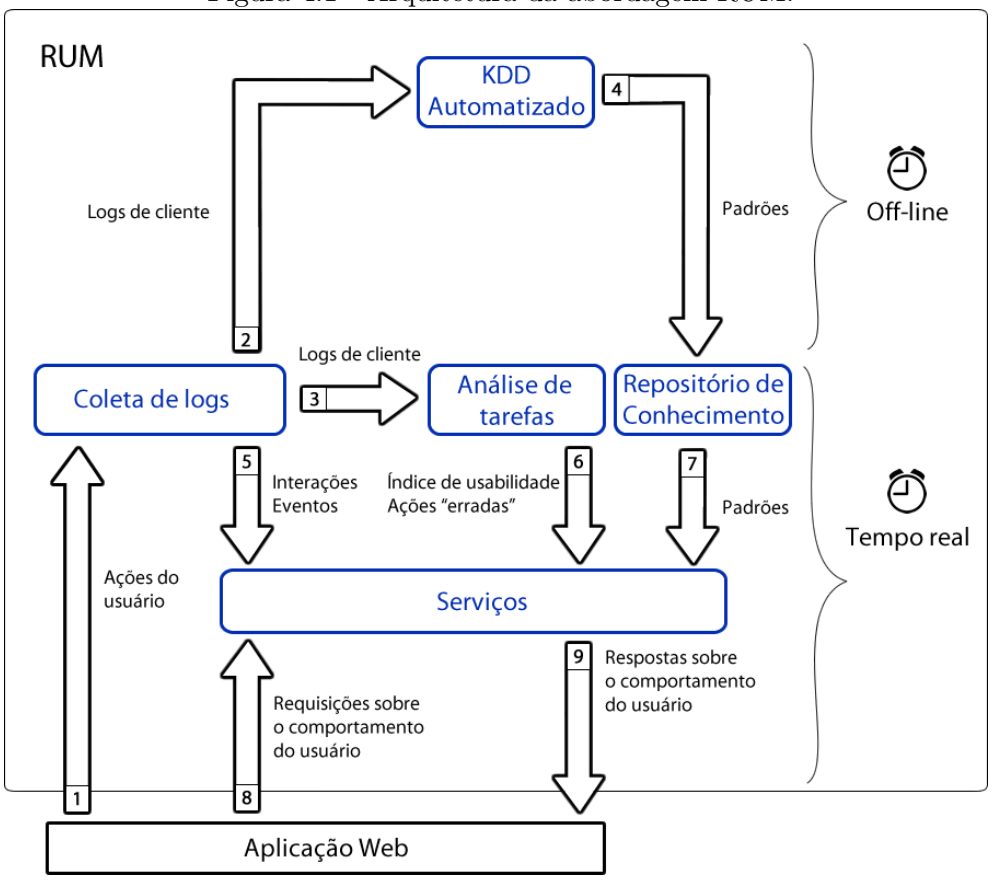
c) **KDD automatizado:** do termo em inglês *Knowledge Discovery in Databases* (descoberta de conhecimento em bases de dados). Detecta padrões de comportamento a partir do histórico de navegação dos usuários da aplicação Web;

d) **Repositório de conhecimento:** armazena e gerencia os padrões de comportamento detectados, considerando parâmetros fornecidos pelo especialista da aplicação Web;

e) **Serviços:** recebe requisições da aplicação Web e fornece informações do comportamento do usuário durante a navegação.

[1] Um *toolkit* é um conjunto de bibliotecas e ferramentas cujas funcionalidades podem ser consumidas por um *software*.

Figura 4.1 - Arquitetura da abordagem RUM.



Na Figura 4.1, as setas indicam o fluxo de dados entre os módulos e a numeração indica a sequência desse fluxo. Inicialmente, como indicado pelo fluxo 1, o módulo de coleta de *logs* detecta as ações dos usuários (eventos) na aplicação Web, tratando as particularidades de cada tipo de dispositivo (computador, *smartphone*, *tablet*, etc.). Após isso, como indicam os fluxos 2 e 3, as ações detectadas são transformadas e armazenadas em *logs* para serem analisados pelos módulos de Análise de Tarefas e de KDD automatizado.

Em seguida, o módulo de análise de tarefas realiza a avaliação remota e automática de usabilidade da aplicação Web durante a navegação, usando tarefas previamente definidas pelo especialista da aplicação. Periodicamente, como indicado pelo fluxo 2, o módulo de descoberta de conhecimento automatizada consome os *logs* gerados pelo módulo de coleta e executa o processo de Descoberta de Conhecimento em Bases de Dados (da sigla em inglês, KDD), detectando padrões de comportamento a partir do histórico dos usuários.

Os padrões de comportamento detectados alimentam o módulo de Repositório de Conhecimento (fluxo 4), que define a relevância de cada padrão, adequando o repositório às mudanças de comportamento dos usuários com o passar do tempo. À medida que o usuário interage com a aplicação Web, seu comportamento irá se especializar e, geralmente, suas ações serão mais rápidas ao executar tarefas conhecidas. Semelhantemente, com o passar do tempo, a aplicação Web se torna mais conhecida para os usuários frequentes, portanto alguns padrões de comportamento que eram detectados quando a aplicação era desconhecida podem não ocorrer mais.

Enquanto o usuário navega, através do módulo de serviços, a aplicação Web requisita informações sobre o comportamento do usuário (fluxo 8) para disparar adaptações pré-programadas na interface. As respostas (fluxo 9) das requisições são: sequências das últimas ações do usuário (fluxo 5), o resultado da avaliação de usabilidade durante a navegação (fluxo 6), e os padrões de comportamento executados pelo usuário ativo (fluxo 7).

As Seções 4.1.1 a 4.1.4 detalham o funcionamento, as vantagens e as limitações de cada módulo da RUM.

4.1.1 Coleta de *logs*

Diferentemente da maioria dos trabalhos reportados na literatura, a abordagem RUM explora *logs* de cliente e, através do módulo de Coleta, são capturadas todas as ações do usuário desde o momento em que ele acessa a aplicação até a página em que decide abandoná-la.

As ações realizadas por um usuário durante a interação com a aplicação Web podem ser representadas por uma lista encadeada, em que cada elemento representa uma ação do usuário. Neste sentido, supondo que algumas ações são executadas repetidamente, pode-se transformar essa lista encadeada em um grafo orientado, em que os vértices representam as ações dos usuários e as arestas representam a relação de precedência entre as ações. Portanto, cada sessão de um usuário, chamada de interação, é representada por um grafo orientado das ações detectadas na interface da aplicação Web.

Uma vez que as ações dos usuários são coletadas e transformadas em *logs*, o módulo de coleta organiza-as em um grafo orientado, que possui dois tipos de vértices: (i) vértice de interação e (ii) vértice de ação do usuário. O vértice de interação representa a sessão do usuário na aplicação Web e, portanto, contém informações gerais sobre a

interação, que são: o tempo inicial, o tempo final, as informações sobre o navegador e o sistema operacional utilizados. Os vértices de ações dos usuários representam as ações de *mouse*, barra de rolagem, etc.

A coleta de *logs* de cliente na RUM tem dois propósitos: (i) apoiar a avaliação remota e automática de usabilidade durante a navegação; e (ii) apoiar a descoberta de padrões de comportamento dos usuários. A Seção 4.1.2 descreve o módulo de análise de tarefas que executa a avaliação de usabilidade e a Seção 4.1.3 explica o processo automatizado de descoberta de conhecimento nos *logs* coletados.

4.1.2 Análise de tarefas

A literatura reporta os trabalhos de Paganelli e Paternò (2002) e Vargas et al. (2010) que se baseiam na análise de tarefas para fins de avaliação de usabilidade. Semelhantemente, em trabalhos anteriores, USABILICS (VASCONCELOS; BALDOCHI, 2011) e MOBILICS (GONCALVES et al., 2016), foi proposto um método para avaliação remota e automática de usabilidade baseada na análise de tarefas. O módulo de análise de tarefas da RUM estende o método usado nas ferramentas USABILICS e MOBILICS. Inicialmente, através da ferramenta USABILICS, era avaliada a usabilidade de aplicações Web apenas em interfaces para computadores. Com o desenvolvimento do MOBILICS, essa capacidade de avaliação foi estendida aos dispositivos móveis. Dentro da abordagem desta pesquisa, há a capacidade de avaliar a usabilidade da aplicação Web durante a navegação, enquanto naquelas ferramentas realizava-se a avaliação apenas após a interação do usuário.

O propósito de se avaliar a usabilidade é prover à aplicação Web informações sobre possíveis dificuldades que o usuário enfrenta enquanto navega, de forma que possam ser disparadas adaptações na interface que o auxiliem na interação.

A Seção 4.1.2.1 descreve brevemente o método de avaliação de usabilidade herdado das ferramentas USABILICS e MOBILICS.

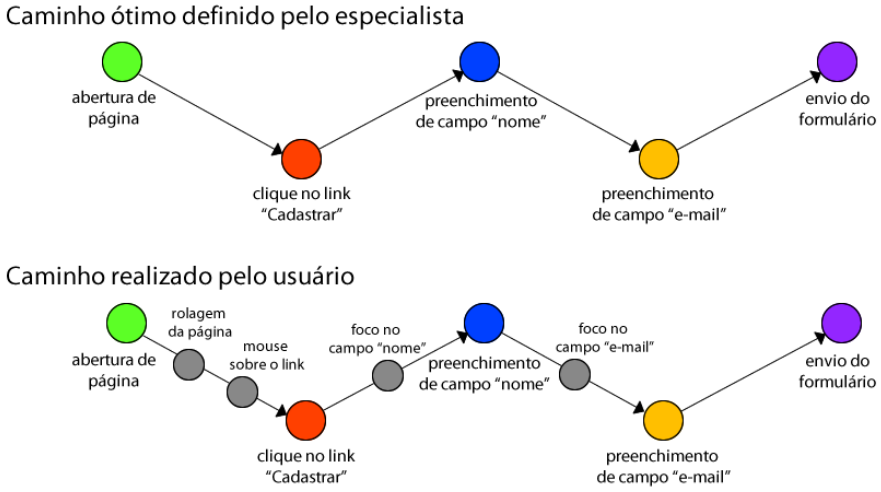
4.1.2.1 Avaliação de usabilidade baseada na análise de tarefas

A avaliação de usabilidade baseada na análise de tarefas parte do princípio de que há um conjunto de tarefas previamente definido para comparação com os caminhos realizados pelo usuário na interface. No método usado pela RUM, a construção do conjunto de tarefas é feita pelo próprio especialista da aplicação Web diretamente na interface. Portanto, o especialista grava o "caminho ótimo" da tarefa.

Uma vez que o caminho ótimo é definido, a usabilidade pode ser avaliada pela comparação entre o caminho realizado pelo usuário e o caminho ótimo. Essa comparação resulta em uma medida de similaridade chamada índice de usabilidade, que representa a eficácia e a eficiência da interface para cada interação de um usuário. A eficácia é a taxa de completude da tarefa, isto é, quantos passos da tarefas foram completados; e a eficiência é mensurada pela similaridade entre as ações realizadas e as ações do caminho ótimo.

A Figura 4.2 ilustra a comparação entre o caminho realizado pelo usuário e o caminho ótimo de uma tarefa hipotética. A tarefa ilustrada refere-se ao cadastro de um visitante em um *web site*. Observa-se que o caminho ótimo contém as ações minimamente necessárias para se completar a tarefa, isto é, são as ações que representam decisões do usuário, tais como: acessar uma nova página, preencher um campo de formulário e clicar em um *link*. Na Figura 4.2, as ações do caminho ótimo são identificadas pelos círculos coloridos. Obviamente, o usuário não realiza apenas as ações do caminho ótimo, pois necessita movimentar o *mouse* ou ainda rolar a tela para encontrar um determinado elemento da interface para interagir. Naquela figura, as ações que o usuário faz que não pertencem ao caminho ótimo são identificadas por círculos cinzas. Nesse contexto, o método de avaliação de usabilidade herdado pela abordagem RUM mensura quão distante o caminho realizado pelo usuário está do caminho ótimo.

Figura 4.2 - Comparação entre o caminho ótimo e o caminho realizado pelo usuário para avaliação de usabilidade.



No entanto, em aplicações Web modernas, há diversos caminhos para se realizar uma tarefa. Dessa forma, o método utiliza um modelo de interface que generaliza um caminho ótimo para abranger caminhos semelhantes. Com o modelo de interface, chamado COP, que é abreviação dos termos em inglês: *Container*, *Object* e *Page*, a interação do usuário não é penalizada apenas porque ele decidiu navegar por outro caminho que não seja exatamente o caminho gravado pelo especialista. Mesmo com caminhos distintos, é possível definir, no caminho ótimo, ações opcionais, conjuntos cíclicos de ações e ações que não sejam sequenciais, a fim de reduzir o esforço do especialista para gravação de tarefas e abranger caminhos alternativos escolhidos pelos usuários.

A limitação do método está relacionada à generalização do modelo de interface COP. Por exemplo, se uma tarefa possui caminhos que percorrem páginas de estruturas de interface completamente distintas, a generalização não se torna possível, sendo necessário gravar diferentes "caminhos ótimos" para a tarefa.

Além do índice de usabilidade gerado na comparação com o caminho ótimo, o método indica possíveis problemas de usabilidade pontuais na aplicação Web. Por meio da medida de similaridade entre cada ação do usuário e cada ação do caminho ótimo, são detectadas as chamadas ações erradas (em inglês, *wrong actions*), termo definido por Vermeeren et al. (2008). Por exemplo, se o caminho ótimo indica que é esperado o clique em um determinado *link* de uma página A e o usuário clica em outro *link* posicionado próximo àquele na mesma página A, essa ação é mais similar à ação do caminho ótimo do que o clique em um *link* de uma página B. Então, a medida de similaridade considera como ações erradas aquelas cuja similaridade é baixa. O cálculo da similaridade é detalhadamente explicado em Vasconcelos e Baldochi (2011).

É importante explicar que o termo "ações erradas" do ponto de vista de interação humano-computador não considera que o usuário cometeu erros em sua interação, mas sim que a interface tem problemas de usabilidade que levam o usuário a desviar-se do caminho ótimo esperado pelo especialista.

Na abordagem RUM, como o propósito é apoiar a construção de aplicações Web adaptativas, o módulo de análise de tarefas executa a avaliação de usabilidade durante a navegação. Para isso, o módulo é alimentado pelas tarefas definidas pelo especialista da aplicação. Enquanto o usuário navega, um algoritmo compara as ações do usuário com o caminho ótimo de cada tarefa, a fim de detectar o início e o fim de uma tarefa. Com isso, a RUM fornece à aplicação Web as seguintes infor-

mações: (i) a tarefa que está sendo executada; (ii) o índice de usabilidade de cada tarefa previamente definida que esteja em execução ou que já tenho sido executada; e (iii) as ações erradas detectadas em cada tarefa. Essas informações provêm à aplicação Web a capacidade de identificar durante a navegação os usuários que estejam enfrentando algum problema de usabilidade, a fim de disparar uma adaptação na interface. O algoritmo e a implementação da análise de tarefas são detalhados na Seção 4.2.2.

Em aplicações Web específicas como, por exemplo, portais de notícias e sites informativos, nem sempre é viável definir tarefas devido à inexistência de caminhos específicos e devido à pequena quantidade de ações necessárias para se realizar uma tarefa. Em um portal de notícias, há diversos caminhos para se ler uma notícia e, muitas vezes, essa é uma tarefa de apenas um clique. Considerando esse contexto, a RUM contempla outro tipo de análise de *logs*, que é a descoberta de padrões do comportamento do usuário independentemente da definição de tarefas. Para isso, foi projetado o módulo de Descoberta de Conhecimento (KDD) em *Logs* em conjunto com o módulo de Repositório de Conhecimento. O funcionamento de ambos os módulos é descrito na Seção 4.1.3.

4.1.3 Processo de descoberta do conhecimento em *logs* da Web

Analogamente ao processo de descoberta do conhecimento em bases de dados fundamentado no Capítulo 3, a RUM contempla especificamente a descoberta de conhecimento em *logs* da Web, fornecidos pelo módulo de coleta.

Para alcançar o propósito da abordagem RUM, o processo de KDD explora os *logs* de cliente e extrai padrões relevantes sobre o comportamento dos usuários. Portanto, dentre os conceitos expostos na Seção 3.3 sobre Mineração de Dados da Web, a RUM concentra-se na Mineração de Uso da Web (*Web Usage Mining*).

O objetivo do processo de KDD da RUM nos *logs* de cliente é caracterizar o usuário. A motivação para isso é que cada aplicação Web possui diferentes usuários com interesses e objetivos específicos. Para isso, os *logs* de cliente são explorados para a detecção de padrões sequenciais e regras de associação a partir de atributos que caracterizam o comportamento dos usuários.

As etapas do processo de KDD da RUM são descritas a seguir.

a) **Seleção**

- Seleciona os atributos que caracterizam o comportamento do usuário.

  Para a seleção dos atributos, foram observadas, em diferentes aplicações Web, as ações do usuário que indicam uma decisão, as ações que antecedem uma decisão e a estrutura das páginas Web.

  Com o crescimento do acesso a aplicações Web através de dispositivos móveis, os usuários interagem, na maior parte do tempo, rolando a tela verticalmente e acessando *links* que habilitam funcionalidades ou levam-nos a outras páginas. Durante a interação, diferentes usuários realizam tarefas em diferentes intervalos de tempo. Além disso, a agilidade com os dispositivos sensíveis ao toque também deve ser observada. Por exemplo, usuários inexperientes nesses dispositivos tendem a operá-los mais lentamente, enquanto usuários que usam esses dispositivos com frequência realizam mais ações em curto espaço de tempo. Além da experiência com dispositivos móveis, a experiência com a própria aplicação Web também interfere nas decisões e no tempo de navegação do usuário. Observa-se ainda que usuários impacientes costumam realizar ações mais rapidamente e, consequentemente, acessam mais páginas.

  O resultado dessa análise foi a seleção de oito atributos que podem ser recuperados dos *logs*: tempo da interação, quantidade de cliques (toques), quantidade de ações de rolagem, intervalo médio de tempo entre as ações de rolagem, área máxima da página percorrida pelo usuário, *links* clicados (tocados), páginas visitadas e quantidade de outras ações realizadas.

  A partir desses atributos, são produzidos quatro conjuntos de dados:

  - **Conjunto "ClickedLinks":** contém atributos booleanos que representam os links clicados em cada interação;
  - **Conjunto "ProfileRules":** contém atributos nominais que representam classes de valores para cada um dos atributos selecionados na etapa de Seleção: duração da interação, quantidade de cliques, quantidade de ações de rolagem, intervalo médio de tempo entre as ações de rolagem, quantidade de ações realizadas (eventos), área máxima da tela visualizada e quantidade de páginas visitadas;
  - **Conjunto "ProfileRulesByPage":** combina os atributos dos conjuntos *ClickedLinks* e *ProfileRules* para reduzir a granularidade na busca por padrões sequenciais, a fim de encontrar sequên-

cias mais específicas entre páginas;

- **Conjunto "TimeByPage":** contém atributos nominais que marcam os links clicados em cada interação e o tempo de interação entre os cliques.

A fim de encontrar padrões sequenciais nos conjuntos *ClickedLinks*, *ProfileRulesByPage* e *TimeByPage* e extrair a relação entre os atributos do conjunto *ProfileRules*, a técnica de regras de associação foi escolhida. Nos três primeiros conjuntos, a existência do modelo transacional dos dados, em que há uma relação de precedência entre as instâncias, direciona ao uso dessa técnica. O motivo da escolha da técnica para o conjunto *ProfileRules* é a capacidade dos algoritmos dessa técnica detectarem padrões associativos entre atributos das instâncias.

b) **Pré-processamento:**

- Nesta etapa automatizada do KDD da RUM, os atributos são extraídos dos *logs* e possíveis ruídos da coleta das ações dos usuários são eliminados.

c) **Transformação:**

- Esta etapa também é automatizada e prepara os *logs* para serem minerados, transformando os conjuntos de atributos selecionados em atributos nominais e binários para identificação de padrões sequenciais e regras de associação.

d) **Mineração:**

- Nesta etapa automatizada, são executados os algoritmos de mineração sobre os dados transformados, de acordo com o objetivo de cada conjunto de atributos selecionados.

e) **Interpretação e avaliação:**

- Na última etapa, o especialista define perfis de usuários conhecidos da aplicação Web. Em seguida, ele avalia os padrões detectados atribuindo-os aos perfis que definiu, ou descartando os padrões que julga irrelevantes. Além disso, o especialista pode atribuir o nome de uma ação a um ou mais padrões, de forma que a ação seja disparada quando um usuário realizar esses padrões na interação. É importante

salientar que essas ações são implementadas pelo desenvolvedor da aplicação, cabendo à RUM informar a aplicação Web sobre a ocorrência de uma ação definida pelo especialista. Esse funcionamento torna a abordagem adaptável à qualquer aplicação Web, flexibilizando as adaptações necessárias a cada tipo de aplicação.

O processo de KDD em *logs* é executado periodicamente e alimenta o módulo de Repositório de Conhecimento da RUM com os padrões detectados em cada execução. Assim, o Repositório de Conhecimento gerencia os padrões através de um procedimento incremental. Quando um padrão é detectado, o módulo compara-o com os padrões já conhecidos. Se o padrão já existe no repositório, então sua prioridade aumenta. Se o padrão detectado equivale a um padrão descartado anteriormente pelo especialista, então esse padrão também será descartado. O objetivo de manter a prioridade dos padrões cumulativamente é adequar o Repositório de Conhecimento às mudanças de comportamento dos usuários com o passar do tempo. Assim, os padrões que deixam de ser executados pelos usuários passam a ser *esquecidos* pelo repositório.

Uma vez que os padrões são armazenados no módulo de Repositório de Conhecimento, esses padrões são comparados durante a navegação com as ações do usuário ativo, no intuito de informar a aplicação sobre a execução de padrões selecionados pelo especialista.

Completando a arquitetura da RUM, a Seção 4.1.4 descreve o funcionamento durante a navegação do usuário e o papel do módulo de Serviços.

4.1.4 Abordagem durante a navegação

A principal vantagem da abordagem proposta nesta tese é a capacidade de analisar o comportamento do usuário durante a navegação, a fim de apoiar adaptações na aplicação. Nesse contexto, o módulo de Serviços estabelece a comunicação entre a aplicação Web e as informações resultantes do processamento dos outros módulos. O módulo de Serviços funciona em conjunto com os módulos de Coleta de *Logs*, de Análise de Tarefas e de Repositório de Conhecimento. Dessa forma, o módulo de Serviços recebe requisições da aplicação Web sobre (i) as ações executadas pelo usuário ativo; (ii) o índice de usabilidade e as ações erradas de uma determinada tarefa definida na aplicação; e (iii) os padrões executados pelo usuário ativo. Com essas requisições, a aplicação Web obtém informações do comportamento do usuário e dispara adaptações pertinentes a cada perfil de usuário.

A fim de tornar a abordagem RUM implantável em aplicações Web, foi implementado um *toolkit*, chamado TAWS (em inglês, *Toolkit for Adaptive Web Sites*), cujos detalhes de implementação são especificados na Seção 4.2.

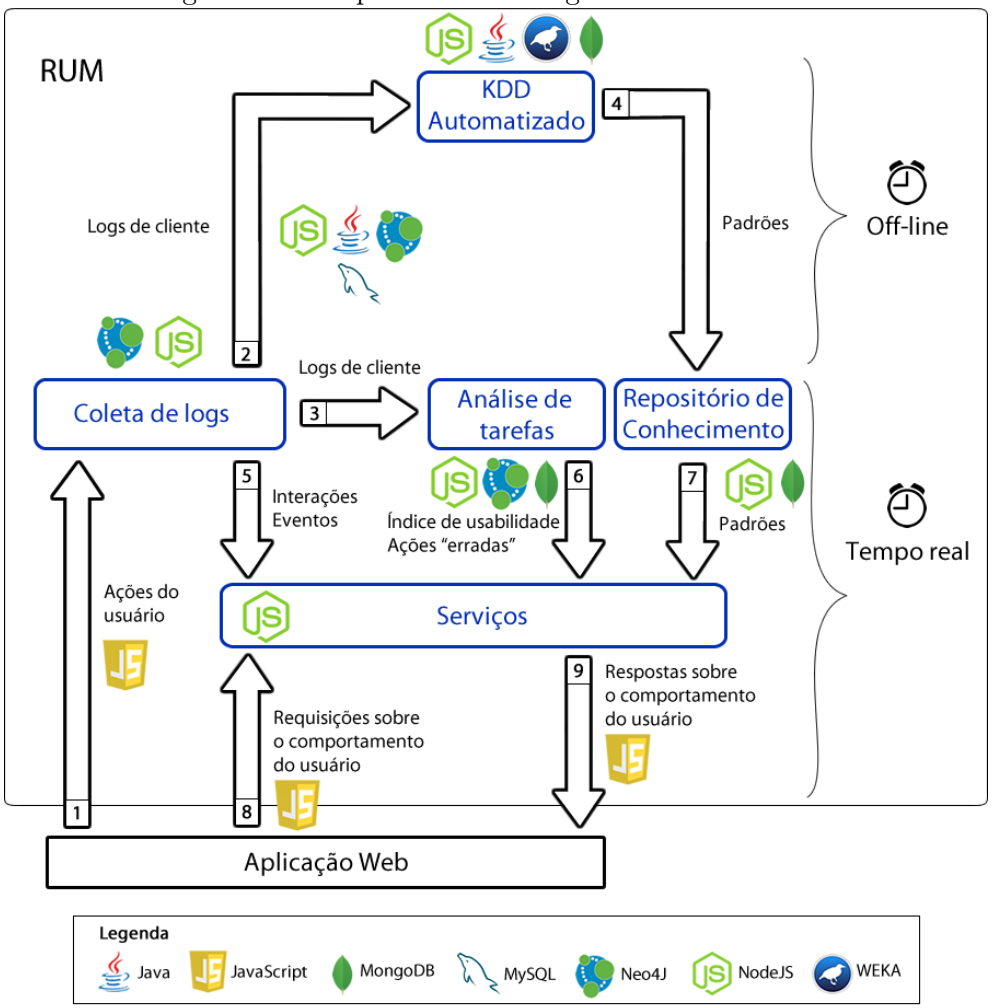
4.2 Implementação do *toolkit* TAWS

Diante da dificuldade em construir aplicações Web adaptativas, uma importante contribuição desta tese é um *toolkit* que implementa a abordagem RUM, chamado TAWS. O TAWS encapsula a coleta de *logs*, o processamento e a análise do comportamento do usuário, reduzindo o esforço de implementação para o desenvolvedor da aplicação. Uma característica importante do TAWS é a facilidade de implantação em qualquer aplicação Web, independentemente das tecnologias utilizadas na etapa de desenvolvimento de software, pois baseia-se nas tecnologias-padrão de desenvolvimento Web.

Para prover escalabilidade desde pequenas a grandes aplicações Web, o TAWS baseia-se em arquitetura distribuída, e sua implementação inclui banco de dados relacional e não-relacional, bibliotecas de algoritmos para mineração de dados e estratégias para otimizar a coleta e o processamento dos *logs* de cliente.

A Figura 4.3 mostra a arquitetura tecnológica do TAWS, destacando as tecnologias utilizadas em cada módulo. Essencialmente, o TAWS utiliza a tecnologia NodeJS (NODEJS.ORG, 2016) nos programas implementados no servidor e a linguagem JavaScript para a coleta e as requisições da aplicação Web. A plataforma Java é usada no módulo de KDD para integração com a biblioteca do software WEKA (WAIKATO, 2013) e a definição de tarefas é feita em uma aplicação Web implementada em Java com banco de dados relacional MySQL.

Figura 4.3 - Arquitetura tecnológica do *toolkit* TAWS.



Na arquitetura, são exploradas as estruturas de dados de diferentes tecnologias de banco de dados. Para o armazenamento e manipulação dos *logs*, é usado um banco de dados NoSQL específico para grafos, o Neo4J (NEO4J.COM, 2016). Os resultados da análise de tarefas e do processo automatizado de KDD são armazenados em um banco de dados de documentos, o MongoDB (MONGODB.COM, 2016).

As Seções 4.2.1 a 4.2.5 descrevem a implementação de cada módulo da RUM no *toolkit*.

4.2.1 Módulo de coleta de *logs*

Um dos aspectos que devem ser considerados para a construção de um mecanismo de coleta eficiente é a capacidade de coletar os dados sem onerar a performance

da aplicação em que foi implantado. Semelhantemente a Azzopardi et al. (2012), o TAWS possui um mecanismo de coleta eficiente.

O mecanismo de coleta é dividido em dois programas: (i) o programa cliente, responsável por detectar as ações do usuário no navegador e enviá-las ao (ii) programa servidor, que recebe os dados, processa-os e armazena-os em um banco de dados. No TAWS, ambos os programas foram reimplementados em relação ao USABILICS para reduzir a latência do mecanismo de coleta.

A cada grupo de ações enviado ao servidor, cujo tamanho é definido arbitrariamente no programa cliente, o programa servidor processa as ações recebidas e amplia o grafo, mantendo a relação de precedência entre as ações.

Para o armazenamento do grafo, são usadas duas instâncias do banco de dados Neo4J: uma instância para armazenar as interações dos usuários ativos (chamada instância *real-time*) e outra para armazenar o histórico das interações dos usuários que já abandonaram a aplicação Web (chamada instância *off-line*). Periodicamente, o programa servidor identifica as interações finalizadas na instância *real-time* e move-as para a instância *off-line*.

Esse recurso é usado para reduzir o volume de dados da instância *real-time*, para prover a performance adequada ao módulo de Serviços da RUM, que recebe requisições da aplicação Web sobre as últimas ações do usuário ativo e recupera-as da instância *real-time*. Já os *logs* armazenados na instância *off-line* são usados periodicamente pelo módulo de KDD para a extração de padrões.

Para detalhar os dados coletados sobre as ações dos usuários, a Tabela 4.1 resume as principais propriedades capturadas pelo mecanismo de coleta do TAWS.

A coleta dessas propriedades possibilita a derivação de muitas informações sobre o comportamento do usuário. Por exemplo, usando a propriedade *timestamp*, pode-se calcular o intervalo de tempo entre as ações de barra de rolagem; combinando as propriedades *objectValue* e *objectTagName*, são detectados os *links* que o usuário visitou; e ainda pode-se identificar o caminho de um usuário na aplicação usando a propriedade *currentURL*, que identifica a página visitada pelo usuário.

Na análise de tarefas para fins de avaliação de usabilidade, todas as propriedades são combinadas com o modelo de interface COP, tanto para definição de tarefas pelo especialista quanto para o cálculo da similaridade em relação ao caminho ótimo. A Seção 4.2.2 descreve a implementação do módulo de Análise de Tarefas da RUM.

Tabela 4.1 - Propriedades capturadas nas ações dos usuários.

Propriedade	**Descrição**
type	Tipo da ação: *click*, *keypress*, *load*, *openpage*, etc.
timestamp	Data/hora de ocorrência da ação no navegador
currentURL	URL da página em que a ação ocorreu
title	Título da página em que a ação ocorreu
offSetScrollY e **offSetScrollX**	Posição da barra de rolagem no momento da ação
objectTagName	Tag HTML do elemento em que a ação ocorreu
objectName	Nome do objeto em que a ação ocorreu
objectId	ID do objeto em que a ação ocorreu
objectValue	Conteúdo do objeto em que a ação ocorreu.

Além disso, a Seção destaca o algoritmo implementado no TAWS para a análise de tarefas durante a navegação.

4.2.2 Módulo de análise de tarefas

O *toolkit* reutiliza o módulo de definição de tarefas das ferramentas USABILICS e MOBILICS, denominado UsaTasker (VASCONCELOS; BALDOCHI JR., 2012).

O método usado por aquelas ferramentas é eficiente para descobrir e corrigir problemas de usabilidade. Depois da avaliação, um baixo índice de usabilidade e as ações erradas que causaram os problemas detectados ajudam o desenvolvedor a corrigir a aplicação.

Essas informações são valiosas para apoiar aplicações Web adaptativas. Por exemplo, o índice de usabilidade pode ser usado para detectar usuários novatos, isto é, usuários que não estão familiarizados com a aplicação. Esses usuários tendem a apresentar um comportamento exploratório movendo o *mouse* e rolando a página mais vezes que o usual para usuários recorrentes. Quando um usuário novato é detectado, a aplicação pode, por exemplo, adaptar sua interface para ajudá-lo.

Para realizar a análise de tarefas e fornecer informações sobre a usabilidade durante a interação, um algoritmo foi proposto (VASCONCELOS et al., 2014) e implementado no TAWS (Algoritmo 1). O algoritmo calcula iterativamente o índice de usabilidade apontando as ações erradas assim que são detectadas durante a navegação na aplicação Web.

Para compreender o algoritmo, inicialmente faz-se necessária a descrição das variá-

veis utilizadas:

- ***currentTask*** tarefa que o usuário está realizando
- ***lastAccomplishedEvent*** último evento do caminho ótimo alcançado pelo usuário
- ***T*** lista de tarefas definidas pelo especialista
- ***usabilityIndex*** índice de usabilidade de cada tarefa iniciada pelo usuário
- ***listWrongActions*** lista de ações erradas detectadas para cada tarefa iniciada pelo usuário

O algoritmo considera que o primeiro evento do caminho ótimo de uma determinada tarefa é único, isto é, ele não inicia o caminho ótimo de outras tarefas. Portanto, quando uma tarefa está sendo realizada e o usuário executa um evento que corresponde ao primeiro evento de outra tarefa, a tarefa atual é terminada e a análise de uma nova tarefa inicia.

A entrada do algoritmo são os eventos capturados pelo mecanismo de coleta enquanto o usuário navega na aplicação. Portanto, o algoritmo inicia-se a cada nova sessão da aplicação Web.

O algoritmo é composto por três partes principais: (i) o programa principal, que executa sempre que uma nova sessão é iniciada; (ii) o procedimento *AnalyzeEvent*, que analisa os eventos capturados; e (iii) o procedimento *SaveResultsOfTask*, que armazena o índice de usabilidade e as ações erradas detectadas durante a execução da tarefa.

Quando uma nova sessão é iniciada, as variáveis são ajustadas para os valores-padrão (linhas 1 a 4). A coleta dos eventos é então iniciada pelo mecanismo de coleta (linha 5). O procedimento *AnalyzeEvent* é chamado quando um novo evento *e'* é capturado (linhas 6 e 7). Esse procedimento verifica se o novo evento pertence à tarefa em execução ou se é o primeiro evento de uma nova tarefa. No primeiro caso, o procedimento calcula a similaridade entre esse evento e o evento esperado no caminho ótimo da tarefa (linha 13 do procedimento *AnalyzeEvent*).

No procedimento *AnalyzeEvent*, se o evento *e'* corresponde ao primeiro evento de uma tarefa e há uma tarefa em execução (*currentTask != null* na linha 3), então

Algorithm 1 Análise de tarefas durante a navegação.

Inicialização: Quando uma nova sessão é iniciada na aplicação Web

Input: os eventos capturados pelo módulo de Coleta

Output: o índice de usabilidade e as ações erradas de cada tarefa iniciada pelo usuário

1: $usabilityIndex \leftarrow 0$
2: $listWrongActions \leftarrow \emptyset$
3: $currentTask \leftarrow null$
4: $lastAccomplishedEvent \leftarrow null$
5: Inicializa a captura de eventos
6: **for** cada evento e' capturado pelo módulo de Coleta **do**
7: AnalyzeEvent(e')
8: **end for**
=0

Procedure AnalyzeEvent(*e'*)

1: **for** cada t' em T **do**
2: **if** e' é o evento inicial do caminho ótimo de t' **then**
3: **if** currentTask != null **then**
4: SaveResultsOfTask(currentTask)
5: Abortar currentTask
6: **end if**
7: $currentTask \leftarrow t'$
8: $lastAccomplishedEvent \leftarrow e'$
9: break
10: **end if**
11: **end for**
12: **if** currentTask != null **then**
13: Calcula a medida de similaridade entre **e'** e o próximo evento do caminho ótimo de **currentTask**
14: Atualiza **usabilityIndex** com o valor da medida de similaridade
15: **if** e' é o próximo evento do caminho ótimo **then**
16: $lastAccomplishedEvent \leftarrow e'$
17: **if** e' é o evento final do caminho ótimo **then**
18: SaveResultsOfTask(currentTask)
19: Finalizar currentTask
20: **end if**
21: **else**
22: **if** e' é uma ação errada **then**
23: Acrescentar e' em listWrongActions
24: **end if**
25: **end if**
26: **end if**
=0

Procedure SaveResultsOfTask(*currentTask*)

1: Salvar *usabilityIndex* de *currentTask* na interação do usuário
2: Salvar *listWrongActions* relacionadas à tarefa *currentTask* na interação
3: $usabilityIndex \leftarrow 0$
4: $listWrongActions \leftarrow \emptyset$
5: $lastAccomplishedEvent \leftarrow null$
6: $currentTask \leftarrow null$ =0

é necessário finalizá-la, pois foi abortada pelo usuário (linha 5). Neste caso, é importante salvar o comportamento do usuário enquanto ele está tentando realizar a tarefa, então o procedimento *SaveResultsOfTask* é chamado na linha 4 antes de abortar a tarefa na linha 5.

Na linha 13, o evento *e'* é comparado ao próximo evento do caminho ótimo da tarefa. Esta comparação resulta em uma medida de similaridade entre 0 e 1. Então, o índice de usabilidade é atualizado com a medida, na linha 14. Se *e'* corresponde ao evento esperado no caminho ótimo, significa que o usuário completou um passo da tarefa, então *lastAccomplishedEvent = e'* (linha 16). De outra maneira, há o caso em que *e'* é uma ação errada (linha 22). Neste caso, o evento deve ser adicionado à lista de ações erradas (linha 23). Se *e'* é o último evento da tarefa, então o procedimento *SaveResultsOfTask* é chamado para armazenar o índice de usabilidade e as ações erradas detectadas (linha 18).

Para execução do algoritmo, os mecanismos de Coleta de *Logs* e de Análise de Tarefas funcionam de maneira integrada. A implementação do algoritmo de análise de tarefas consome os *logs* da instância *real-time* do banco de dados de grafos, apresentada na seção anterior. O índice de usabilidade e as ações erradas detectadas em cada interação são armazenados em uma instância do banco de dados MongoDB, que é consumida pelo módulo de Serviços da RUM para atender as requisições da aplicação Web.

Nesta tese, são relatados os estudos de caso que foram conduzidos para validar o algoritmo e demonstrar seu funcionamento em aplicações Web reais, e são descritos nas Seções 5.1 e 5.2.

Além da análise de tarefas, o TAWS automatiza o processo de KDD para extração de padrões nos *logs* de comportamento dos usuários, como detalha a Seção 4.2.3.

4.2.3 Módulo de KDD automatizado

A entrada de dados da implementação deste módulo são os *logs* dos usuários que já encerraram a interação. Diariamente, um programa recupera os *logs* das interações do dia anterior e realiza o pré-processamento e a transformação dos dados para a execução da técnica de mineração.

Após a transformação dos dados, um programa executa um algoritmo de mineração de dados implementado na biblioteca do software WEKA e extrai os padrões. O algoritmo selecionado para a extração das regras de associação nos conjuntos é o algoritmo *Predictive Apriori* (SCHEFFER, 2001), que é um dos algoritmos da família Apriori. Este algoritmo usa uma medida chamada precisão preditiva (em inglês, *predictive accuracy*), a fim de maximizar a probabilidade de uma predição correta de dados ainda não analisados nas iterações do algoritmo. Essa medida auxilia a geração das n principais regras geradas, ou seja, as regras mais precisas. Além disso, o algoritmo *Predictive Apriori* elimina regras redundantes dos resultados e extrai regras de associação fortes entre os itens do conjunto de dados que seriam descartadas pelo algoritmo Apriori.

Os padrões extraídos do processo de KDD são enviados ao módulo de Repositório de Conhecimento da RUM, cuja implementação é descrita na Seção 4.2.4.

4.2.4 Módulo de repositório de conhecimento

Este módulo é responsável por armazenar de forma estruturada os padrões extraídos do KDD automatizado e gerenciar o conhecimento sobre o comportamento do usuário, com a participação do especialista da aplicação Web.

Cada padrão recebido do módulo de KDD automatizado é procurado em um repositório de padrões. Se não for encontrado, o padrão é inserido no repositório com status “em avaliação”. Se o padrão for encontrado, há quatro classificações possíveis:

a) **Padrão não avaliado:** é um padrão adicionado ao repositório recentemente ou um padrão antigo que ainda está em avaliação pelo especialista;

b) **Padrão descartado:** quando o especialista da aplicação considera um padrão como irrelevante, este é descartado. Apesar de todo padrão extraído do processo de KDD ser relevante dentro do conjunto de dados, há padrões óbvios para o especialista, por exemplo;

c) **Padrão vinculado a um perfil de usuário:** conhecendo os interesses

dos usuários da aplicação Web, o especialista reconhece um padrão como característico de um perfil específico, então associa o padrão a esse perfil;

d) **Padrão vinculado a uma ação para adaptação:** quando o especialista julga um padrão como relevante mas não compreende que é característico de um perfil de usuário, ele pode associar o padrão a uma ação a ser disparada. Por exemplo, o especialista pode abstrair um padrão sequencial para um objetivo ou comportamento do usuário em alto nível, classificando-o semanticamente;

No repositório, cada padrão é representado por um objeto da notação JSON. A seguir, são apresentados dois exemplos de padrões armazenados. O primeiro é um exemplo de padrão detectado no conjunto "‘ClickedLinks"’, e o segundo é um padrão do conjunto "‘ProfileRules"’.

```
[{
  type: CLICKED_LINK,
  items: [
    'Menu', 'Category', 'Product A'
  ],
  frequency: 0,
  status: EVALUATING
},
{
  type: PROFILE_RULE,
  items: [
    {
      name: SCROLL_INTERVAL,
      value: LESS_200MS
    },
    {
      name: CLICKS,
      value: 3_TO_5
    },
    {
      name: VISITED_PAGES,
      value: 1_TO_2
    }
  ],
  frequency: 0,
  status: EVALUATING
}]
```

Para a avaliação dos padrões no TAWS, uma aplicação Web consome os dados desse repositório e exibe-os ao especialista, que pode classificar cada padrão. Nessa aplicação Web, o especialista cria perfis de usuário e ações para adaptação, conforme sua interpretação do modelo de negócios da aplicação Web que está sendo analisada.

Os padrões associados pelo especialista a perfis de usuários ou a ações para adaptação são armazenados em um banco de dados específico. Esse banco de dados é consumido pelo módulo de Serviços para comparar, durante a navegação, as ações do usuário ativo com os padrões classificados pelo especialista.

Todo padrão no repositório é priorizado por sua frequência, que é um contador de sua ocorrência. Através de sua frequência, o repositório aprende o comportamento dos usuários e, com o passar do tempo, descarta padrões que deixaram de ser executados.

A Seção 4.2.5 detalha a implementação do módulo de Serviços no TAWS, que viabiliza o consumo das informações da abordagem RUM por parte da aplicação Web.

4.2.5 Módulo de serviços

No TAWS, este módulo da RUM foi implementado baseado na arquitetura de serviços Web (em inglês, *Web services*), estabelecendo a comunicação entre a aplicação Web e os módulos que processam e analisam os *logs*. Assim, as informações fornecidas pelo serviço Web alimentam a aplicação Web para apoiar adaptações pré-programadas pelo desenvolvedor.

Durante a navegação do usuário, o serviço Web do TAWS recebe requisições da aplicação Web e, de modo assíncrono, realiza consultas específicas aos mecanismos de Coleta de *Logs*, de Análise de Tarefas e de Repositório de Conhecimento. Por exemplo, se a aplicação requer as páginas visitadas pelo usuário ativo, o serviço Web realiza uma consulta no banco de dados *real-time* do mecanismo de Coleta de *Logs*; se a aplicação requer os padrões de comportamento executados pelo usuário ativo, o serviço Web compara as ações do usuário com o banco de dados implementado no Repositório de Conhecimento.

Para as requisições, a aplicação Web utiliza uma biblioteca, chamada **jUsabilics**. Esta biblioteca realiza as requisições ao serviço Web do TAWS de modo assíncrono. Após o serviço Web processar a requisição, o TAWS retorna à aplicação Web as informações consultadas.

As requisições, que consultam os outros módulos da RUM, são classificadas em cinco categorias, descritas nas Seções 4.2.5.1 a 4.2.5.5.

4.2.5.1 Consultas em nível de domínio

As requisições em nível de domínio recuperam informações sobre as interações dos usuários ativos em toda a aplicação. Através dessas consultas, o desenvolvedor pode implementar, por exemplo, ferramentas de visualização em tempo real, caso queira monitorar as ações dos usuários. A Tabela 4.2 lista as consultas disponíveis na biblioteca jUsabilics para o nível de domínio.

Tabela 4.2 - Consultas da biblioteca jUsabilics em nível de domínio.

Método / Descrição
getInteractionsByScreenSize Seleciona as interações de tamanhos de tela específicos
getInteractionsByBrowserAndPlatformAndLanguage Seleciona as interações a partir de navegadores, sistemas operacionais e línguas específicos

4.2.5.2 Consultas em nível de interação

Se o desenvolvedor necessita de mais detalhes do comportamento de cada usuário, então devem ser usadas as consultas em nível de interação, listadas na Tabela 4.3. Estas consultas retornam a duração da interação e as páginas visitadas.

Tabela 4.3 - Consultas da biblioteca jUsabilics em nível de interação.

Método / Descrição
getElapsedTimeOfInteraction Recupera o tempo decorrido de interação
getLastVisitedPages Recupera as últimas N páginas visitadas
isVisitedPageLike Verifica se uma página foi visitada

4.2.5.3 Consultas em nível de eventos

Além das informações de páginas visitadas, o serviço Web provê à aplicação Web todos os eventos realizados pelo usuário ativo. As consultas em nível de eventos fornecem ao desenvolvedor dados sobre os eventos do usuário na interface, que podem ser filtrados pela combinação de seus atributos. Por meio dessas consultas, o desenvolvedor pode estender a análise do TAWS implementando tipos de análise es-

pecíficos do modelo de negócio de sua aplicação. A Tabela 4.4 descreve as consultas desse nível.

Tabela 4.4 - Consultas da biblioteca jUsabilics em nível de eventos.

Método / Descrição
getEventsOfInteraction Recupera os eventos de uma interação específica
getLastEventsOfInteraction Recupera os últimos N eventos de uma interação
getEventsOfInteractionByTimestamp Seleciona os eventos da interação entre um intervalo de tempo
getEventsByScroll Seleciona os eventos realizados em uma determinada área da página
getEventsByPage Seleciona os eventos realizados em uma página específica
getEventsInContainerAndPage Seleciona os eventos realizados em um *container* específico, conforme o modelo de interface COP
getEventsByObjectAndContainerAndPage Seleciona os eventos realizados em um objeto e/ou container específico, conforme o modelo de interface COP

As requisições apontadas anteriormente são atendidas pelo mecanismo de Coleta de *Logs* e objetivam fornecer à aplicação Web dados estruturados sobre a interação do usuário ativo. No entanto, a aplicação Web consume também os resultados da avaliação de usabilidade, conforme relatado na Seção .

4.2.5.4 Consultas de avaliação de usabilidade

As requisições desta categoria, descritas na Tabela 4.5, informam a aplicação Web sobre o índice de usabilidade e as ações erradas de tarefas em andamento ou já executadas pelo usuário. Portanto, uma vez que a aplicação Web detecta um baixo índice de usabilidade de uma tarefa, adaptações na interface podem ser disparadas para auxiliar o usuário. Não limitado a usar somente o índice de usabilidade, o serviço Web fornece detalhes sobre as ações erradas detectadas durante a análise de tarefas.

Por fim, o serviço Web, em conjunto com o Repositório de Conhecimento, executa a análise de padrões durante a navegação, comparando as ações do usuário ativo aos padrões selecionados pelo especialista da aplicação. A Seção 4.2.5.5 detalha a forma

Tabela 4.5 - Consultas da biblioteca jUsabilics sobre avaliação de usabilidade.

Método / Descrição
getWrongActionsByTask Recupera as ações erradas detectadas em uma determinada tarefa, de acordo com a análise de tarefas.
getWrongActionsByObjectAndContainerAndPage Recupera as ações erradas detectadas em um objeto e/ou container específico
getCurrentUsabilityIndexByTask Recupera o índice de usabilidade atual de uma tarefa específica.

pela qual a aplicação consome os resultados desse processamento.

4.2.5.5 Consultas de padrões de comportamento

Como foi explicado na Seção 4.2.4, o especialista da aplicação associa padrões de comportamento a perfis de usuários ou a ações para adaptação, que correspondem a um conjunto de padrões. Sendo assim, a aplicação Web requisita ao serviço Web os padrões já executados pelos usuários e, consequentemente, faz requisições específicas para detectar se um perfil de usuário foi detectado ou se uma adaptação deve ser disparada. Essas requisições são descritas na Tabela 4.6.

Se uma ação para adaptação for engatilhada no Repositório de Conhecimento, o serviço Web reportará os padrões detectados. Então, a aplicação Web receberá uma mensagem indicando qual ação deve ser disparada, e uma ação pré-programada na interface pelo desenvolvedor poderá ser invocada. Portanto, a aplicação Web pode se adaptar de acordo com o comportamento e as necessidades do usuário ativo.

Tabela 4.6 - Consultas da biblioteca jUsabilics sobre padrões de comportamento.

Método / Descrição
getCompletedPatterns Recupera os padrões de comportamento já executados pelo usuário na interação.

Os estudos de caso para construção de aplicações Web adaptativas com a abordagem RUM são demonstrados no Capítulo 5, que inclui exemplos de uso da biblioteca jUsabilics.

No intuito de posicionar a abordagem entre os trabalhos reportados na literatura,

foi realizada uma pesquisa bibliográfica no campo de análise do comportamento do usuário através de *logs*. Neste sentido, a Seção 4.3 discorre sobre diferentes abordagens, ferramentas e métodos reportados nessa área, comparando-os com a abordagem RUM.

4.3 Trabalhos relacionados

A análise do comportamento de usuários da Web tem sido investigada com diferentes objetivos na última década. Serdyukov (2014), por exemplo, explora dados comportamentais para melhorar a experiência de busca, enquanto outros trabalhos exploram esse tipo de análise para fornecer estatísticas dos dados dos usuários (QUINCEY et al., 2009), melhorar a visualização dos dados (KHOURY et al., 2011) e mineração de *logs* (THOMAS, 2014).

A literatura também reporta trabalhos direcionados a tipos específicos de aplicações Web, tais como *e-learning* (KUO et al., 2005) ou *web sites* que hospedam e distribuem vídeos (SARUKKAI, 2013).

Quincey et al. (2009) desenvolveram uma abordagem que executa análise estatística dos dados capturados. A informação produzida a partir desse trabalho não apenas fornece um *framework* para uma análise de usabilidade posterior, mas também um *framework* para predição do comportamento humano no campo da computação gráfica. Este *framework* opera em modo assíncrono.

Peska et al. (2011) propuseram um componente implementado na linguagem PHP chamado *UPComp*, que usa as preferências dos usuários para sugerir recomendações. UPComp é um componente *standalone* que pode ser integrado com qualquer aplicação Web desenvolvida em PHP. O *toolkit* TAWS, de outra maneira, implementa um código JavaScript que qualquer aplicação Web pode usá-lo, não apenas aplicações escritas na linguagem PHP.

Apaolaza et al. (2013) desenvolveram uma ferramenta que é facilmente implantável em qualquer aplicação Web e captura dados das interações dos usuários entre as páginas. Diferentemente da abordagem RUM, essa ferramenta é destinada a melhorar a acessibilidade em aplicações Web. Nenhuma das ferramentas realiza a análise durante a navegação.

DAL é proposto por Sarukkai (2013). É um sistema que considera a interação do usuário para sugestão de anúncios dinamicamente, e foi projetado para aprimorar a experiência do usuário no YouTube.

Thomas apresentou o LATTE (THOMAS, 2014), uma ferramenta de software que extrai informação das interações e correlaciona padrões de páginas visualizadas às ações do usuário. Esses padrões ajudam o desenvolvedor Web a entender as dificuldades de navegação dos usuários em seus *web sites*.

Semelhantemente ao TAWS, Google Analytics fornece uma API (GOOGLE, 2016) que permite à aplicação Web consumir dados sobre o usuário ativo durante a navegação, o navegador utilizado, as páginas visualizadas, etc. Além de fornecer esses mesmos dados, a RUM também fornece à aplicação Web informações concernentes à usabilidade da aplicação, tais como ações erradas e o índice de usabilidade das tarefas, além de padrões de comportamento executados pelo usuário ativo.

Abbar et al. (2013) produziram uma abordagem para análise durante a navegação. A análise é realizada no conteúdo acessado pelo usuário, diferentemente da abordagem RUM, que analisa ações dos usuários na interface. A abordagem de Abbar et al. é direcionada apenas a sites de notícias e seu objetivo é recomendar artigos relevantes ao usuário durante a navegação.

Mobasher et al. (1999) conduziram uma pesquisa para analisar a sessão do usuário ativo e construir *web sites* adaptativos com a recomendação de páginas. Os *logs* coletados armazenam apenas as URL visitadas. A partir de Mobasher et al. (1999), há pesquisas mais recentes para construção de perfis de usuários baseada em *logs* de servidor (FUJIMOTO et al., 2011) e ainda trabalhos dedicados à predição da próxima página a ser visitada (SEN et al., 2016). Khonsha e Sadreddini (2011) propuseram um *framework* para personalização Web baseada em mineração de dados que combina dados da interação com a estrutura e o conteúdo do *web site*, a fim de realizar a predição de requisições de páginas. Os resultados das recomendações são relevantes se comparados a trabalhos anteriores. Em nenhuma das pesquisas há uma maneira de consumir os resultados da análise durante a navegação para adaptar a interface, como é possível na RUM.

A respeito de melhorias de usabilidade nas interfaces, Vora e Bojewar (VORA; BOJEWAR, 2011) projetaram uma ferramenta que usa análise estatística de Mineração de Dados na Web direcionada à personalização Web. A ferramenta fornece diferentes formas de visualização de informações sobre o comportamento do usuário, entretanto ela não fornece informações durante a navegação do usuário.

Ramya e Sajeev (RAMYA; SAJEEV, 2015) implementaram um sistema de personalização que explora o atributo tempo do comportamento do usuário. O sistema proposta

minera *logs* de servidor a fim de detectar padrões usando os seguintes atributos: duração da visita, intervalo entre visitas e tempo de saída. Porém, o sistema não oferece uma maneira de adaptar a interface durante a navegação.

W3Touch (NEBELING et al., 2013) é um sistema que coleta *logs* de cliente e informações sobre os elementos HTML da página para detectar problemas de espaçamento no layout, em termos de tamanho e posição do texto, imagens e *links*. Esse sistema é limitado à análise de duas métricas: cliques errados nos elementos e média das ações de *zoom* na tela.

Cerny et. al. (CERNY et al., 2013) descreveram o *framework* READ, uma técnica que fornece interfaces adaptativas reduzindo o esforço de desenvolvimento e manutenção. READ está de acordo com os padrões de desenvolvimento de aplicações usados na indústria para suportar uma transição fácil da fase de projeto para os sistemas em produção. Comparando com nossa abordagem, o *framework* READ requer a integração da técnica proposta com a camada de domínio da aplicação Web. A abordagem RUM interfere menos no desenvolvimento porque está integrado apenas com a interface da aplicação Web.

Akiki (AKIKI, 2013) desenvolveram uma plataforma genérica, escalável e extensível para construir interface adaptativas em aplicações empresariais. A plataforma objetiva simplificar a interface da aplicação através de modelos e níveis de abstração, até alcançar o nível de implementação da interface. Entretanto, a plataforma proposta não considera padrões de navegação e ações do usuário na interface, que são essenciais para adaptar a interface de acordo com o comportamento do usuário ativo.

4.4 Considerações finais

Neste capítulo, foi explicada a principal contribuição desta pesquisa, a abordagem RUM. Inicialmente, detalhou-se a arquitetura da abordagem, destacando o propósito e o funcionamento de cada módulo. Em seguida, foram especificados detalhes de implementação do TAWS, um *toolkit* construído com tecnologias que o tornam implantável em qualquer aplicação Web. Por fim, foram citados os aspectos mais relevantes dos trabalhos relacionados à abordagem RUM reportados na literatura.

A fim de validar e demonstrar a abordagem, quatro estudos de caso foram planejados e conduzidos destacando a aplicabilidade da RUM na construção de aplicações Web adaptativas. O Capítulo 5 descreve esses estudos.

5 ESTUDOS DE CASO E RESULTADOS

Aplicações Web modernas não são compostas por apenas simples páginas HTML para exibição de um texto. Atualmente, as aplicações Web oferecem ao usuário diferentes recursos interativos que buscam facilitar a sua navegação, como as caixas de busca e os *links* que carregam conteúdos de modo assíncrono, por exemplo. Nesse cenário, demonstrar a abordagem RUM requer diferentes aplicacões Web, com diferentes perfis de usuários.

Além disso, há a necessidade de demonstrar as diferentes possibilidades de uso da RUM e analisar os diferentes mecanismos de análise do comportamento do usuário durante a navegação, desde a análise de tarefas à detecção de padrões. Portanto, foram realizados quatro estudos de caso em aplicações Web diferentes. O primeiro (VASCONCELOS et al., 2014), apresentado na Seção 5.1, foi o passo inicial para o desenvolvimento da abordagem durante a interação do usuário, em que o objetivo era classificar os usuários pelo nível de experiência com a aplicação Web. Posteriormente, no segundo estudo de caso (VASCONCELOS et al., 2016), relatado na Seção 5.2, o propósito foi adaptar a interface durante a navegação a partir da análise de tarefas. No terceiro, discutido na Seção 5.3, o objetivo foi disparar adaptações na interface consumindo os resultados da detecção de padrões durante a navegação do usuário. Por fim, no quarto estudo de caso, apresentado na Seção 5.4, desenvolvedores externos utilizaram o *toolkit* em diferentes cenários e descreveram suas percepções quanto ao uso e, principalmente, quanto à biblioteca *jUsabilics*.

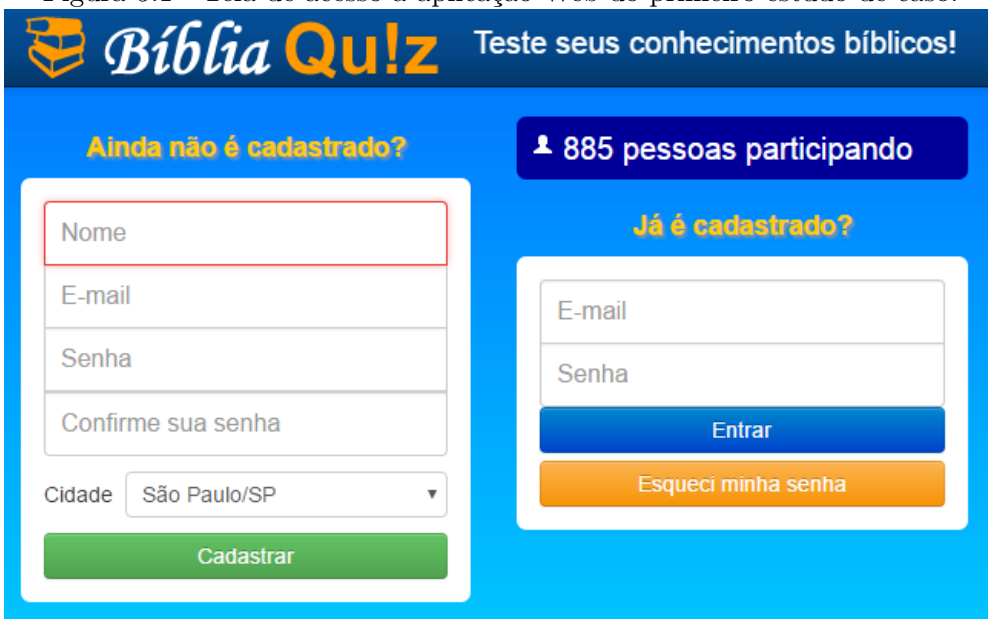
5.1 Classificação da experiência do usuário durante a navegação

Este estudo de caso foi motivado pela seguinte hipótese: *é possível classificar os usuários analisando logs de cliente durante a navegação?* O maior desafio para a proposta de uma solução que envolva *logs* de cliente é o processamento do grande volume de dados. Portanto, é explorado o método de análise de tarefas apresentado na Seção 4.1.2.1, que processa subconjuntos dos logs. Os subconjuntos correspondem a tentativas de execução de alguma tarefa, reduzindo significativamente o volume de dados para processamento.

Então, foi desenvolvido o algoritmo apresentado na Seção 4.2.2, a fim de realizar a análise de tarefas durante a navegação para fins de avaliação de usabilidade. O objetivo foi demonstrar a capacidade do módulo de tarefas da abordagem RUM para calcular o índice de usabilidade durante a navegação do usuário.

Neste estudo de caso, foi selecionado um *web site* em que os usuários respondem perguntas de conhecimento bíblico e ganham pontos pelas respostas corretas. Para acessar o *web site*, o usuário deve se cadastrar e, em seguida, deve informar os dados de acesso (e-mail e senha), conforme mostra a Figura 5.1. Quando uma questão é exibida ao usuário, um relógio decrementa o tempo a cada segundo e as opções "Responder" e "Não sei, quero responder depois" são apresentadas (Figura 5.2).

Figura 5.1 - Tela de acesso à aplicação Web do primeiro estudo de caso.



Como o *web site* é atualizado diariamente com novas perguntas, os usuários que participam do quiz têm o costume de acessá-lo e, em seguida, responder as perguntas do dia. No entanto, também há o acesso de usuários iniciantes, que realizam o cadastro e, em seguida, acessam o quiz pela primeira vez. Para mapear essas interações, a tarefa *Entrar e responder* foi definida pelo especialista usando a ferramenta UsaTasker (VASCONCELOS; BALDOCHI JR., 2012), sendo que o caminho ótimo da tarefa é composto pelas seguintes ações:

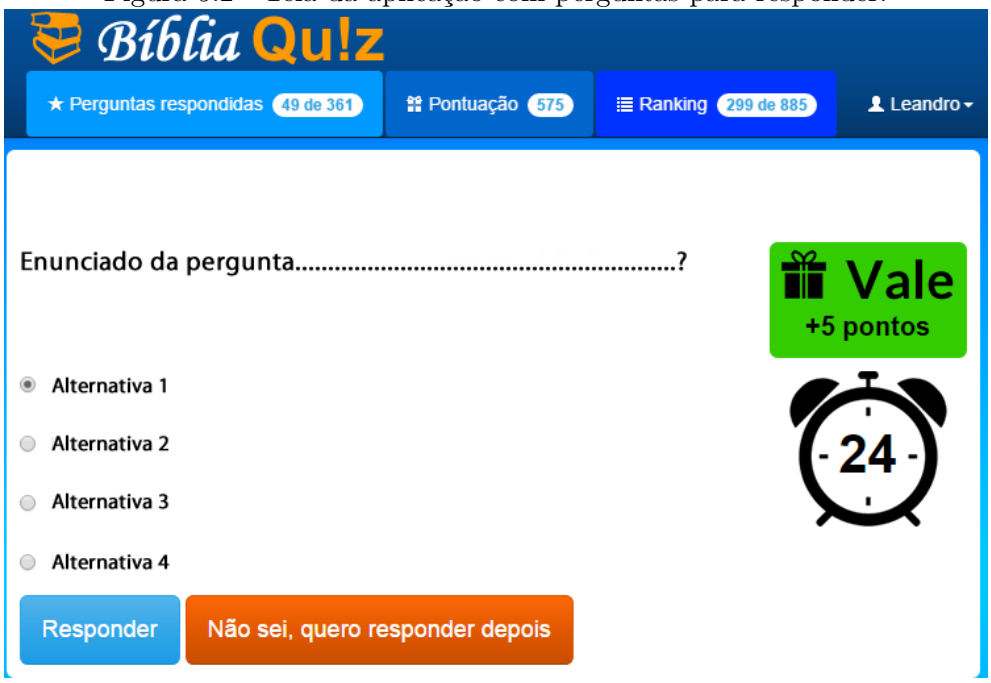
a) Digitar o e-mail

b) Digitar a senha

c) Clicar no botão *Entrar*

d) Aguardar a exibição da questão

Figura 5.2 - Tela da aplicação com perguntas para responder.



e) Selecionar uma alternativa

f) Clicar no botão *Responder*

À medida que o usuário interage com o *web site*, o índice de usabilidade é calculado pelo algoritmo, a fim de classificar a experiência do usuário na interação. Para otimizar o processamento das ações do usuário para o cálculo do índice, o repositório temporário definido no algoritmo foi implementado com o mecanismo *Memory* do sistema gerenciador de banco de dados MySQL. Esse mecanismo armazena temporariamente os dados em memória.

Durante uma semana, as interações dos usuários foram coletadas e analisadas durante a navegação. Em quarenta e uma interações, houve a tentativa de execução da tarefa definida pelo especialista. Para cada interação, foram calculados os índices de eficácia e eficiência da interface.

A eficácia é a quantidade de passos que o usuário completou no caminho ótimo da tarefa. A eficiência é a taxa de erros entre o caminho percorrido pelo usuário e o caminho ótimo. Portanto, quando uma interação tem uma pequena taxa de erros, a eficiência é alta; quando o usuário completa todos os passos da tarefa, a eficácia é alta, mas isso não garante que a eficiência será proporcionalmente alta. Por exemplo, se um usuário completa todos os passos da tarefa acima, mas, antes do

passo *Selecionar uma alternativa*, desvia-se do caminho ótimo clicando em outros *links* da página, então o índice da eficiência será prejudicado, apesar de o índice de eficácia ser máximo por ter completado a tarefa.

Das 41 interações, 22 usuários (53,6%) completaram a tarefa definida pelo especialista e 19 (46,4%) não completaram. Das interações em que a tarefa não foi completada, a média do índice de eficácia foi igual a 0,68, ou seja, os usuários que não completaram a tarefa percorreram, em média, 68% do caminho ótimo. Entre todas as interações, o índice médio de eficiência foi 0,87, que representa uma baixa taxa de erros e suporta a conclusão de que, de acordo com o índice, a interface oferece boa usabilidade nessa tarefa.

Diante da hipótese que motivou o estudo de caso, o objetivo foi classificar o usuário conforme sua experiência com a interface do *web site*. Arbitrariamente, três níveis foram definidos: Iniciante, Intermediário e *Expert*. Um usuário iniciante tende a executar mais ações na interface e a desviar-se do caminho ótimo da tarefa; já um usuário *expert* realiza a tarefa com mais eficiência, executando menos ações. A classificação foi realizada através de dois métodos baseados na eficácia e eficiência da interface, esses métodos são explicados nas Seções 5.1.1 e 5.1.2.

5.1.1 Classificação empírica

Neste método de classificação, valores arbitrários foram atribuídos aos níveis de experiência da interação: Iniciante, valores menores que 0,3; Intermediário, valores entre 0,3 e 0,7; e *Expert*, valores entre 0,7 e 1,0. Esses valores foram usados para balizar os índices de eficácia e de eficiência de cada interação. O resultado da classificação empírica é mostrado na Figura 5.3. Segundo este método de classificação, observa-se que nenhum usuário foi classificado como iniciante e a maioria das interações (75,6%) foi classificada como *Expert*, devido ao alto índice de eficiência.

5.1.2 Classificação estatística

Neste método, foi realizada uma interpretação estatística a partir do princípio de funcionamento de uma carta de controle, que é um tipo de gráfico usado para acompanhamento de um processo. Neste caso, o processo a ser analisado é o comportamento dos usuários ao longo do tempo, a fim de classificar a experiência de cada usuário em relação à experiência dos usuários anteriores. Esse tipo de gráfico determina estatisticamente uma faixa de limites de controle, além de uma linha média. Para cada interação, a média da eficiência das interações anteriores foi calculada.

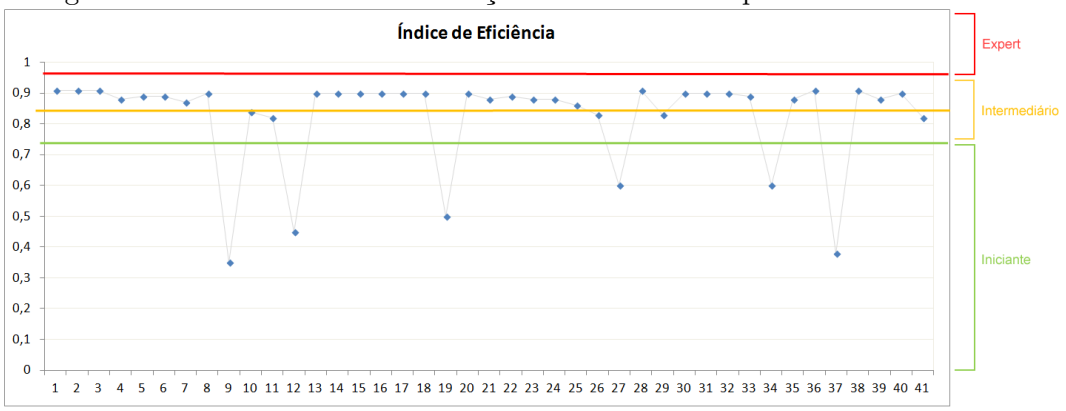
Figura 5.3 - Resultado da classificação empírica da experiência dos usuários.

Em seguida, foi calculado o desvio-padrão dos índices para definir os limites inferior e superior de controle. O limite inferior de controle é a diferença entre a média e o desvio-padrão; já o limite superior é a média acrescida do desvio-padrão.

Interações com índice de eficiência entre os limites de controle foram classificadas como Intermediário, pois representam o comportamento padrão dos usuários; interações cujo índice de eficiência era menor que o limite inferior de controle foram classificadas como Iniciante, pois a eficiência menor que o limite inferior de controle representa interações abaixo do comportamento padrão dos usuários. Para o nível *Expert* foi considerado o índice de eficiência maior que o limite superior de controle.

A Figura 5.4 apresenta o resultado desse método de classificação. Na análise de carta de controle, a maioria dos dados é classificada dentro dos limites de controle, cuja área refere-se ao nível Intermediário de experiência do usuário nesta análise. No entanto, a vantagem deste método de classificação é a identificação de usuários que apresentam comportamento fora do padrão. Apesar de não ser objetivo da análise informar detalhes sobre as interações fora dos limites de controle, é possível identificá-las para que sejam investigadas com outros métodos de análise posteriormente. Por exemplo, as interações cujo índice está abaixo do limite inferior de controle podem ser analisadas individualmente para a detecção de problemas de usabilidade que os usuários inexperientes enfrentam. Já as interações classificadas acima do limite superior de controle podem ser analisadas para saber se há outros caminhos escolhidos pelos usuários que podem também ser considerados como caminhos ótimos da tarefa.

Figura 5.4 - Resultado da classificação estatística da experiência dos usuários.



5.1.3 Discussão

O uso dos métodos empírico e estatístico permite compreender que a classificação do nível de experiência depende do histórico de comportamento dos usuários. No resultado do método empírico de classificação, nenhum usuário foi classificado como Iniciante e a maioria dos usuários foi classificada como *Expert*, devido ao alto índice de eficiência. No entanto, no resultado do método estatístico observa-se que o comportamento dos usuários na interface avaliada é, em geral, próximo do caminho ótimo, pois a média do índice de eficiência foi 0,87. Sendo assim, nenhum usuário foi classificado como *Expert*. Além disso, seis usuários foram classificados como Iniciantes, tendo em vista o baixo índice de eficiência.

A implementação do algoritmo foi monitorada continuamente para mensurar a latência entre o envio das ações do usuário e o cálculo do índice de usabilidade. O monitoramento ocorreu através do registro da latência em um arquivo. Com o mecanismo *Memory* do sistema gerenciador de banco de dados relacional, o tempo médio de latência para o cálculo do índice de usabilidade foi de 194 milissegundos, que é considerado um tempo suficiente para resposta durante a navegação do usuário.

Um fator que leva à eficiência é a limitação dos *logs* que são analisados, que considera apenas subconjuntos de dados relacionados às tarefas definidas pelo especialista. Outro fator é o cálculo incremental do índice de usabilidade no algoritmo implementado. Com isso, foi possível classificar o usuário durante a navegação conforme seu nível de experiência com o *web site*.

Projetando o cenário para milhares de usuários simultâneos em uma aplicação Web, é provável que haja um gargalo no mecanismo de banco de dados, pois usar um

mecanismo em memória requer uma infraestrutura robusta de servidores, principalmente no que se refere à capacidade de memória RAM. A aplicação usada neste estudo foi hospedada em um servidor *Cloud* com sistema operacional Linux, distribuição Ubuntu Server 12.04 64-bits, com 1 processador de 2 GHz, 2 GB de memória RAM e 40 GB de disco rígido.

Sendo assim, após a análise deste estudo de caso, foi realizado um estudo de novos conceitos e tecnologias que otimizem a coleta e o processamento dos logs. Apesar do sucesso do estudo para a hipótese investigada, houve a necessidade de projetar uma nova arquitetura tecnológica para tornar escalável a análise de tarefas durante a navegação.

No segundo estudo de caso, apresentado na Seção 5.2, com a implementação da nova arquitetura, há a demonstração da análise de tarefas e a construção de adaptações na interface de um *web site*.

5.2 Análise de tarefas durante a navegação para apoiar aplicações Web adaptativas

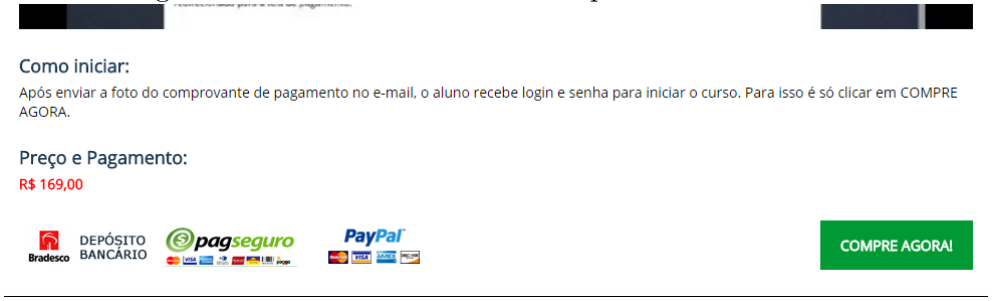
Para demonstrar a eficácia dos módulos de Coleta, Análise de Tarefas e de Serviços da abordagem RUM, foi realizado um estudo de caso com o desafio de aprimorar a usabilidade de um *web site* durante a navegação do usuário. Consequentemente, objetivou-se verificar se a abordagem RUM, através do TAWS, está apta a fornecer informações relevantes sobre o comportamento do usuário durante a navegação, isto é, enquanto o usuário está realizando uma tarefa. Neste estudo, portanto, há a integração da análise de tarefas durante a navegação e do uso da biblioteca *jUsabilics*.

Neste estudo de caso, foi escolhido um *web site* de *e-commerce* que vende cursos de treinamentos esportivos e foi analisada a principal tarefa do *e-commerce*: comprar um curso *online*. Essa tarefa é composta pelos seguintes passos:

1. Selecionar um curso no menu principal para ver seus detalhes (Figura 5.5);
2. Clicar no botão *Compre agora* (Figura 5.6);
3. Selecionar o método de pagamento (Figura 5.7);
4. Preencher um formulário contendo os campos: nome, endereço, e-mail, telefone, data de nascimento e informações adicionais (Figura 5.8).

Figura 5.5 - Passo 1 da tarefa “Comprar um curso online”.

Figura 5.6 - Passo 2 da tarefa “Comprar um curso online”.



Depois do preenchimento do formulário, uma página de confirmação é apresentada ao usuário indicando o procedimento de pagamento (PayPal ou depósito bancário). O pagamento é feito fora do *web site*.

Inicialmente, a tarefa foi definida pelo especialista da aplicação. Em seguida, o especialista analisou os passos da tarefa, especificamente a disposição dos elementos nas páginas, e identificou dois possíveis problemas de usabilidade:

- **Problema 1:** a posição do botão *Comprar* na página de detalhes do curso está no rodapé e, então, pode ser difícil encontrá-lo para realizar o passo 2 da tarefa; e
- **Problema 2:** a falta de orientação no preenchimento do formulário do passo 4.

Então, o objetivo do estudo de caso foi verificar a capacidade da abordagem RUM em apoiar a adaptação da interface durante a navegação após a detecção dos problemas de usabilidade de maneira automatizada.

Figura 5.7 - Passo 3 da tarefa “Comprar um curso online”.

Portanto, foram implementadas regras de adaptação da interface que consomem informações durante a navegação, a fim de detectar a presença dos problemas de usabilidade apresentados.

Para o *Problema 1*, a regra implementada verifica o uso da barra de rolagem alternando a direção vertical (para cima/para baixo). Se isso ocorre e o índice de usabilidade é inferior a um valor arbitrário, o botão *Comprar* é reposicionado a fim de facilitar sua visualização. Isso foi feito explorando os métodos *getEventsByPage* e *getCurrentUsabilityIndexByTask* da biblioteca jUsabilics, que foram invocados dinamicamente em intervalos de 500 ms.

O *script* abaixo ilustra a chamada do método *getCurrentUsabilityIndexByTask*, que requer como parâmetro a identificação da tarefa definida pelo especialista da aplicação. Se o resultado índice de usabilidade for menor que 0,5 , o método *getEventsByPage* é invocado. Este método requer como parâmetros a página em que o usuário está navegando e o tipo de ação que se deseja coletar do usuário ativo. No exemplo, eventos de rolagem (*scroll*) foram selecionados a fim de identificar a posição da barra de rolagem nas ações dos usuários.

```
jUsabilics.getCurrentUsabilityIndexByTask(123, function(data){
    if (data.usabilityIndex < 0.5){
        jUsabilics.getEventsByPage(window.location.href, "scroll",
            function(events){
                //código para verificar se o usuário
```

Figura 5.8 - Recorte da tela do Passo 4 da tarefa “Comprar um curso online”.


```
                //está alternando a direção da barra
                //de rolagem
    })
    }
})
```

Para o *Problema 2*, a regra de adaptação implementada verifica o uso da barra de rolagem enquanto o usuário está preenchendo o formulário. Se o usuário rola até o final da página e depois retorna ao topo, uma mensagem de ajuda é apresentada, orientando o usuário sobre o preenchimento do formulário.

5.2.1 Recrutamento de usuários

Para este estudo de caso, foram selecionados dez usuários inexperientes em relação ao *web site* avaliado, isto é, usuários que nunca acessaram o *web site* previamente. Através de um breve questionário, os usuários informaram a faixa etária e a última formação acadêmica. A idade dos usuários está entre 24-39 anos e eles têm diferentes níveis de formação acadêmica, do ensino médio ao ensino superior.

Todos os usuários receberam a mesma instrução: usar o *web site* para comprar um curso específico. A escolha de um curso específico foi feita para que os usuários tenham um alvo para realizar a tarefa. Os usuários realizaram a tarefa em seus computadores pessoais.

5.2.2 Análise

Durante a execução das dez tarefas pelos usuários recrutados, o índice de usabilidade foi calculado durante a navegação e as ações erradas foram identificadas. A hipótese do *Problema 2* não foi encontrada em nenhuma das dez execuções, ou seja, nenhum usuário teve dificuldade em preencher o formulário. Entretanto, o *Problema 1* foi identificado em seis execuções de tarefa e, consequentemente, a regra de adaptação foi disparada durante a navegação, exibindo o botão *Comprar* mais apropriadamente. Em quatro tarefas, os usuários completaram a compra sem a necessidade de adaptação da interface.

O objetivo de adaptar a interface durante a navegação é auxiliar o usuário durante a execução da tarefa. Portanto, a adaptação da interface ocorre depois da identificação de qualquer dificuldade na interação. Então, a adaptação deve melhorar a usabilidade da interface para o usuário ativo.

No intuito de verificar a eficácia das regras de adaptação para melhorar a usabilidade, as interações em que houve adaptação são comparadas com aquelas em que não houve. São critérios da comparação: a quantidade de ações erradas e o valor do índice de usabilidade. Seis dos dez usuários encontraram dificuldades na execução da tarefa, disparando a adaptação.

A Tabela 5.1 mostra a quantidade média de eventos *mouseover* e *scroll* detectados nas interações. Esses eventos são comumente realizados por usuários quando estão procurando um elemento na interface. A tabela também mostra o índice de usabilidade da tarefa.

Para as interações em que não houve adaptação (4 interações), o índice de usabilidade foi de 76%, e a média do número de eventos *scroll* foi 20,7. Para as 6 interações em que houve adaptação, a quantidade média de eventos *mouseover* e de eventos *scroll* foi alta, de forma que as interações adaptadas apresentaram 45% mais eventos *mouseover* e *scroll*. Entretanto, devido à adaptação, o índice de usabilidade foi 75% – apenas 1% menor que o índice das interações em que os usuários não encontraram dificuldade.

A adaptação foi disparada apenas para os usuários que enfrentaram algum problema de usabilidade, ou seja, sem a adaptação esses usuários executaríam mais ações diferentes do caminho ótimo, o que resultaria em um índice de usabilidade inferior. Devido à adaptação, esses usuários foram auxiliados durante a navegação, o que permitiu que corrigissem o caminho que estavam percorrendo. Isso resultou em um índice de usabilidade praticamente igual ao índice de usabilidade das interações dos usuários que não enfrentaram dificuldades.

Esses resultados mostram que a regra de adaptação implementada para o primeiro problema identificou as ações erradas (eventos *mouseover* e *scroll*) em tempo suficiente para disparar a adaptação. Tão logo o problema de usabilidade foi identificado e a adaptação foi disparada, o número de ações erradas foi reduzido significativamente. Mais importante que isso, o valor do índice de usabilidade é praticamente o mesmo nos dois casos, retratando que a adaptação foi capaz de aprimorar a usabilidade enquanto o usuário estava tentando realizar a tarefa.

Tabela 5.1 - Comparação das interações.

Sem adaptação			**Com adaptação**		
MouseOver	Scroll	Index	MouseOver	Scroll	Index
79,2	20,7	76%	114,6	30,3	75%

5.2.3 Discussão

Neste estudo de caso, houve a demonstração da análise de tarefas da abordagem RUM integrada à biblioteca *jUsabilics*. Através da biblioteca *jUsabilics*, implementada para consumir o serviço Web do TAWS, a aplicação Web detectou o índice de usabilidade e ações erradas durante as interações, disparando uma adaptação pré-programada para aprimorar a usabilidade em um determinado passo da tarefa.

Para os dois estudos de caso apresentados anteriormente, os módulos de Coleta de *Logs*, de Análise de Tarefas e de Serviços foram necessários, pois os estudos estão relacionados à análise de tarefas.

A análise de tarefas cobre muitos aspectos de uma aplicação Web, porém, como discutido no final da Seção 4.1.2.1, há a necessidade de compreender o comportamento do usuário também quando não está realizando uma tarefa. Para isso, um terceiro estudo de caso foi realizado para demonstrar a detecção de padrões durante a navegação, consumindo os resultados do módulo de KDD e de Repositório de

Conhecimento da RUM.

5.3 Aplicação Web adaptativa baseada em padrões de comportamento dos usuários

Em alguns *web sites*, há perfis de usuários muito bem definidos, pois há conteúdos específicos para cada perfil. No Instituto Nacional de Pesquisas Espaciais (INPE), um dos principais *web sites* é disponibilizado pelo Centro de Previsão de Tempo e Estudos Climáticos - CPTEC. Segundo informações fornecidas pelos desenvolvedores do *web site*, há mais de 2 milhões de visitas por mês, de variados perfis de usuários, tais como: cidadãos comuns, pesquisadores, jornalistas e estudantes. Dentro do *web site* do CPTEC, há uma área especializada sobre Tempo (*tempo.cptec.inpe.br*, mostrada na Figura 5.9), que fornece informações técnicas e não-técnicas sobre previsão do tempo nas diferentes regiões do território brasileiro.

Figura 5.9 - Página principal do CPTEC sobre Tempo.

Para os cidadãos em geral, o *web site* oferece a previsão do tempo por cidade, em que o usuário pode pesquisar sua cidade através de um formulário. Além disso, o *web site* exibe na página principal a previsão para as capitais dos estados brasileiros e para as cidades onde há aeroportos.

Diariamente, os meteorologistas do CPTEC/INPE atualizam no *web site* o resultado de diferentes análises, tais como: análise de carta sinótica de superfície, análise da imagem de satélite para o Brasil, análise de temperaturas máxima e mínima para as

capitais e análise das condições de tempo para cada região do Brasil. Esses recursos são destinados a pesquisadores e jornalistas.

Para pesquisadores interessados nos dados do CPTEC/INPE, estão disponíveis informações sobre avisos meteorológicos e um recurso para o pesquisador realizar sua própria análise sinótica.

Este estudo de caso foi realizado em parceria com os desenvolvedores do *web site* do CPTEC/INPE. Inicialmente, o TAWS foi implantado para coletar as interações dos usuários. Para isso, o TAWS foi hospedado em um servidor *Cloud* semelhante ao usado no primeiro estudo de caso, com sistema operacional Linux, distribuição Ubuntu Server 14.04 64-bits, com 1 processador de 2 GHz, 2 GB de memória RAM e 40 GB de disco rígido.

Na primeira fase do estudo, uma amostra de 60.664 sessões de usuários distintos foi coletada a partir de diversos dispositivos, detectando mais de 3,6 milhões de ações na interface. Com os *logs* coletados, entrou em funcionamento o módulo de KDD automatizado para extração de padrões. Uma característica do *web site* do Tempo é a existência de uma única página que carrega conteúdo dinamicamente, como muitas aplicações Web modernas. Sendo assim, dentre os conjuntos de atributos especificados na Seção 4.2.3, foi selecionado o conjunto *ClickedLinks*, que contém os links clicados em cada interação para detecção de padrões sequenciais. Esse conjunto foi escolhido com o objetivo de analisar como o usuário interage com a principal área da página, que contém os links *Análise Sinótica*, *Satélite*, *Condições de Tempo*, etc.

Devido à inovação do método proposto nesta tese, optou-se por agendar uma reunião com os desenvolvedores do *web site*, que são especialista no *web site* e conhecem os diferentes perfis de usuários e todo o conteúdo disponível ao público. Nessa reunião, os padrões detectados pelo *toolkit* foram apresentados a eles. Foram detectadas 64 sequências relevantes, que representam padrões sequenciais dos usuários. Um subconjunto dessas sequências é apresentado na Tabela 5.2.

Em grupo, os especialistas analisaram as sequências a fim de inferir a relação entre cada sequência e possíveis perfis conhecidos de usuários. Após a análise, foram definidas 5 ações de adaptação associadas a determinadas sequências selecionadas pelos especialistas. A Tabela 5.3 lista as ações de adaptação e os padrões associados a cada uma. As ações escolhidas para adaptação visam atender a perfis específicos de pesquisadores que acessam o *web site* do Tempo do CPTEC/INPE. Esses perfis foram escolhidos devido ao fato de que, na visão dos especialistas do *web site*, os

Tabela 5.2 - Padrões sequenciais detectados nos *logs* do *web site* tempo.cptec.inpe.br.

Prioridade	**Padrão**
1	Condições de Tempo,Máximas,Mínimas,Satélite
2	Análise Sinótica,Máximas,Mínimas
3	Condições de Tempo,Mínimas,Satélite,Sudeste
4	Condições de Tempo,Máximas,Satélite
5	Condições de Tempo,Mínimas,Satélite
6	Análise Sinótica,Máximas,Satélite
7	Análise Sinótica,Condições de Tempo,Satélite
8	Centro-Oeste,Sudeste,Sul
9	Brasil,Condições de Tempo,Satélite
10	Análise Sinótica,Condições de Tempo,Máximas,Mínimas
11	Análise Sinótica,Máximas,Mínimas,Satélite
12	Condições de Tempo,Satélite,setadireita(imagem)
13	Análise Sinótica,Condições de Tempo,Máximas
14	Análise Sinótica,Máximas,Mínimas,Sudeste
15	Análise Sinótica,Condições de Tempo,Máximas,Sudeste

padrões detectados estão relacionados ao comportamento de pesquisadores.

Tabela 5.3 - Ações para adaptação no *web site* tempo.cptec.inpe.br.

#	**Ação para adaptação**	**Padrões sequenciais extraídos da Tabela 5.2**
1	Recomendar "Faça sua Análise Sinótica"	Análise Sinótica, Boletim Técnico
2	Recomendar "Previsão para semi-árido brasileiro"	Condições do Tempo, Centro-Oeste, Norte, Nordeste
3	Recomendar visita ao subdomínio "Dados observacionais"	Análise Sinótica, Condições de Tempo, Mínimas, Satélite, Sudeste
4	Recomendar novo site sobre dados de "Satélite"	Condições de Tempo, Mínimas, Satélite, Sudeste
5	Recomendar subdomínio de "Previsão Numérica do Tempo"	Brasil, Condições de Tempo, Satélite

Uma vez que os padrões foram associados às ações, o TAWS foi novamente implantado no web site para disparar as ações. Em segundo plano, um *script* foi implementado para mensurar a aceitação das adaptações sugeridas ao usuário.

No *web site*, o único esforço de implementação para o desenvolvedor é um *script*

para receber do TAWS os padrões detectados durante a navegação e disparar as adaptações pré-programadas. O *script* implementado no *web site* é exemplificado a seguir, e é invocado pelo TAWS sempre que um padrão é detectado na interação do usuário.

```
function onExecutePattern(pattern){
    if (pattern.action == ‘‘ACTION_1’’){
        //disparar a ação ACTION 1
    }else if (pattern.action == ‘‘ACTION_2’’){
        //disparar a ação ACTION 2
    }
}
```

Na segunda fase do estudo, foram coletadas 86.070 interações e mais de 3 milhões de ações dos usuários. Durante as interações, o TAWS operou com todos os módulos, desde a coleta até a detecção de padrões selecionados no Repositório de Conhecimento. As adaptações foram disparadas à medida que os padrões foram detectados. A Tabela 5.4 mostra os resultados para cada adaptação sugerida, detalhando a quantidade de vezes que apareceu para os usuários e a quantidade de vezes que foi clicada. A razão entre a quantidade de cliques e a quantidade de visualizações foi chamada de taxa de aceitação do usuário. Observa-se que o padrão referente à Previsão Numérica foi o mais frequente, com 164 ocorrências. No entanto, a ação que teve maior aceitação dos usuários foi a ação referente à página de Satélite, com 38,5% de aceitação.

Tabela 5.4 - Resultados das adaptações.

Ação para adaptação	**Exibições**	**Cliques**	**Taxa de aceitação**
1 (Análise Sinótica)	82	21	26,0%
2 (Semi-árido)	27	4	13,7%
3 (Dados observacionais)	51	1	1,6%
4 (Satélite)	142	55	**38,5**%
5 (Previsão Numérica)	**164**	33	20,1%
Total	466	114	24%

5.3.1 Discussão

O estudo de caso do *web site tempo.cptec.inpe.br* é considerado satisfatório para análise da detecção de padrões durante a navegação. Observa-se nos resultados que as adaptações foram exibidas 466 vezes (0,5% em relação ao número de interações do período avaliado). É importante destacar que as adaptações selecionadas pelos

especialistas foram destinadas a perfis específicos de pesquisadores, que não representam a maior parte dos usuários do site. No mesmo período, por exemplo, 7% dos usuários (6182) acessaram o *web site* para buscar a previsão do tempo para uma cidade específica, através do formulário de busca. Portanto, observa-se que há um volume significativamente maior de cidadãos interessados no site do que pesquisadores. Segundo os especialistas, um fator que motiva os pesquisadores a acessarem o *web site* é a previsão de ocorrência de um evento meteorológico específico, fato que não ocorreu no período avaliado.

Os resultados do estudo de caso mostram que a abordagem RUM é capaz de detectar padrões durante a navegação. Considerando as adaptações sugeridas pelos especialistas, há melhorias a serem feitas. As hipóteses são de que os padrões associados pelos especialistas às ações precisam ser refinados. No entanto, o resultado mostra que 114 usuários (24%) aceitaram as recomendações durante a navegação. Isso mostra um potencial do uso da abordagem RUM no *web site* do CPTEC/INPE para atrair e reter visitantes, divulgando as pesquisas e resultados produzidos pelo instituto.

Os três estudos de caso apresentados neste Capítulo foram conduzidos pelo próprio autor desta tese. Como um dos propósitos da abordagem RUM é a capacidade de ser implantada em qualquer aplicação Web e o objetivo do *toolkit* é reduzir o esforço dos desenvolvedores, houve a necessidade de realizar um estudo para verificar a aceitação de ambos por desenvolvedores externos. O experimento é discutido na Seção 5.4.

5.4 Análise do uso da abordagem e do *toolkit* por desenvolvedores externos

A Web tem proporcionado a colaboração entre desenvolvedores de software em todo o mundo, facilitando o compartilhamento de conceitos, ideias e dos próprios códigos-fonte, através de repositórios como o GitHub (DABBISH et al., 2012). Assim, tornou-se comum entre os desenvolvedores a criação de bibliotecas, ferramentas e *frameworks*, que podem ser implantados facilmente nas aplicações. O *toolkit* TAWS adequa-se a esse cenário, visando à fácil implantação em qualquer aplicação Web.

No entanto, devido às diferentes experiências dos desenvolvedores de software, tornou-se necessária a participação de desenvolvedores externos nesta pesquisa para fornecerem suas percepções quanto à relevância da abordagem RUM e quanto ao uso do *TAWS* em suas aplicações.

Dois desenvolvedores foram recrutados para este estudo de caso, nomeados aqui como *desenvolvedor A* e *desenvolvedor B*. Ambos os desenvolvedores possuem experiência em diferentes tecnologias de desenvolvimento Web, sendo que o desenvolvedor A possui aproximadamente cinco anos de experiência com desenvolvimento de software, e o desenvolvedor B atua há dez anos nessa área.

Os desenvolvedores receberam os Apêndices desta tese, que orientam como implantar o *toolkit* na aplicação Web e como usar a biblioteca *jUsabilics*. No entanto, no Apêndice , os desenvolvedores não necessitaram seguir as orientações da Seção A.1, que trata da implantação da arquitetura do *toolkit* em um servidor. Ou seja, a arquitetura do *toolkit* foi implantada em um servidor previamente, pois a arquitetura interna é transparente à aplicação Web.

Sendo assim, os desenvolvedores necessitaram executar os seguintes passos:

a) Na aplicação Web, inserir o *script* de coleta de *logs*, que insere automaticamente a biblioteca *jUsabilics*;

b) Observar os padrões detectados pelo *toolkit*;

c) Associar os padrões detectados a perfis de usuários; e

d) Usar a biblioteca *jUsabilics* para consumir informações sobre o comportamento do usuário.

Após isso, os desenvolvedores relataram suas experiências destacando os seguintes aspectos:

a) **Mecanismo de coleta:** facilidade de implantação e performance;

b) **Qualidade dos padrões detectados:** a relação entre a expectativa dos desenvolvedores e os padrões detectados;

c) **Associação dos padrões a perfis de usuários:** esforço necessário;

d) **Uso da biblioteca *jUsabilics*:** facilidade de uso e recursos disponíveis; e

e) **Curva de aprendizado:** esforço para aprender o funcionamento do *toolkit* e para construir adaptações na aplicação Web.

As Seções 5.4.1 e 5.4.2 descrevem os cenários em que os desenvolvedores realizaram o estudo de caso e, em seguida, são apresentadas suas percepções.

5.4.1 Relato do desenvolvedor A

O desenvolvedor A está desenvolvendo uma pesquisa de Mestrado como consequência desta tese. O propósito de sua pesquisa é detectar padrões de comportamento de usuários idosos nos *logs*, para identificar usuários com comportamento semelhante ao deles durante a navegação, no intuito de disparar adaptações na interface que os auxiliem.

O desenvolvedor implementou uma aplicação Web com algumas funcionalidades de um *Internet Banking*, a fim de disponibilizá-la aos diferentes usuários recrutados em sua pesquisa. Após usar o *toolkit* TAWS, estas foram as percepções do desenvolvedor sobre cada aspecto da abordagem RUM:

a) **Mecanismo de coleta:** o mecanismo de coleta não interferiu na performance da aplicação Web, porém houve a necessidade de corrigir um problema de conflito com a biblioteca *jQuery* já usada na aplicação;

b) **Qualidade dos padrões detectados:** O desenvolvedor julgou os padrões como relevantes. No entanto, no cenário analisado em sua pesquisa, cujo interesse são principalmente os atributos de tempo, quantidade de ações e cliques, os padrões referentes a caminhos do usuário não pareceram relevantes. Ele informou que esperava encontrar padrões mais resumidos em termos da aplicação. Para isso, sugeriu que o *toolkit* permita que o especialista da aplicação selecione os atributos para mineração;

c) **Associação dos padrões a perfis de usuários:** O desenvolvedor relatou que não enfrentou dificuldades neste passo;

d) **Uso da biblioteca *jUsabilics*:** O desenvolvedor reconheceu a facilidade em consumir os padrões detectados durante a navegação. Porém, quanto a consultas de eventos referentes a elementos da página, sua expectativa era que as consultas fossem mais transparentes, o que o levou a dedicar tempo em compreender o funcionamento da biblioteca para isso. Ele sugeriu que haja uma consulta para consultar todos os dados quantitativos da interação resumidamente;

e) **Curva de aprendizado:** por se tratar de uma pesquisa de Mestrado, o desenvolvedor necessitou compreender o processo automatizado de KDD,

pois não tinha experiência prévia com mineração de dados e, por isso, não estava seguro de que os padrões fornecidos pelo TAWS o auxiliaríam. Sendo assim, houve uma reunião para explicar o processo de KDD e, após isso, o desenvolvedor reconheceu a facilidade de consumir os padrões do *toolkit*.

5.4.2 Relato do desenvolvedor B

O desenvolvedor B implantou o TAWS em um *web site* de comércio eletrônico de artigos religiosos. O *web site* atende clientes em todo o Brasil e é acessado por mais de 6000 usuários mensalmente. Sua expectativa era encontrar padrões de comportamento de clientes, para identificá-los durante a navegação e disparar adaptações.

Em um período de duas semanas, o desenvolvedor utilizou o TAWS em sua aplicação e, após isso, relatou estas percepções sobre cada aspecto da abordagem RUM:

a) **Mecanismo de coleta:** O desenvolvedor informou que a implantação é fácil por seguir o mesmo formato de bibliotecas populares disponíveis na Web. Ele também reconheceu que esse passo é suficiente para iniciar a coleta das ações dos usuários.

b) **Qualidade dos padrões detectados:** O desenvolvedor reconheceu a relevância dos padrões, porém destacou que há padrões muito semelhantes, o que dificulta a interpretação em alguns casos.

c) **Associação dos padrões a perfis de usuários:** Para o desenvolvedor, essa foi a etapa mais difícil devido ao grande número de padrões detectados em sua aplicação e à semelhança entre os padrões. Ele definiu três perfis no contexto de sua aplicação: Comprador, Cliente em potencial e Visitante rápido.

d) **Uso da biblioteca *jUsabilics*:** Neste aspecto, o desenvolvedor conseguiu implementar facilmente um *script* para receber os perfis detectados durante a navegação. Ele relatou ainda que a biblioteca atendeu a maioria das informações que gostaria de requisitar sobre o usuário. No entanto, semelhantemente ao desenvolvedor A, este desenvolvedor sugeriu a criação de uma consulta para retornar um resumo sobre todos os dados da interação.

e) **Curva de aprendizado:** O desenvolvedor compreendeu o funcionamento do TAWS rapidamente e considerou que o esforço maior foi direcionado

à implementação das adaptações para cada perfil de usuário. Ele ainda observou que é necessário ter consciência de que as informações fornecidas pelo *toolkit* levam a mudanças na interface da aplicação para beneficiar os usuários.

5.4.3 Discussão

Diante dos relatos dos desenvolvedores externos, observa-se que a implantação do TAWS na aplicação Web é a tarefa mais simples, por se tratar da inserção de apenas um bloco de código nas páginas da aplicação Web. Geralmente, as tecnologias de desenvolvimento Web possuem algum mecanismo para incorporar facilmente arquivos externos, de forma que é necessário inserir o bloco de código em um único arquivo.

Os desenvolvedores utilizaram o *toolkit* para propósitos distintos. O desenvolvedor A tinha a expectativa de compreender, para fins acadêmicos, como os resultados do processo de KDD poderiam ajudá-lo a identificar usuários idosos, que geralmente executam tarefas em maior tempo e rolam a tela diferentemente de outros usuários. Já o desenvolvedor B buscou, de um ponto de vista mercadológico, encontrar padrões que o levassem a compreender o comportamento dos usuários no *e-commerce*, a fim de oferecer alguma adaptação durante a interação para cada perfil de usuário. A interpretação do desenvolvedor B sobre os padrões levou-o a reconhecer três perfis de usuários.

Observando os padrões selecionados pelo desenvolvedor B, percebe-se que o perfil *Comprador* foi identificado usando o conjunto do KDD que considera as ações em cada página (*ProfileRulesByPage*). Especificamente para esse perfil, foram selecionados padrões referentes a ações na página de categoria de produtos. O perfil *Cliente em potencial* foi reconhecido pelo desenvolvedor a partir de um caminho entre os *links* do *web site*. Já o perfil *Visitante rápido* considera a quantidade de cliques, o intervalo médio de rolagem da tela e a quantidade de ações realizadas, que pertencem ao conjunto *ProfileRules* do KDD.

Quanto à tarefa de associar os padrões a perfis de usuários, os desenvolvedores discordaram em suas percepções, sendo que o desenvolvedor B informou que essa foi a tarefa mais trabalhosa devido à semelhança entre os padrões. Essa observação indica que a filtragem de padrões aplicada no *toolkit* ainda não é eficaz para diferentes aplicações Web.

Ambos os desenvolvedores reconheceram que o *toolkit* reduz o esforço de imple-

mentação quanto à análise do comportamento do usuário e a detecção de padrões relevantes. Também observaram que a biblioteca *jUsabilics* colabora nesse processo. Entretanto, ambos sentiram falta de consultas específicas para suas necessidades e, especialmente, de uma consulta para retornar resumidamente os dados da interação.

Quanto à curva de aprendizado, o desenvolvedor A encontrou dificuldades em compreender como o *toolkit* poderia acelerar o desenvolvimento de sua pesquisa. Percebe-se que tal dificuldade foi decorrente da necessidade de extensão do *toolkit* para inserir atributos customizados, como a idade dos usuários, por exemplo. Em contrapartida, o desenvolvedor B, que apenas usou o TAWS sem a necessidade de estendê-lo, afirmou que concentrou seu esforço de implementação nas adaptações da interface. Essa percepção reforça a contribuição da abordagem RUM para apoiar a construção de aplicações Web adaptativas. No entanto, uma extensão importante desta pesquisa é a construção de um método para facilitar a implementação das adaptações pelo desenvolvedor.

No Capítulo seguinte, são apresentadas as limitações observadas a partir das percepções dos desenvolvedores que participaram neste estudo de caso e limitações decorrentes dos desafios enfrentados durante a pesquisa.

5.5 Considerações finais

Neste capítulo, foram apresentados quatro estudos de caso que, de forma complementar, auxiliaram a avaliação da abordagem RUM em aplicações Web reais. Cada estudo de caso utilizou *web sites* distintos, tanto do ponto de vista de interação quanto de conteúdo. O primeiro passo foi investigar a possibilidade de processar *logs* de cliente durante a navegação. Em seguida, o objetivo foi realizar durante a navegação a avaliação de usabilidade baseada em tarefas. O terceiro estudo demonstrou a capacidade da RUM em detectar padrões de comportamento do usuário enquanto ele navega. Por fim, através do último estudo, foi possível observar a aceitação de desenvolvedores externos quanto à abordagem e quanto ao uso do *toolkit*.

Complementando a discussão desta tese, o Capítulo 6 apresenta as conclusões, os resultados obtidos, as limitações e as perspectivas de trabalhos futuros.

6 CONCLUSÕES

Na Web, a análise do comportamento do usuário através de *logs* tem se tornado cada vez mais importante devido à necessidade de construir aplicações que se adaptam às necessidades e aos interesses de cada usuário. A motivação para isso se resume em uma pergunta: "se as pessoas são diferentes, por que se faz o mesmo *web site* para todas as pessoas?". Atualmente, as pessoas querem a informação certa no lugar certo e na hora certa.

Neste sentido, há diversas abordagens na literatura para o processamento de *logs* visando à análise do comportamento do usuário e a construção de aplicações Web adaptativas. No entanto, a maioria das abordagens utiliza *logs* de servidor, que não contemplam todas as ações da interação, ignorando informações relevantes sobre o usuário. Para suprir isso, a abordagem proposta nesta tese processa *logs* de cliente e apoia a construção de aplicações Web adaptativas, reduzindo o esforço de implementação através de um *toolkit*.

A essência da abordagem RUM é a mineração de *logs* de cliente para extração de padrões de comportamento. Tais padrões são analisados pelo especialista da aplicação, a fim de selecionar os padrões relevantes para caracterização de perfis de usuários. Os padrões selecionados pelo especialista são comparados durante a navegação com as ações do usuário e, quando detectados, são disparadas adaptações pré-programadas pelo desenvolvedor.

Além disso, a RUM implementa um método de avaliação remota e automática de usabilidade que auxilia a descoberta de dificuldades de navegação durante a interação do usuário.

Uma contribuição importante desta tese é a capacidade de implantação da abordagem em qualquer aplicação Web, através do *toolkit* TAWS. Assim, qualquer aplicação Web pode tornar-se adaptativa consumindo informações geradas pela RUM durante a navegação do usuário.

6.1 Resultados obtidos

O desenvolvimento da abordagem RUM gerou contribuições relevantes para pesquisas relacionadas à análise do comportamento do usuário e à construção de aplicações Web adaptativas, destacando-se entre elas:

- A avaliação de usabilidade durante a navegação, que contribui para a de-

tecção de problemas de usabilidade enquanto o usuário navega e, portanto, gera a possibilidade de auxiliá-lo e mantê-lo no *web site* (Seção 4.1.2);

- A definição dos conjuntos de atributos para o processo de descoberta de conhecimento em *logs* de cliente, que são aplicáveis a diferentes aplicações Web. Com esses atributos, são contemplados os principais aspectos do comportamento do usuário (Seção 4.1.3);

- O processo automatizado de KDD, que permite aprender o comportamento do usuário com o passar do tempo, detectando novos padrões e descartando os padrões que deixam de ser relevantes (Seção 4.1.3);

- A detecção de padrões durante a navegação, que oferece ao especialista a capacidade de configurar ações de adaptação para serem disparadas durante a interação do usuário. Essas ações podem auxiliar a interação do usuário com recomendações e adaptações projetadas pelo especialista da aplicação (Seção 4.1.3);

- O *toolkit* TAWS, que implementa a abordagem utilizando estratégias para otimizar o processamento nas diversas fases, desde a coleta dos *logs* até a detecção dos padrões durante a navegação (Seção 4.2);

- A biblioteca *jUsabilics*, que reduz o esforço de implementação na aplicação Web para consumir as informações do comportamento do usuário (Seção 4.2.5).

Finalmente, é importante salientar que foram realizados estudos de caso para verificar a eficácia da abordagem RUM; dentre eles, quatro estudos foram apresentados nesta tese. Todos os módulos da RUM foram demonstrados através dos estudos de caso, reforçando o valor das contribuições produzidas (VASCONCELOS et al., 2014; VASCONCELOS et al., 2016).

6.2 Limitações

Durante esta pesquisa, houve diversos desafios concernentes ao processamento de *logs* durante a navegação, à definição de atributos para o processo de KDD, à extração desses atributos dos *logs* e à construção de uma forma simplificada de envolver o especialista da aplicação Web no processo de interpretação dos padrões detectados. Além disso, também surgiram desafios na definição de uma forma de consumir durante a navegação as informações sobre o comportamento do usuário. A solução

desses desafios acarretou em limitações, também observadas pelos desenvolvedores externos que participaram de um dos estudos de caso. Essas limitações são detalhadas a seguir.

O *toolkit* TAWS estabelece uma única estrutura de dados para o armazenamento dos *logs*, definida na forma de grafos. Assim, não há a flexibilidade de explorar outros mecanismos de banco de dados sem o esforço de implementação para mudança da arquitetura do *toolkit*.

Na Seção 4.1.3, a etapa de seleção de atributos do processo de KDD foi detalhada, identificando os atributos que são extraídos dos *logs* para caracterização do comportamento do usuário. A abordagem RUM realiza o processo automatizado de KDD usando esses atributos. Portanto, se o especialista desejar inserir um atributo do modelo de negócio da sua aplicação para associá-lo ao comportamento do usuário, isso exigirá a recuperação dos *logs* com a biblioteca *jUsabilics* e a criação de um banco de dados externo ao *toolkit*. Nesse contexto, haveria um enorme esforço e o único benefício do *toolkit* seria a coleta dos *logs*.

No processo de mineração dos *logs*, a limitação é a implementação de um único algoritmo (*Predictive Apriori*), que restringe os padrões a regras de associação. Caso o especialista tenha conhecimentos de mineração de dados e queira executar outros algoritmos ou usar outras técnicas, isso exigirá o mesmo esforço descrito anteriormente, criar um banco de dados externo ao *toolkit*.

Para a avaliação dos padrões detectados, a abordagem inclui uma aplicação Web em que o especialista visualiza os padrões detectados e associa-os a perfis de usuários ou ações para adaptação. No entanto, a grande quantidade de padrões pode tornar complexa essa atividade, ou confundi-lo, apesar de haver um filtro inicial nos padrões para ocultar aqueles que são óbvios.

Quanto à forma de consumir as informações geradas pelo *toolkit*, a biblioteca *jUsabilics* apresenta uma limitação referente ao encadeamento das requisições enviadas ao módulo de Serviços. Por exemplo, para o desenvolvedor recuperar a duração da interação atual e as últimas páginas visitadas, serão necessárias duas requisições. Essa limitação implica em um código mais complexo nos cenários em que o desenvolvedor não consegue obter as respostas através de um único método da biblioteca.

Considerando essas limitações e o contexto apresentado nesta tese, a Seção 6.3 descreve expectativas e sugestões de trabalhos futuros.

6.3 Trabalhos futuros

A fim de motivar a pesquisa na construção de aplicações Web adaptativas e na análise do comportamento do usuário, são apresentadas as seguintes sugestões para trabalhos futuros:

- Estender as funcionalidades do *toolkit* TAWS implementando um *framework* com a possibilidade de implantação de diferentes mecanismos de banco de dados e à implementação de diferentes algoritmos para mineração dos *logs*. Dessa forma, outros pesquisadores poderiam usar o *framework*, por exemplo, para testar seus algoritmos de mineração de dados na análise do comportamento dos usuários, ou ainda testar e validar mecanismos de armazenamento de *logs*;

- Projetar e implementar uma estrutura de dados específica para *logs* de cliente de aplicações Web. O armazenamento e a manipulação dos *logs* podem ser aprimorados na arquitetura do *toolkit* para viabilizar a operação simultânea em múltiplas aplicações Web com milhares de usuários. Portanto, há o desafio de desenvolver um mecanismo de armazenamento e manipulação de *logs* de acordo com a estrutura dos dados de *logs* de cliente, a fim de otimizar a performance para inserção e seleção dos dados;

- Construir uma linguagem de consulta a *logs* de cliente com o objetivo de estender e flexibilizar as requisições da biblioteca *jUsabilics*. Buscar dados em *logs* é necessário em diferentes cenários, principalmente para fins de segurança, e há diferentes linguagens para consulta a *logs*. Já no cenário de análise do comportamento de usuários, a empresa CoolaData (COOLADATA, 2017) desenvolveu a linguagem CQL (em inglês, *CoolaData Query Language*), mas não especificamente para *logs* de cliente. Portanto, há o desafio para desenvolver uma DSL (*Domain-Specific Language*) para esse tipo de *log* e, posteriormente, uma linguagem de consulta;

- Explorar nos *logs* de cliente o conteúdo das páginas e dos elementos da interface, a fim de coletar dados para a Mineração de Conteúdo da Web;

- Usar a abordagem RUM para explorar a predição de ações do usuário para recomendar páginas relevantes, integrando a Mineração de Uso da Web com a Mineração de Estrutura da Web; e

- Construir um mecanismo para recomendar aos desenvolvedores as

adaptações na interface, no intuito de reduzir o esforço de implementação dedicado nessa etapa da construção de aplicações Web adaptativas.

REFERÊNCIAS BIBLIOGRÁFICAS

ABBAR, S.; AMER-YAHIA, S.; INDYK, P.; MAHABADI, S. Real-time recommendation of diverse related articles. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 22., 2013, New York, NY, USA. **Proceedings...** New York, NY, USA: ACM, 2013. p. 1–12. Disponível em: <http://doi.acm.org/10.1145/2488388.2488390>. 49

AGHABOZORGI, S. R.; WAH, T. Y. Recommender systems: incremental clustering on web log data. In: INTERNATIONAL CONFERENCE ON INTERACTION SCIENCES: INFORMATION TECHNOLOGY, CULTURE AND HUMAN, 2., 2009, New York, NY, USA. **Proceedings...** New York, NY, USA: ACM, 2009. p. 812–818. Disponível em: <http://doi.acm.org/10.1145/1655925.1656073>. 2, 3, 12, 13

AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. Mining association rules between sets of items in large databases. **ACM SIGMOD**, v. 22, n. 2, p. 207–216, jun. 1993. Disponível em: <http://doi.acm.org/10.1145/170036.170072>. 18, 19

AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules in large databases. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 20., 1994, San Francisco, CA, USA. **Proceedings...** San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994. p. 487–499. Disponível em: <http://dl.acm.org/citation.cfm?id=645920.672836>. 19

AKIKI, P. A. Engineering adaptive user interfaces for enterprise applications. In: ACM SIGCHI SYMPOSIUM ON ENGINEERING INTERACTIVE COMPUTING SYSTEMS, 5., 2013, London, United Kingdom. **Proceedings...** New York, NY, USA: ACM, 2013. p. 151–154. Disponível em: <http://doi.acm.org/10.1145/2494603.2480333>. 50

ALQURASHI, T.; WANG, W. A graph based methodology for web structure mining - with a case study on the webs of UK universities. In: INTERNATIONAL CONFERENCE ON WEB INTELLIGENCE, MINING AND SEMANTICS, 4., 2014, Thessaloniki, Greece. **Proceedings...** New York, NY, USA: ACM, 2014. p. 40:1–40:11. Disponível em: <http://doi.acm.org/10.1145/2611040.2611058>. 22

ANAND, S. S.; MOBASHER, B. **Intelligent techniques for web personalization**. Berlin: Springer, 2005. (Lecture Notes in Computer Science, v. 3169). Disponível em: <https://doi.org/10.1007/11577935_1>. 2, 12

APACHE.ORG. **Log files**. 2016. Disponível em: <https://httpd.apache.org/docs/1.3/logs.html>. Acesso em: 18 out. 2016. 9

APAOLAZA, A.; HARPER, S.; JAY, C. Understanding users in the wild. In: INTERNATIONAL CROSS-DISCIPLINARY CONFERENCE ON WEB ACCESSIBILITY, 10., 2013, Rio de Janeiro, Brazil. **Proceedings...** New York, NY, USA: ACM, 2013. p. 13:1–13:4. Disponível em: <http://doi.acm.org/10.1145/2461121.2461133>. 3, 48

ARTHUR, D.; VASSILVITSKII, S. K-means++: the advantages of careful seeding. In: ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS, 18., 2007, New Orleans, Louisiana. **Proceedings...** Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007. p. 1027–1035. Disponível em: <http://dl.acm.org/citation.cfm?id=1283383.1283494>. 18

AZZOPARDI, L.; DOOLAN, M.; GLASSEY, R. Alf: a client side logger and server for capturing user interactions in web applications. In: INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 35., 2012, Portland, Oregon, USA. **Proceedings...** New York, NY, USA: ACM, 2012. p. 1003–1003. Disponível em: <http://doi.acm.org/10.1145/2348283.2348428>. 11, 37

BORDINO, I.; DONATO, D.; POBLETE, B. Extracting interesting association rules from toolbar data. In: ACM INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 21., 2012, Maui, Hawaii, USA. **Proceedings...** New York, NY, USA: ACM, 2012. p. 2543–2546. Disponível em: <http://doi.acm.org/10.1145/2396761.2398687>. 11

BOTHOREL, C.; CHEVALIER, K. How to use enriched browsing context to personalize web site access. In: INTERNATIONAL AND INTERDISCIPLINARY CONFERENCE ON MODELING AND USING CONTEXT, 4., 2003, Stanford, CA, USA. **Proceedings...** Berlin, Heidelberg: Springer-Verlag, 2003. p. 419–426. Disponível em: <http://dl.acm.org/citation.cfm?id=1763142.1763177>. 2, 12, 14

BREIMAN, L.; FRIEDMAN, J. H.; OLSHEN, R. A.; STONE, C. J. **Classification and regression trees**. Florida, USA: Chapman and Hall, 1984. 17

BURKE, R. Hybrid systems for personalized recommendations. In: INTERNATIONAL CONFERENCE ON INTELLIGENT TECHNIQUES FOR WEB PERSONALIZATION, 1., 2003, Acapulco, Mexico. **Proceedings...** Berlin,

Heidelberg: Springer-Verlag, 2003. p. 133–152. Disponível em: <http://dx.doi.org/10.1007/11577935_7>. 2, 12, 14

CARTA, T.; PATERNò, F.; SANTANA, V. Support for remote usability evaluation of web mobile applications. In: ACM INTERNATIONAL CONFERENCE ON DESIGN OF COMMUNICATION, 29., 2011, Pisa, Italy. **Proceedings...** New York, NY, USA: ACM, 2011. p. 129–136. Disponível em: <http://doi.acm.org/10.1145/2038476.2038502>. 1, 12

CERNY, T.; DONAHOO, M. J.; SONG, E. Towards effective adaptive user interfaces design. In: RESEARCH IN ADAPTIVE AND CONVERGENT SYSTEMS, 2013, Montreal, Quebec, Canada. **Proceedings...** New York, NY, USA: ACM, 2013. p. 373–380. Disponível em: <http://doi.acm.org/10.1145/2513228.2513278>. 50

CHANG, G.; HEALEY, M.; MCHUGH, J.; WANG, J. **Mining the World Wide Web**. Dordrecht, the Netherlands: Kluwer Academic Publishers, 2003. 20

CHEN, J.; COOK, T. Mining contiguous sequential patterns from web logs. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 16., 2007, Banff, Alberta, Canada. **Proceedings...** New York, NY, USA: ACM, 2007. p. 1177–1178. Disponível em: <http://doi.acm.org/10.1145/1242572.1242753>. 20

CHEN, L.; BHOWMICK, S.; LI, J. Cowes: clustering web users based on historical web sessions. In: LEE, M.; TAN, K.-L.; WUWONGSE, V. (Ed.). **Database Systems for Advanced Applications**. Berlin: Springer, 2006, (Lecture Notes in Computer Science, v. 3882). p. 541–556. Disponível em: <http://dx.doi.org/10.1007/11733836_38>. 23

CHOI, J.; LEE, G. New techniques for data preprocessing based on usage logs for efficient web user profiling at client side. In: IEEE/WIC/ACM INTERNATIONAL JOINT CONFERENCE ON WEB INTELLIGENCE AND INTELLIGENT AGENT TECHNOLOGY, 3., 2009, Washington, DC, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2009. p. 54–57. Disponível em: <http://dx.doi.org/10.1109/WI-IAT.2009.229>. 1, 2, 11, 12, 13

COOLADATA. **Behavioral data analysis and visualization**. 2017. Disponível em: <http://www.cooladata.com/>. Acesso em: 18 out. 2016. 76

COOLEY, R.; MOBASHER, B.; SRIVASTAVA, J. Web mining: information and pattern discovery on the World Wide Web. In: IEEE INTERNATIONAL

CONFERENCE ON TOOLS WITH ARTIFICIAL INTELLIGENCE, 9., 1997, Newport Beach, CA, USA. **Proceedings...** Washington, USA: IEEE, 1997. p. 558–567. ISSN 1082-3409. 3, 20, 22

_____. Data preparation for mining World Wide Web browsing patterns. **Knowledge and information systems**, New York, NY, USA, v. 1, n. 1, p. 5–32, 1999. 20, 22

DABBISH, L.; STUART, C.; TSAY, J.; HERBSLEB, J. Social coding in github: transparency and collaboration in an open software repository. In: ACM CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK, 2012, Seattle, Washington, USA. **Proceedings...** New York, NY, USA: ACM, 2012. p. 1277–1286. Disponível em: <http://doi.acm.org/10.1145/2145204.2145396>. 67

DUDA, P. H. R.; STORK, D. **Pattern classification**. 2. ed. New Jersey, USA: Wiley-Interscience, 2000. 17

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 2., 1996, Portland, Oregon. **Proceedings...** AAAI Press, 1996. p. 226–231. Disponível em: <http://dl.acm.org/citation.cfm?id=3001460.3001507>. 18

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. The KDD process for extracting useful knowledge from volumes of data. **Communications ACM**, v. 39, n. 11, p. 27–34, nov. 1996. Disponível em: <http://doi.acm.org/10.1145/240455.240464>. 15, 16, 21, 23

FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery: an overview. In: FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P.; UTHURUSAMY, R. (Ed.). **Advances in Knowledge Discovery and Data Mining**. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996. p. 1–34. Disponível em: <http://dl.acm.org/citation.cfm?id=257938.257942>. 18

FRAWLEY, W. J.; PIATETSKY-SHAPIRO, G.; MATHEUS, C. J. Knowledge discovery in databases: an overview. **AI Magazine**, v. 13, n. 3, p. 57–70, set. 1992. Disponível em: <http://dl.acm.org/citation.cfm?id=140629.140633>. 15

FUJIMOTO, H.; ETOH, M.; KINNO, A.; AKINAGA, Y. Web user profiling on proxy logs and its evaluation in personalization. In: ASIA-PACIFIC WEB CONFERENCE ON WEB TECHNOLOGIES AND APPLICATIONS, 13., 2011, Beijing, China. **Proceedings...** Berlin, Heidelberg: Springer-Verlag, 2011. p. 107–118. Disponível em: <http://dl.acm.org/citation.cfm?id=1996794.1996811>. 49

GEEL, M.; CHURCH, T.; NORRIE, M. C. Sift: an end-user tool for gathering web content on the go. In: ACM SYMPOSIUM ON DOCUMENT ENGINEERING, 2012, Paris, France. **Proceedings...** New York, NY, USA: ACM, 2012. p. 181–190. Disponível em: <http://doi.acm.org/10.1145/2361354.2361395>. 1, 2, 11, 12, 13

GÉRY, M.; HADDAD, H. Evaluation of web usage mining approaches for user's next request prediction. In: ACM INTERNATIONAL WORKSHOP ON WEB INFORMATION AND DATA MANAGEMENT, 5., 2003, New Orleans, Louisiana, USA. **Proceedings...** New York, NY, USA: ACM, 2003. p. 74–81. Disponível em: <http://doi.acm.org/10.1145/956699.956716>. 9, 11

GHORAB, M. R.; ZHOU, D.; O'CONNOR, A.; WADE, V. Personalised information retrieval: survey and classification. **User Modeling and User-Adapted Interaction**, v. 23, n. 4, p. 381–443, set. 2013. Disponível em: <http://dx.doi.org/10.1007/s11257-012-9124-1>. 2

GONCALVES, L. F.; VASCONCELOS, L. G.; MUNSON, E. V.; BALDOCHI, L. A. Supporting adaptation of web applications to the mobile environment with automated usability evaluation. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 31., 2016, Pisa, Italy. **Proceedings...** New York, NY, USA: ACM, 2016. p. 787–794. Disponível em: <http://doi.acm.org/10.1145/2851613.2851863>. 1, 6, 11, 12, 28

GOOGLE. **Google analytics**. 2011. Disponível em: <http://www.google.com/analytics>. Acesso em: 18 out. 2016. 11

_____. **Google analytics API**. 2016. Disponível em: <https://developers.google.com/analytics/devguides/reporting/realtime/dimsmets/?hl=pt-br>. Acesso em: 18 out. 2016. 49

GUNDUZ, S.; OZSU, M. A poisson model for user accesses to web pages. In: **Computer and Information Sciences**. Berlin: Springer, 2003, (Lecture Notes

in Computer Science, v. 2869). p. 332–339. Disponível em: <http://dx.doi.org/10.1007/978-3-540-39737-3_42>. 1, 2, 12, 13

HONG, J. I.; HEER, J.; WATERSON, S.; LANDAY, J. A. Webquilt: a proxy-based approach to remote web usability testing. **ACM Transactions on Information Systems**, v. 19, p. 263–285, 2001. Disponível em: <http://doi.acm.org/10.1145/502115.502118>. 11, 12

HUANG, e.; CERCONE, N.; AN, A. Comparison of interestingness functions for learning web usage patterns. In: INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 11., 2002, McLean, Virginia, USA. **Proceedings...** New York, NY, USA: ACM, 2002. p. 617–620. Disponível em: <http://doi.acm.org/10.1145/584792.584896>. 2

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS. **Missão, visão e valores**. 2017. Disponível em: <http://www.inpe.br/institucional/sobre_inpe/missao.php>. Acesso em: 18 out. 2016. 4

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. **ACM Computing Survey**, v. 31, n. 3, p. 264–323, set. 1999. Disponível em: <http://doi.acm.org/10.1145/331499.331504>. 18

JOHN, G. H.; LANGLEY, P. Estimating continuous distributions in bayesian classifiers. In: CONFERENCE ON UNCERTAINTY IN ARTIFICIAL INTELLIGENCE, 11., 1995, Montreal, Canada. **Proceedings...** San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995. p. 338–345. Disponível em: <http://dl.acm.org/citation.cfm?id=2074158.2074196>. 17

KELLER, M.; NUSSBAUMER, M. Menuminer: revealing the information architecture of large web sites by analyzing maximal clicks. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 21., 2012, Lyon, France. **Proceedings...** New York, NY, USA: ACM, 2012. p. 1025–1034. Disponível em: <http://doi.acm.org/10.1145/2187980.2188237>. 22

KHALIL, F. **Combining web data mining techniques for web page access prediction**. 175 p. Tese (Doutorado) — University of Southern Queensland, Australia, 2008. 2, 12, 14, 23

KHASAWNEH, N.; CHAN, C.-C. Active user-based and ontology-based web log data preprocessing for web usage mining. In: IEEE/WIC/ACM

INTERNATIONAL CONFERENCE ON WEB INTELLIGENCE, 2006, Hong Kong, Hong Kong. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2006. p. 325–328. Disponível em: <http://dx.doi.org/10.1109/WI.2006.32>. 11

KHONSHA, S.; SADREDDINI, M. New hybrid web personalization framework. In: INTERNATIONAL CONFERENCE ON COMMUNICATION SOFTWARE AND NETWORKS, 3., 2011, Xian, China. **Proceedings...** Xian, China, 2011. p. 86–92. 4, 49

KHOURY, R.; DAWBORN, T.; HUANG, W. Visualising web browsing data for user behaviour analysis. In: AUSTRALIAN COMPUTER-HUMAN INTERACTION CONFERENCE, 23., 2011, Canberra, Australia. **Proceedings...** New York, NY, USA: ACM, 2011. p. 177–180. Disponível em: <http://doi.acm.org/10.1145/2071536.2071564>. 48

KLEFTODIMOS, A.; EVANGELIDIS, G. An overview of web mining in education. In: PANHELLENIC CONFERENCE ON INFORMATICS, 17., 2013, Thessaloniki, Greece. **Proceedings...** New York, NY, USA: ACM, 2013. p. 106–113. Disponível em: <http://doi.acm.org/10.1145/2491845.2491863>. 21

KOTLER, P.; ARMSTRONG, G. **Marketing an introduction**. New Jersey, USA: Prentice Hall, 2000. 1

KUO, Y. H.; CHEN, J. N.; JENG, Y. L.; HUANG, Y. M. Real-time learning behavior mining for e-learning. In: IEEE/WIC/ACM INTERNATIONAL CONFERENCE ON WEB INTELLIGENCE, 2005, Compiegne Cedex, France. **Proceedings...** Washington, USA, 2005. p. 653–656. 3, 48

LI, C.-h.; KIT, C.-c. Web structure mining for usability analysis. In: IEEE/WIC/ACM INTERNATIONAL CONFERENCE ON WEB INTELLIGENCE, 2005, Compiegne Cedex, France. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2005. p. 309–312. Disponível em: <http://dx.doi.org/10.1109/WI.2005.160>. 22

LI, K.; TAKANO, K. Modelling user behaviour on page content and layout in recommender systems. In: BIBA, M.; XHAFA, F. (Ed.). **Learning Structure and Schemas from Documents**. Berlin: Springer, 2011, (Studies in Computational Intelligence, v. 375). p. 289–313. Disponível em: <http://dx.doi.org/10.1007/978-3-642-22913-8_14>. 2, 12, 14

LINOFF, G.; BERRY, M. **Mining the web**. New Jersey, USA: Jon Wiley & Sons, 2001. 20

LIU, H.; HE, J.; GU, Y.; XIONG, H.; DU, X. Detecting and tracking topics and events from web search logs. **ACM Transactions on Information Systems**, v. 30, n. 4, p. 21:1–21:29, nov. 2012. Disponível em: <http://doi.acm.org/10.1145/2382438.2382440>. 1, 2, 12, 13

LIU, Y.; HUANG, X.; AN, A. Personalized recommendation with adaptive mixture of markov models. **Journal of the American Society for Information Science and Technology**, v. 58, n. 12, p. 1851–1870, out. 2007. Disponível em: <http://dx.doi.org/10.1002/asi.v58:12>. 23

LU, Y.; EZEIFE, C. Position coded pre-order linked WAP-tree for web log sequential pattern mining. In: WHANG, K.-Y.; JEON, J.; SHIM, K.; SRIVASTAVA, J. (Ed.). **Advances in Knowledge Discovery and Data Mining**. Berlin: Springer, 2003, (Lecture Notes in Computer Science, v. 2637). p. 337–349. Disponível em: <http://dx.doi.org/10.1007/3-540-36175-8_33>. 23

MASSEGLIA, F.; PONCELET, P.; TEISSEIRE, M. Using data mining techniques on web access logs to dynamically improve hypertext structure. **ACM SIGWEB Newsletter**, v. 8, n. 3, p. 13–19, out. 1999. Disponível em: <http://doi.acm.org/10.1145/951440.951443>. 23

MOBASHER, B.; COOLEY, R.; SRIVASTAVA, J. Creating adaptive web sites through usage-based clustering of URLs. In: WORKSHOP ON KNOWLEDGE AND DATA ENGINEERING EXCHANGE, 2., 1999, Chicago, Illinois, USA. **Proceedings...** Chicago, USA, 1999. p. 19–25. 4, 49

MONGODB.COM. **MongoDB for giant ideas**. 2016. Disponível em: <https://www.mongodb.com/>. Acesso em: 18 out. 2016. 36

NEBELING, M.; SPEICHER, M.; NORRIE, M. W3Touch: metrics-based web page adaptation for touch. In: SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 5., 2013, Paris, France. **Proceedings...** New York, NY, USA: ACM, 2013. p. 2311–2320. Disponível em: <http://doi.acm.org/10.1145/2470654.2481319>. 50

NEO4J.COM. **Neo4J: the world's leading graph database**. 2016. Disponível em: <https://neo4j.com/>. Acesso em: 18 out. 2016. 36

NODEJS.ORG. **NodeJS**. 2016. Disponível em: <https://nodejs.org/>. Acesso em: 18 out. 2016. 35

PAGANELLI, L.; PATERNò, F. Intelligent analysis of user interactions with web applications. In: INTERNATIONAL CONFERENCE ON INTELLIGENT USER INTERFACES, 7., 2002, San Francisco, California, USA. **Proceedings...** ACM, 2002. p. 111–118. Disponível em: <http://doi.acm.org/10.1145/502716.502735>. 1, 12, 28

PATERNò, F. Task models in interactive software systems. In: **Handbook of Software Engineering and knowledge**. [S.l.]: World Scientific Publishing Co, 2002, (Fundamentals, v. 1). p. 99–114. 13

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, Seattle, Washington. **Proceedings...** ACM, 2001. p. 283–284. Disponível em: <http://doi.acm.org/10.1145/634067.634235>. 13

PATIL, U. M.; PATIL, J. B. Web data mining trends and techniques. In: INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTING, COMMUNICATIONS AND INFORMATICS, 2012, Chennai, India. **Proceedings...** New York, NY, USA: ACM, 2012. p. 961–965. Disponível em: <http://doi.acm.org/10.1145/2345396.2345551>. 21

PELLEG, D.; MOORE, A. W. X-means: extending k-means with efficient estimation of the number of clusters. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 17., 2000, San Francisco, CA, USA. **Proceedings...** San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. p. 727–734. Disponível em: <http://dl.acm.org/citation.cfm?id=645529.657808>. 18

PESKA, L.; ECKHARDT, A.; VOJTAS, P. UPComp - a PHP component for recommendation based on user behaviour. In: IEEE/WIC/ACM INTERNATIONAL CONFERENCES ON WEB INTELLIGENCE AND INTELLIGENT AGENT TECHNOLOGY, 3., 2011, Washington, DC, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2011. p. 306–309. Disponível em: <http://dx.doi.org/10.1109/WI-IAT.2011.180>. 48

PIERRAKOS, D.; PALIOURAS, G.; PAPATHEODOROU, C.; SPYROPOULOS, C. D. Web usage mining as a tool for personalization: a survey. **User Modeling**

and User-Adapted Interaction, v. 13, n. 4, p. 311–372, nov. 2003. Disponível em: <http://dx.doi.org/10.1023/A:1026238916441>. 3

QUINCEY, E. d.; OLIVER, H.; KOSTKOVA, P.; JAWAHEER, G.; MADLE, G.; DIALLO, G.; ALEXOPOULOU, D.; SCHROEDER, M.; HABERMANN, B.; KHELIF, K.; JUPP, S.; STEVENS, R. Mining for patterns of semantic link usage: do domain users actually like semantic browsing? In: IEEE/WIC/ACM INTERNATIONAL JOINT CONFERENCES ON WEB INTELLIGENCE AND INTELLIGENT AGENT TECHNOLOGIES, 2009, Washington, DC, USA. **Proceedings...** Washington, DC, USA, 2009. v. 3, p. 50–53. 48

QUINLAN, J. R. Induction of decision trees. **Machine Learning**, v. 1, n. 1, p. 81–106, mar 1986. Disponível em: <https://doi.org/10.1007/BF00116251>. 17

_____. **C4.5: programs for machine learning**. San Mateo, CA, USA: Morgan Kaufmann Publishers, 1993. 302 p. 17

RAMYA, P. T.; SAJEEV, G. P. Building web personalization system with time-driven web usage mining. In: INTERNATIONAL SYMPOSIUM ON WOMEN IN COMPUTING AND INFORMATICS, 3., 2015, Kochi, India. **Proceedings...** New York, NY, USA: ACM, 2015. p. 38–43. Disponível em: <http://doi.acm.org/10.1145/2791405.2791481>. 23, 49

RASMUSSEN, E. Information retrieval. In: FRAKES, W. B.; BAEZA-YATES, R. (Ed.). **Clustering Algorithms**. New Jersey: Prentice-Hall, Inc., 1992. p. 419–442. Disponível em: <http://dl.acm.org/citation.cfm?id=129687.129703>. 18

RATHEE, S.; KAUL, M.; KASHYAP, A. R-apriori: an efficient apriori based algorithm on spark. In: WORKSHOP ON PH.D. WORKSHOP IN INFORMATION AND KNOWLEDGE MANAGEMENT, 8., 2015, Melbourne, Australia. **Proceedings...** New York, NY, USA: ACM, 2015. p. 27–34. Disponível em: <http://doi.acm.org/10.1145/2809890.2809893>. 19

RIVOLLI, A.; MARINHO, D. A.; PANSANATO, L. T. E. WAUTT: a tool for tracking the user interaction in interactive web applications (in portuguese). In: BRAZILIAN SYMPOSIUM ON MULTIMEDIA AND THE WEB, 14., 2008, Vila Velha, Brazil. **Proceedings...** ACM, 2008. p. 179–181. Disponível em: <http://doi.acm.org/10.1145/1809980.1810033>. 11, 12

SANTANA, V. F.; BARANAUSKAS, M. C. C. Summarizing observational client-side data to reveal web usage patterns. In: ACM SYMPOSIUM ON

APPLIED COMPUTING, 25., 2010, Sierre, Switzerland. **Proceedings...** ACM, 2010. p. 1219–1223. Disponível em: <http://doi.acm.org/10.1145/1774088.1774344>. 1, 11, 12

SANTOS, R. D. C.; GREGIO, A. R. A.; RADDICK, J.; VATTKI, V.; SZALAY, A. Analysis of web-related threats in ten years of logs from a scientific portal. **Proceedings of SPIE**, v. 8408, p. 84080H–13, 2012. Disponível em: <http://dx.doi.org/10.1117/12.919545>. 11

SARUKKAI, R. R. Real-time user modeling and prediction: examples from youtube. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 22., 2013, Rio de Janeiro, Brazil. **Proceedings...** Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2013. p. 775–776. Disponível em: <http://dl.acm.org/citation.cfm?id=2487788.2488041>. 3, 48

SCHEFFER, T. Finding association rules that trade support optimally against confidence. In: EUROPEAN CONFERENCE ON PRINCIPLES OF DATA MINING AND KNOWLEDGE DISCOVERY, 5., 2001, Freiburg, Germany. **Proceedings...** London, UK, UK: Springer-Verlag, 2001. p. 424–435. Disponível em: <http://dl.acm.org/citation.cfm?id=645805.670142>. 19, 42

SCHLEGEL, B.; KIEFER, T.; KISSINGER, T.; LEHNER, W. pcapriori: scalable apriori for multiprocessor systems. In: INTERNATIONAL CONFERENCE ON SCIENTIFIC AND STATISTICAL DATABASE MANAGEMENT, 25., 2013, Baltimore, Maryland, USA. **Proceedings...** New York, NY, USA: ACM, 2013. p. 20:1–20:12. Disponível em: <http://doi.acm.org/10.1145/2484838.2484879>. 19

SEN, E.; TOROSLU, I. H.; KARAGOZ, P. Improving the prediction of page access by using semantically enhanced clustering. **Journal of Intelligent Information Systems**, v. 47, n. 1, p. 165–192, aug 2016. Disponível em: <https://doi.org/10.1007/s10844-016-0398-3>. 4, 49

SERDYUKOV, P. Analyzing behavioral data for improving search experience. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 23., 2014, Seoul, Korea. **Proceedings...** Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2014. p. 607–608. Disponível em: <http://dx.doi.org/10.1145/2567948.2578605>. 3, 48

SHIRGAONKAR, S.; RAJKUMAR, T.; SINGH, V. Application of improved apriori in university library. In: INTERNATIONAL CONFERENCE AND WORKSHOP ON EMERGING TRENDS IN TECHNOLOGY, 1., 2010, Mumbai, Maharashtra, India. **Proceedings...** New York, NY, USA: ACM, 2010. p. 535–540. Disponível em: <http://doi.acm.org/10.1145/1741906.1742027>. 19

SPILIOPOULOU, M. Data mining for the web. In: EUROPEAN CONFERENCE ON PRINCIPLES OF DATA MINING AND KNOWLEDGE DISCOVERY, 3., 1999, Prague, Czech Republic. **Proceedings...** London, UK, UK: Springer-Verlag, 1999. p. 588–589. Disponível em: <http://dl.acm.org/citation.cfm?id=645803.669522>. 20

SRIVASTAVA, J.; COOLEY, R.; DESHPANDE, M.; TAN, P.-N. Web usage mining: discovery and applications of usage patterns from web data. **ACM SIGKDD Explorations Newsletter**, v. 1, n. 2, p. 12–23, jan. 2000. Disponível em: <http://doi.acm.org/10.1145/846183.846188>. 21

SRIVASTAVA, T.; DESIKAN, P.; KUMAR, V. Web mining – concepts, applications and research directions. In: CHU, W.; LIN, T. Y. (Ed.). **Foundations and Advances in Data Mining**. Berlin: Springer, 2005. p. 275–307. Disponível em: <http://dx.doi.org/10.1007/11362197_10>. 21

SUDHAMATHY, G. Mining web logs: an automated approach. In: AMRITA ACM-W CELEBRATION ON WOMEN IN COMPUTING IN INDIA, 1., 2010, Coimbatore, India. **Proceedings...** New York, NY, USA: ACM, 2010. p. 57:1–57:4. Disponível em: <http://doi.acm.org/10.1145/1858378.1858435>. 22

THOMAS, P. Using interaction data to explain difficulty navigating online. **ACM Transactions on Web**, v. 8, n. 4, p. 24:1–24:41, nov. 2014. Disponível em: <http://doi.acm.org/10.1145/2656343>. 48, 49

VARGAS, A.; WEFFERS, H.; ROCHA, H. V. da. 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, Eindhoven, Netherlands. **Proceedings...** ACM, 2010. p. 19:1–19:5. Disponível em: <http://doi.acm.org/10.1145/1931344.1931363>. 11, 12, 28

VASCONCELOS, L. G.; BALDOCHI JR., L. A. Usatasker: a task definition tool for supporting the usability evaluation of web applications. In: IADIS

INTERNATIONAL CONFERENCE ON WWW/INTERNET, 2012, Lisboa, Portugal. **Proceedings...** Madri, Spain: IADIS, 2012. p. 307–314. 38, 52

VASCONCELOS, L. G.; BALDOCHI, L. A. Usabilics: avaliação remota de usabilidade e métricas baseadas na análise de tarefas. In: BRAZILIAN SYMPOSIUM ON HUMAN FACTORS IN COMPUTING SYSTEMS AND THE LATIN AMERICAN CONFERENCE ON HUMAN-COMPUTER INTERACTION, 10., 2011, Porto de Galinhas, Brazil. **Proceedings...** Porto Alegre, Brazil, Brazil: Brazilian Computer Society, 2011. p. 303–312. Disponível em: <http://dl.acm.org/citation.cfm?id=2254436.2254488>. 1, 6, 11, 12, 28, 30

VASCONCELOS, L. G.; SANTOS, R. D. C.; BALDOCHI, L. A. Classifying user experience of web applications in real time using client logs. In: INTERNATIONAL CONFERENCE WWW/INTERNET, 13., 2014, Porto, Portugal. **Proceedings...** Porto, Portugal, 2014. p. 11–18. 38, 51, 74

_____. Exploiting client logs to support the construction of adaptive e-commerce applications. In: INTERNATIONAL CONFERENCE ON E-BUSINESS ENGINEERING, 13., 2016, Macau, China. **Proceedings...** Macau, China, 2016. p. 164–169. 6, 51, 74

VELASQUEZ, J.; YASUDA, H.; AOKI, T. Combining the web content and usage mining to understand the visitor behavior in a web site. In: IEEE INTERNATIONAL CONFERENCE ON DATA MINING, 3., 2003, Melbourne, Florida, USA. **Proceedings...** Melbourne, Florida, USA, 2003. p. 669–672. 11

VELASQUEZ, J. D.; PALADE, V. **Adaptive web sites: a knowledge extraction from web data approach**. Amsterdam, Netherlands: IOS Press, 2008. 1, 2, 9, 20, 21, 22

VERMEEREN, A. P.; ATTEMA, J.; AKAR, E.; RIDDER, H. de; DOORN, A. J. von; ERBUG, c.; BERKMAN, A. E.; MAGUIRE, M. C. Usability problem reports for comparative studies: consistency and inspectability. **Human Computer Interaction**, v. 23, n. 4, p. 329–380, 2008. 30

VORA, D.; BOJEWAR, S. Design of a tool using statistical approach for personalization and usability improvement. In: INTERNATIONAL CONFERENCE & WORKSHOP ON EMERGING TRENDS IN TECHNOLOGY, 2011, Mumbai, Maharashtra, India. **Proceedings...** New York,

NY, USA: ACM, 2011. p. 228–229. Disponível em: <http://doi.acm.org/10.1145/1980022.1980074>. 23, 49

WAIKATO, U. **WEKA 3 - data mining software in Java**. 2013. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/>. Acesso em: 18 out. 2016. 35

ZULKERNINE, F.; MARTIN, P.; POWLEY, W.; SOLTANI, S.; MANKOVSKII, S.; ADDLEMAN, M. Capri: a tool for mining complex line patterns in large log data. In: INTERNATIONAL WORKSHOP ON BIG DATA, STREAMS AND HETEROGENEOUS SOURCE MINING: ALGORITHMS, SYSTEMS, PROGRAMMING MODELS AND APPLICATIONS, 2., 2013, Chicago, Illinois, USA. **Proceedings...** New York, NY, USA: ACM, 2013. p. 47–54. Disponível em: <http://doi.acm.org/10.1145/2501221.2501228>. 12

APÊNDICE A - IMPLANTAÇÃO DA ARQUITETURA DO *TOOLKIT* TAWS EM SERVIDORES E IMPLANTAÇÃO NA APLICAÇÃO WEB

O *Toolkit* TAWS pode ser implantado em um único servidor ou em um conjunto de servidores a fim de usufruir dos benefícios de uma arquitetura distribuída. Uma vez que a arquitetura é instalada no servidor, o *toolkit* pode ser implantado em diversas aplicações Web através de um *script* da linguagem JavaScript. A Seção A.1 detalha o procedimento para implantação do TAWS em uma aplicação Web; e a Seção A.2 descreve os passos para implantação da arquitetura em um servidor.

A.1 Implantação do TAWS na aplicação Web

Para exemplificar a implantação do TAWS em uma aplicação Web, considere um *web site* hipotético chamado *www.empresa.com.br*. Estes são os passos para a implantação nesse *web site*.

A.1.1 Administrador do TAWS cadastra a aplicação no sistema de gerenciamento

O administrador insere o *web site* no sistema de gerenciamento (*www.avaliacaodeusabilidade.com.br*) e vinculando-o a uma conta criada para o desenvolvedor/especialista da aplicação.

A.1.2 Especialista acessa a conta no TAWS e copia o *script* de implantação

O especialista da aplicação acessa o sistema de gerenciamento e copia o *script* de implantação do *toolkit*, demonstrado na Figura A.1.

Figura A.1 - Página do especialista da aplicação no sistema de gerenciamento do TAWS

A.1.3 Especialista insere o *script* na aplicação

O *script* deve ser inserido antes do fechamento do corpo da página (tag *</body>*) em todas as páginas em que se deseja analisar o comportamento do usuário. A partir deste momento, o *toolkit* funcionará integralmente na aplicação, coletando as ações dos usuários, processando-as e permitindo o consumo das informações através da biblioteca *jUsabilics*.

O *script* é semelhante a este:

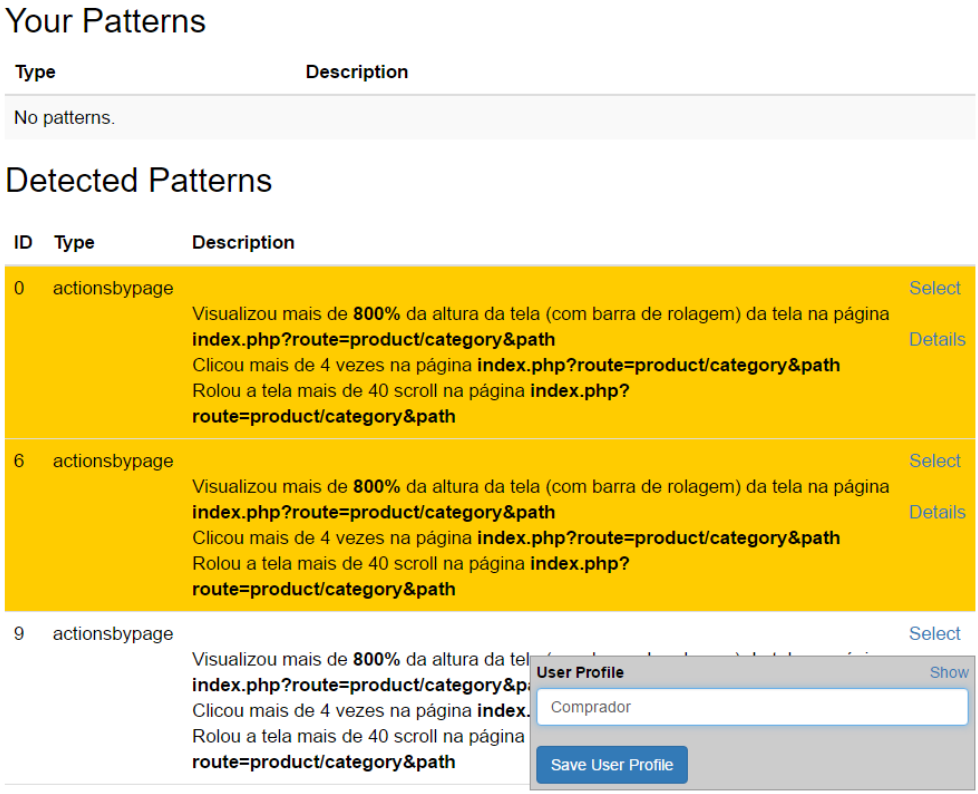
```
<script type="text/javascript" language="javascript">
try { var usabilicsDomainId = "9bf31c7ff062936a96d3c8bd1f8f2ff3"; } catch(error) {}
</script>
<script type="text/javascript" src="http://www.avaliacaodeusabilidade.com.br/rum/usabilics_init.js">
</script>
```

A.1.4 Especialista vincula os padrões detectados a perfis de usuários

Após o primeiro dia de coleta de *logs*, o *toolkit* irá minerá-los para detecção de padrões de comportamento. Esses padrões são disponibilizados ao especialista da aplicação.

No sistema de gerenciamento, na opção *Detected Patterns*, o especialista da aplicação visualiza todos os padrões detectados, podendo criar perfis de usuários (*User Profiles*) associados a um padrão ou a um conjunto de padrões. A Figura A.2 exemplifica a criação de um perfil de usuário **Comprador** cujo comportamento corresponde à combinação dos dois padrões selecionados (destacados em amarelo).

Figura A.2 - Página dos padrões detectados na aplicação Web



Após selecionar os padrões e definir o nome do perfil do usuário, o especialista clica no botão **Save User Profile**.

Automaticamente, o *toolkit* irá avaliar os padrões selecionados durante a navegação dos próximos usuários, a fim de detectar o perfil de cada usuário. Se os padrões ocorrerem, o *toolkit* irá notificar a aplicação Web.

Para isso, um *script* deve ser inserido na aplicação Web para receber as notificações.

A.1.5 Especialista insere o *script* que recebe os padrões detectados durante a navegação

Um arquivo JavaScript pode ser criado para inserir o *script* abaixo, que é responsável por receber do *toolkit* os padrões detectados durante a navegação do usuário.

```
var rumProxyAdaptation=function(pattern){
    switch(pattern.action) {
    case "Comprador":
        //implementar uma ação de adaptação na interface
        break
    }
```

}

O *script* acima verifica se o perfil de usuário **Comprador** foi detectado. Em caso afirmativo, um código por ser implementado para disparar uma adaptação na interface. É importante destacar que o nome do perfil de usuário definido anteriormente pelo especialista da aplicação é o valor retornado pela propriedade *pattern.action*, que é comparado dentro da estrutura *switch...case*.

A.2 Implantação da arquitetura do TAWS em um servidor

Nesta seção, são detalhadas as configurações necessárias para a execução de cada módulo do TAWS, porém não são especificados os comandos para instalação e configuração dos softwares necessários, pois dependem do sistema operacional do servidor.

Os softwares necessários para implantação do *toolkit* são:

- MySQL Server
- Apache Tomcat 7
- NodeJS
- MongoDB
- Neo4J (2 instâncias: *real-time* e *off-line*)

Após a instalação dos softwares, os arquivos de conexão com o banco de dados devem ser alterados de acordo com os dados de acesso (usuário, senha e porta). A estrutura de arquivos do *toolkit* é detalhada a seguir.

- rumserver
  - rum
    - **rum.war:** aplicação Web que gerencia os domínios em que o *toolkit* está implantado, e permite ao especialista associar os padrões detectados com perfis de usuários, além de definir e analisar tarefas.
  - java/rummining
    - dist

            · **rummining.jar:** implementa os algoritmos de mineração de dados.

    – js/datamining

        * **preprocess:** extraem dos *logs* os conjuntos de atributos definidos na Seção 4.2.3.

            · clickedlinks.js, profilerules.js, profilerulesbypage.js

        * **process:** executam os algoritmos de mineração de dados do programa **rummining.war** e armazena os padrões na instância do banco de dados MongoDB.

            · runclickedlinks.js, runprofilerules.js, runprofilerulesbypage.js

    – **mongodbrealtimeconnection.js, neo4jconnection.js, neo4jrealtimeconnection.js:** arquivos de conexão com o banco de dados.

    – **cron.js:** copia os *logs* das interações encerradas no banco de dados *real-time* para o banco de dados *off-line*, e executa diariamente a extração de padrões.

    – **patternAnalysis.js:** realiza a análise de padrões durante a navegação.

    – **taskAnalysis.js:** realiza a análise de tarefas durante a navegação.

    – **rum.js:** processa às requisições da biblioteca *jUsabilics* consultando os *logs*, o resultado da análise de tarefas e o repositório de conhecimento.

    – **rumquery.js:** recebe e responde às requisições da biblioteca *jUsabilics*.

    – **server.js:** recebe as ações do usuário e armazena nos *logs*; e dispara a análise de tarefas e padrões durante a navegação, notificando .

Nas Seções seguintes, são descritos os procedimentos para implantar cada módulo do TAWS.

A.2.1 Inicialização dos servidores de banco de dados

Inicialmente, os servidores de bancos de dados (MySQL, MongoDB e as instâncias Neo4J) devem estar em execução. Por padrão, estas são as portas usadas:

- MySQL: porta 3306

- MongoDB: porta 27017
- Neo4J (instância *real-time*): porta 7475
- Neo4J (instância *off-line*): porta 7474

A.2.2 Implantação da aplicação Web de gerenciamento

Para a implantação da aplicação Web que permite ao especialista gerenciar os domínios, as tarefas e os padrões detectados, é necessário publicar o arquivo **rum.war** no servidor Apache Tomcat 7. Assim, é possível adicionar domínios e definir tarefas. Por padrão, o servidor inicia a aplicação na porta 8080.

A.2.3 Módulo de coleta de *logs*

Uma vez que o domínio é adicionado e o *script* está implantado na aplicação Web para coleta das ações do usuário, conforme descrito na Seção A.1, basta iniciar a aplicação servidora executando o *script* **server.js**. Por padrão, o servidor é executado na porta 8124.

A.2.4 Módulo de definição de tarefas

O módulo de definição de tarefas é fornecido pela aplicação Web de gerenciamento. Este módulo funciona em conjunto com o módulo de coleta de *logs*, portanto ambos os módulos devem estar em execução.

A.2.5 Módulo de análise de tarefas

Uma vez que a aplicação servidora é iniciada com a execução do *script* **server.js**, o módulo de análise de tarefas é iniciado, então nenhuma configuração é necessária.

A.2.6 Módulo de KDD automatizado

A análise de padrões durante a navegação é invocada pelo *server.js* à medida que as interações dos usuários são coletadas. Para habilitar a análise diária dos padrões, o *script* **cron.js** deve ser iniciado. Assim, diariamente, será executado o processo de KDD automatizado através dos *scripts* dos diretórios **preprocess** e **process**.

A.2.7 Módulo de serviços

Para consumir os padrões detectados durante a navegação e demais informações requisitadas pela biblioteca *jUsabilics*, é necessário iniciar o *script* **rumquery.js**,

que iniciará uma nova aplicação servidora. Por padrão, o servidor do módulo de serviços é executado na porta 8125.

A.2.8 Considerações

A arquitetura do *toolkit* pode ser implantada em um único servidor físico ou em um conjunto de servidores. Do ponto de vista de implementação, apenas os arquivos de conexão com os bancos de dados devem ser alterados.

APÊNDICE B - DOCUMENTAÇÃO TÉCNICA DA BIBLIOTECA JUSABILICS

A biblioteca **jUsabilics** é implementada na linguagem JavaScript e realiza requisições assíncronas através de AJAX à aplicação servidora do *toolkit* TAWS. As requisições são transparentes para a aplicação Web, pois são invocadas a partir de simples chamadas de métodos. Para cada requisição, a aplicação servidora retorna à aplicação Web um objeto no formato JSON.

B.1 Requisições da biblioteca jUsabilics

Em seguida, são listadas as requisições da biblioteca jUsabilics, destacando os parâmetros da requisição e os dados retornados pela aplicação servidora do TAWS. Para cada tipo de requisição é apresentado um breve exemplo de uso.

B.1.1 Consultas em nível de domínio

B.1.1.1 getInteractionsByScreenSize()

Retorna as interações ocorridas em janelas com tamanhos específicos.

Parâmetros:

a) minWidth: largura mínima da tela

b) minHeight: altura mínima da tela

c) maxWidth: largura máxima da tela

d) maxHeight: altura máxima da tela

e) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
   {
        id: identificação da interação
        initialTimestamp: tempo inicial da interação
        ip: endereço IP do dispositivo da interação
   },
   {
        id: identificação da interação
        initialTimestamp: tempo inicial da interação
        ip: endereço IP do dispositivo da interação
```

```
    }
]
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "‘mensagem"’
}
```

Exemplo:

Lista as interações realizadas em janelas de 200x300 a 1024x768

```
jUsabilics.getInteractionsByScreenSize(200, 300, 1024, 768, function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
})
```

B.1.1.2 getInteractionsByBrowserAndPlatformAndLanguage()

Retorna as interações ocorridas em um determinado navegador de um sistema operacional em uma língua.

Parâmetros:

a) browser: valores possíveis ’mozilla’, ’chrome’, ’msie’, ’opera’, ’safari’, ’webkit’, ’*’. Se informado o valor ’*’, o parâmetro é desconsiderado.

b) platform: ’win32’, ’win64’, etc..., ’*’. Se informado o valor ’*’, o parâmetro é desconsiderado.

c) language: ’pt-br’, ’en-us’, etc..., ’*’. Se informado o valor ’*’, o parâmetro é desconsiderado.

d) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
    {
         id: identificação da interação
         initialTimestamp: tempo inicial da interação
         ip: endereço IP do dispositivo da interação
    },
    {
```

```
        id: identificação da interação
        initialTimestamp: tempo inicial da interação
        ip: endereço IP do dispositivo da interação
    }
]
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "‘mensagem"’
}
```

Exemplo:

Lista as interações realizadas no navegador 'chrome' na língua 'pt-br', independente do sistema operacional.

```
jUsabilics.getInteractionsByBrowserAndPlatformAndLanguage('chrome',
'*', 'pt-br', function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
})
```

B.1.1.3 getAverageBrowsingTime()

Recupera o tempo médio das interações de um domínio.

Parâmetros:

a) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
{
    averageTime: tempo em milissegundos
}
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "‘mensagem"’
}
```

Exemplo:

```
Exibe o tempo médio das interações do domínio ou uma mensagem de erro.
```

```
jUsabilics.getAverageBrowsingTime(function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data.averageTime)
    }
})
```

B.1.2 Consultas em nível de interação

B.1.3 getElapsedTimeOfInteraction()

Retorna o tempo decorrido da interação atual.

Parâmetros:

a) função de *callback*.

Estrutura do objeto retornado em caso de sucesso:

```
{
    elapsedTime: tempo em milissegundos
}
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "‘mensagem"’
}
```

Exemplo:

Exibe no console do navegador o tempo decorrido da interação ou uma mensagem de erro.

```
jUsabilics.getElapsedTimeOfInteraction(function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data.elapsedTime)
    }
})
```

B.1.4 getLastVisitedPages()

Retorna as N últimas páginas visitadas na interação atual.

Parâmetros:

a) pagesAmount: quantidade de N últimas páginas que deseja recuperar; e

b) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
    lista de N últimas páginas visitadas
]
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "‘mensagem"’
}
```

Exemplo:

Exibe no console do navegador as 10 últimas páginas visitadas ou uma mensagem de erro.

```
jUsabilics.getLastVisitedPages(10, function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
})
```

B.1.5 isVisitedPageLike()

Verifica se uma página já foi visitada na interação atual.

Parâmetros:

a) like: uma URL completa ou parte de uma URL; e

b) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
{
    result: true | false
}
```

Estrutura do objeto retornado em caso de erro:

```
{
```

```
    error: "‘mensagem"’
}
```

Exemplo:

Verifica se alguma URL que contenha a palavra **cursos** foi visitada.

```
jUsabilics.isVisitedPageLike('cursos', function(data){
    if (data.error){
        console.log(data.error)
    }else{
        if (data.result == true){
            console.log('A página de cursos foi visitada')
        }else{
            console.log('Nenhuma página de cursos foi visitada')
        }
    }
})
```

B.1.6 getBrowsingTimeByPage()

Retorna o tempo de navegação em cada página visitada em uma interação.

Parâmetros:

a) page: URL da página ou parte de uma URL

b) idInteraction: id da interação

c) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
    {
        idInteraction: id da interação,
        interaction: {
            objeto do tipo interação
        },
        url: URL da página visitada,
        minTimestamp: primeiro timestamp do usuário na página,
        maxTimestamp: último timestamp do usuário na página,
        time: tempo em segundos de permanência do usuário na página
    },
    {
        ...
    }
]
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "‘mensagem"’
}
```

Exemplo:

```
Lista as páginas e os respectivos tempos de permanência durante a interação com id igual a 123.

jUsabilics.getBrowsingTimeByPage('*', 123, function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
})
```

B.1.7 getEventsOfInteraction()

Retorna os eventos de uma interação.

Parâmetros:

a) idInteraction: id da interação

b) type: tipo de evento do usuário. Se for informado '*', o tipo de evento é ignorado.

c) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
    {
        id: id do evento,
        type: tipo do evento,
        currentURL: página em que ocorreu o evento,
        timestamp: timestamp do momento de ocorrência do evento,
        previousURL: página anterior acessada pelo usuário,
        interactionId: id da interação,
        offSetScrollY: posição vertical da barra de rolagem,
        offSetScrollX: posição horizontal da barra de rolagem,
        browserWidth: largura da janela do navegador,
        browserHeight: altura da janela do navegador,
        objectName: atributo 'name' do objeto manipulado no evento,
        objectTagName: tag HTML do objeto manipulado no evento,
        objectClassName: classe CSS do objeto manipulado no evento,
        objectGeneratedId: identificação do objeto na DOM Tree,
        objectParentId: identificacão do container do objeto em que ocorreu o evento,
        objectValue: conteúdo do objeto manipulado (links, imagens)
    },
```

```
    {
        ...
    }
]
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "‘mensagem"’
}
```

Exemplo:

Lista os eventos do tipo 'click' da interação com id igual a 123, ou exibe uma mensagem de erro.

```
jUsabilics.getEventsOfInteraction(123, 'click', function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
})
```

B.1.8 getLastEventsOfInteraction()

Retorna os últimos N eventos de uma interação.

Parâmetros:

a) idInteraction: id da interação

b) eventsAmount: quantidade de últimos eventos para serem retornados

c) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
    {
        id: id do evento,
        type: tipo do evento,
        currentURL: página em que ocorreu o evento,
        timestamp: timestamp do momento de ocorrência do evento,
        previousURL: página anterior acessada pelo usuário,
        interactionId: id da interação,
        offSetScrollY: posição vertical da barra de rolagem,
        offSetScrollX: posição horizontal da barra de rolagem,
        browserWidth: largura da janela do navegador,
        browserHeight: altura da janela do navegador,
```

```
        objectName: atributo 'name' do objeto manipulado no evento,
        objectTagName: tag HTML do objeto manipulado no evento,
        objectClassName: classe CSS do objeto manipulado no evento,
        objectGeneratedId: identificação do objeto na DOM Tree,
        objectParentId: identificacão do container do objeto em que ocorreu o evento,
        objectValue: conteúdo do objeto manipulado (links, imagens)
    },
    {
        ...
    }
]
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "'mensagem"'
}
```

Exemplo:

Lista os últimos 10 eventos da interação com id igual a 123, ou exibe uma mensagem de erro.

```
jUsabilics.getLastEventsOfInteraction(123, 10, function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
})
```

B.1.9 getEventsOfInteractionByTimestamp()

Retorna os eventos de uma interação dentro de um intervalo de tempo.

Parâmetros:

a) idInteraction: id da interação

b) type: tipo de evento do usuário. Se for informado '*', o tipo de evento é ignorado.

c) initialTimestamp: timestamp inicial do intervalo

d) endTimestamp: timestamp final do intervalo

e) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
    {
        id: id do evento,
        type: tipo do evento,
        currentURL: página em que ocorreu o evento,
        timestamp: timestamp do momento de ocorrência do evento,
        previousURL: página anterior acessada pelo usuário,
        interactionId: id da interação,
        offSetScrollY: posição vertical da barra de rolagem,
        offSetScrollX: posição horizontal da barra de rolagem,
        browserWidth: largura da janela do navegador,
        browserHeight: altura da janela do navegador,
        objectName: atributo 'name' do objeto manipulado no evento,
        objectTagName: tag HTML do objeto manipulado no evento,
        objectClassName: classe CSS do objeto manipulado no evento,
        objectGeneratedId: identificação do objeto na DOM Tree,
        objectParentId: identificacão do container do objeto em que ocorreu o evento,
        objectValue: conteúdo do objeto manipulado (links, imagens)
    },
    {
        ...
    }
]
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "‘mensagem"’
}
```

Exemplo:

Lista todos os eventos da interação com id igual a 123 dentro do intervalo 31/01/2017 15:55 a 18:55, ou exibe uma mensagem de erro.

```
jUsabilics.getEventsOfInteractionByTimestamp(123, 'click', 1485885348000, 1485896148000,
function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
})
```

B.1.10 getEventsByPage()

Retorna os eventos de uma interação que ocorreram em uma página específica.

Parâmetros:

a) page: URL completa ou parte da URL investigada. Se for informado '*', o

parâmetro é ignorado.

b) type: tipo de evento do usuário. Se for informado '*', o tipo de evento é ignorado.

c) idInteraction: id da interação

d) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
    {
        id: id do evento,
        type: tipo do evento,
        currentURL: página em que ocorreu o evento,
        timestamp: timestamp do momento de ocorrência do evento,
        previousURL: página anterior acessada pelo usuário,
        interactionId: id da interação,
        offSetScrollY: posição vertical da barra de rolagem,
        offSetScrollX: posição horizontal da barra de rolagem,
        browserWidth: largura da janela do navegador,
        browserHeight: altura da janela do navegador,
        objectName: atributo 'name' do objeto manipulado no evento,
        objectTagName: tag HTML do objeto manipulado no evento,
        objectClassName: classe CSS do objeto manipulado no evento,
        objectGeneratedId: identificação do objeto na DOM Tree,
        objectParentId: identificacão do container do objeto em que ocorreu o evento,
        objectValue: conteúdo do objeto manipulado (links, imagens)
    },
    {
        ...
    }
]
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "'mensagem"'
}
```

Exemplo:

Lista os cliques da interação com id igual a 123 ocorridos na página '/cursos', ou exibe uma mensagem de erro.

```
jUsabilics.getEventsByPage('/cursos', 'click', 123, function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
```

```
})
```

B.1.11 getEventsInContainerAndPage()

Retorna os eventos de uma interação que ocorreram em um *container* específico de uma página.

Parâmetros:

a) container: identificação do *container* investigado. Exemplos: div.topo (tagName.CSSClassName) ou #divContent (#elementId) ou div#divContent (tagName#elementId)

b) page: URL completa ou parte da URL investigada. Se for informado '*', o parâmetro é ignorado.

c) type: tipo de evento do usuário. Se for informado '*', o tipo de evento é ignorado.

d) idInteraction: id da interação

e) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
    {
        id: id do evento,
        type: tipo do evento,
        currentURL: página em que ocorreu o evento,
        timestamp: timestamp do momento de ocorrência do evento,
        previousURL: página anterior acessada pelo usuário,
        interactionId: id da interação,
        offSetScrollY: posição vertical da barra de rolagem,
        offSetScrollX: posição horizontal da barra de rolagem,
        browserWidth: largura da janela do navegador,
        browserHeight: altura da janela do navegador,
        objectName: atributo 'name' do objeto manipulado no evento,
        objectTagName: tag HTML do objeto manipulado no evento,
        objectClassName: classe CSS do objeto manipulado no evento,
        objectGeneratedId: identificação do objeto na DOM Tree,
        objectParentId: identificacão do container do objeto em que ocorreu o evento,
        objectValue: conteúdo do objeto manipulado (links, imagens)
    },
    {
        ...
    }
]
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "‘mensagem"’
}
```

Exemplo:

Lista os cliques da interação com id igual a 123 ocorridos no *container* com id #content em qualquer página, ou exibe uma mensagem de erro.

```
jUsabilics.getEventsInContainerAndPage('\#content', '*', 'click', 123, function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
})
```

B.1.12 getEventsByObjectAndContainerAndPage()

Retorna os eventos de uma interação que ocorreram em um objeto específico da interface.

Parâmetros:

a) objectId: id do objeto investigado. Se for informado '*', o parâmetro é ignorado.

b) objectName: atributo 'name' do objeto investigado. Se for informado '*', o parâmetro é ignorado.

c) objectTagName: tag HTML do objeto investigado. Se for informado '*', o parâmetro é ignorado.

d) objectClassName: classe CSS do objeto investigado. Se for informado '*', o parâmetro é ignorado.

e) container: identificação do *container* investigado. Exemplos: div.topo (tagName.CSSClassName) ou #divContent (#elementId) ou div#divContent (tagName#elementId)

f) page: URL completa ou parte da URL investigada. Se for informado '*', o parâmetro é ignorado.

g) type: tipo de evento do usuário. Se for informado '*', o tipo de evento é ignorado.

h) idInteraction: id da interação

i) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
    {
        id: id do evento,
        type: tipo do evento,
        currentURL: página em que ocorreu o evento,
        timestamp: timestamp do momento de ocorrência do evento,
        previousURL: página anterior acessada pelo usuário,
        interactionId: id da interação,
        offSetScrollY: posição vertical da barra de rolagem,
        offSetScrollX: posição horizontal da barra de rolagem,
        browserWidth: largura da janela do navegador,
        browserHeight: altura da janela do navegador,
        objectName: atributo 'name' do objeto manipulado no evento,
        objectTagName: tag HTML do objeto manipulado no evento,
        objectClassName: classe CSS do objeto manipulado no evento,
        objectGeneratedId: identificação do objeto na DOM Tree,
        objectParentId: identificacão do container do objeto em que ocorreu o evento,
        objectValue: conteúdo do objeto manipulado (links, imagens)
    },
    {
        ...
    }
]
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "'mensagem"'
}
```

Exemplo:

Lista os cliques da interação com id igual a 123 ocorridos na caixa de entrada com id #cidade, no *container* #content em qualquer página, ou exibe uma mensagem de erro.

```
jUsabilics.getEventsByObjectAndContainerAndPage('cidade', '*',
'input', '*', '\#content', '*', 'click', 123, function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
})
```

B.1.13 getEventsByScroll()

Retorna os eventos de uma interação que ocorreram em uma página específica.

Parâmetros:

a) minScroll: menor posição da barra de rolagem vertical.

b) maxScroll: maior posição da barra de rolagem vertical.

c) page: URL completa ou parte da URL investigada. Se for informado '*', o parâmetro é ignorado.

d) type: tipo de evento do usuário. Se for informado '*', o tipo de evento é ignorado.

e) idInteraction: id da interação

f) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
    {
        id: id do evento,
        type: tipo do evento,
        currentURL: página em que ocorreu o evento,
        timestamp: timestamp do momento de ocorrência do evento,
        previousURL: página anterior acessada pelo usuário,
        interactionId: id da interação,
        offSetScrollY: posição vertical da barra de rolagem,
        offSetScrollX: posição horizontal da barra de rolagem,
        browserWidth: largura da janela do navegador,
        browserHeight: altura da janela do navegador,
        objectName: atributo 'name' do objeto manipulado no evento,
        objectTagName: tag HTML do objeto manipulado no evento,
        objectClassName: classe CSS do objeto manipulado no evento,
        objectGeneratedId: identificação do objeto na DOM Tree,
        objectParentId: identificacão do container do objeto em que ocorreu o evento,
        objectValue: conteúdo do objeto manipulado (links, imagens)
    },
    {
        ...
    }
]
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "'mensagem"'
}
```

Exemplo:

Lista todos os eventos da interação com id igual a 123 ocorridos na página '/cursos' entre 200 e 800 pixels, ou exibe uma mensagem de erro.

```
jUsabilics.getEventsByScroll(200, 800, '/cursos', 'click', 123, function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
})
```

B.1.14 Consultas de avaliação de usabilidade

B.1.15 getWrongActionsByTask()

Recupera as ações erradas detectadas em uma determinada tarefa, de acordo com a análise de tarefas.

Parâmetros:

a) idInteraction: id da interação

b) idTask: id da tarefa definida pelo especialista

c) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
    {
        eventType,
        countOccurrence,
        stepOfTask,
        object,
        container,
        page
    },
    {
        ...
    }
]

{
    error: "'mensagem"'
}
```

Exemplo:

Lista as ações erradas detectadas na tentativa de execução da tarefa com ID 1000 durante a interação com ID 123.

```
jUsabilics.getWrongActionsByTask(123, 1000, function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
})
```

B.1.16 getWrongActionsByObjectAndContainerAndPage()

Recupera as ações erradas detectadas que estão relacionadas a um elemento específico da interface.

Parâmetros:

a) objectId: id do objeto investigado. Se for informado '*', o parâmetro é ignorado.

b) objectName: atributo 'name' do objeto investigado. Se for informado '*', o parâmetro é ignorado.

c) objectTagName: tag HTML do objeto investigado. Se for informado '*', o parâmetro é ignorado.

d) objectClassName: classe CSS do objeto investigado. Se for informado '*', o parâmetro é ignorado.

e) container: identificação do *container* investigado. Exemplos: div.topo (tagName.CSSClassName) ou #divContent (#elementId) ou div#divContent (tagName#elementId)

f) page: URL completa ou parte da URL investigada. Se for informado '*', o parâmetro é ignorado.

g) idInteraction: id da interação

h) idTask: id da tarefa definida pelo especialista

i) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
    {
        eventType: tipo de ação do usuário,
        countOccurrence: quantidade de ocorrências,
        stepOfTask: número do passo da tarefa em que ocorreu,
        object: elemento da interface relacionado ao passo da tarefa,
        container: container do elemento da interface,
        page: página em que ocorreram as ações
    },
    {
        ...
    }
]
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "‘mensagem"’
}
```

Exemplo:

Lista as ações erradas detectadas na página ’/courses’ durante a interação 123 na tentativa de execução da tarefa 1000.

```
jUsabilics.getWrongActionsByObjectAndContainerAndPage(’*’, ’*’, ’*’, ’*’,
’*’, ’/courses’, 123, 1000, function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
})
```

B.1.17 getCurrentUsabilityIndexByTask()

Recupera o índice de usabilidade de uma tarefa durante a interação.

Parâmetros:

a) idInteraction: id da interação

b) idTask: id da tarefa definida pelo especialista

c) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
{
    usabilityIndex: índice de usabilidade entre 0 e 1
```

```
}

{
    error: "‘mensagem"’
}
```

Exemplo:

Recupera o índice de usabilidade da interação 123 na execução da tarefa 1000.

```
jUsabilics.getCurrentUsabilityIndexByTask(123, 1000, function(data){
    if (data.usabilityIndex < 0.5){
        //action to adapt the interface
    }
})
```

B.1.18 Consultas de padrões de comportamento

B.1.19 getCompletedPatterns()

Retorna os padrões de navegação já realizados na interação.

Parâmetros:

a) idInteraction: id da interação

b) função de *callback*

Estrutura do objeto retornado em caso de sucesso:

```
[
    {
        pattern 1
    },
    {
        pattern 2
    }
]
```

Estrutura do objeto retornado em caso de erro:

```
{
    error: "‘mensagem"’
}
```

Exemplo:

```
Lista os padrões já realizados na interação com id 123, ou exibe uma mensagem de erro.
```

```
jUsabilics.getCompletedPatterns(123, function(data){
    if (data.error){
        console.log(data.error)
    }else{
        console.log(data)
    }
})
```

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programas de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. Aceitam-se tanto programas fonte quanto os executáveis.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.