UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

USABILICS: avaliação remota e automática de usabilidade de aplicações Web baseada em um modelo de interface

Leandro Guarino de Vasconcelos

Junho de 2012

Itajubá - MG

2012	**Leandro Guarino de Vasconcelos**	**Dissertação de Mestrado**

UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Leandro Guarino de Vasconcelos

USABILICS: avaliação remota e automática de usabilidade de aplicações Web baseada em um modelo de interface

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciências em Ciência e Tecnologia da Computação

Área de Concentração: Sistemas de Computação

Orientador: Prof. Dr. Laércio Augusto Baldochi Júnior

Junho de 2012

Itajubá - MG

Ficha catalográfica elaborada pela Biblioteca Mauá –
Bibliotecária Cristiane N. C. Carpinteiro- CRB_6/1702

V331u

Vasconcelos, Leandro Guarino de
USABILICS: avaliação remota e automática de usabilidade de aplicações web baseada em um modelo de interface. / por Leandro Guarino de Vasconcelos. -- Itajubá (MG) : [s.n.], 2012.

88 p. : il.

Orientador: Prof. Dr. Laércio Augusto Baldochi Júnior.
Dissertação (Mestrado) – Universidade Federal de Itajubá.

1. Avaliação remota de usabilidade. 2. Modelo de interface. 3. Análise de tarefas. I. Baldochi Júnior, Laércio Augusto, orient. II. Universidade Federal de Itajubá. III. Título.

Dedicatória

Dedico este trabalho a minha família
e, especialmente, a minha esposa

Agradecimentos

Em primeiro lugar, agradeço a Deus por me abençoar desde a concepção deste trabalho até sua conclusão, pois sem a capacidade Dele não seria possível sequer escrever estes agradecimentos.

Agradeço ao professor Dr. Laércio Augusto Baldochi Júnior, que confiou na proposta do trabalho, não mediu esforços durante a orientação desta pesquisa e extrapolou seu horário regular de trabalho buscando a excelência nos resultados. Peço desculpas à esposa e à filha do professor por isso.

Agradeço aos meus pais, Luiz Antonio e Margaret, que investiram em mim e apoiaram-me todos os dias da minha vida. Agradeço ao meu irmão, Luiz Eduardo, pela motivação e pelo apoio para ingressar no mestrado, por todas as ideias sugeridas durante o desenvolvimento do trabalho e pelas conversas durante as viagens a Itajubá. Sou imensamente grato a Deus por fazer parte desta família.

Em especial, agradeço à minha esposa, Tálita, pelo amor, carinho, apoio e incentivo em todos os momentos desta jornada para a conclusão do mestrado. Você é o presente de Deus para mim.

Agradeço também a todas as pessoas da Faculdade de Engenharia de Guaratinguetá - UNESP e da Faculdade de Tecnologia de Guaratinguetá - FATEC pelo apoio durante o mestrado, principalmente à Sra. Regina Célia e à Sra. Deborah Orsi Murgel.

Meu muito obrigado a todos os membros do LUMI (Laboratório de Usabilidade e Mídias Interativas) da UNIFEI, bem como aos professores e colegas do programa de Mestrado em Ciência e Tecnologia da Computação.

Agradeço também aos professores/amigos André Amarante e Samuel Lucena pelo apoio no ingresso do mestrado.

Resumo

Avaliar a usabilidade de aplicações computacionais usando testes tradicionais em laboratório é custoso e consome tempo. Para aplicações Web modernas, geralmente desenvolvidas, testadas e publicadas no *Internet time*, essa abordagem não é viável. Uma maneira mais eficaz para avaliar a usabilidade de aplicações Web consiste em coletar informações sobre as interações dos usuários e processar automaticamente esses dados, a fim de detectar problemas de usabilidade na execução de tarefas pré-definidas. As soluções reportadas baseadas nessa abordagem geralmente falham no fornecimento eficiente de ferramentas para a definição de tarefas, especialmente em aplicações Web grandes e dinâmicas. Para resolver este problema, foi desenvolvido o USABILICS, um sistema destinado à avaliação remota e automática de usabilidade baseada em um modelo de interface. O modelo proposto permite a definição de tarefas usando uma abordagem simples e intuitiva, que pode ser aplicada em aplicações Web grandes e dinâmicas. USABILICS analisa a execução das tarefas calculando a similaridade entre a sequência de eventos produzida pelos usuários e a sequência de eventos definida pelo avaliador. Baseado na análise, o USABILICS fornece um índice de usabilidade, bem como recomendações para solucionar os problemas de usabilidade detectados na execução de cada tarefa.

Palavras-chave: Avaliação remota de usabilidade. Modelo de interface. Análise de tarefas.

Abstract

Evaluating the usability of computer applications using traditional laboratory-based tests is costly and time consuming. For modern Web applications, usually developed, tested and deployed in "Internet Time", this approach is simply not feasible. A more effective way to evaluate the usability of Web applications consists in gathering information from the user's interactions and automatically processing this data in order to detect usability problems in the execution of pre-defined tasks. The reported solutions based on this approach usually fail on providing efficient tools for the definition of tasks, specially in large and dynamic Web applications. In order to tackle this problem, we developed , a system targeted for the automatic remote evaluation of usability based on an interface model. The proposed model allows the definition of tasks using a simple and intuitive approach, which can be applied to large and dynamic Web applications. USABILICS analyzes the execution of tasks by calculating the similarity among sequence of events produced by users and those previously captured by evaluators. Based on this analysis, USABILICS provides an usability index, as well as recommendations for solving usability problems detected on the execution of each task.

Keywords: Remote usability evaluation. Interface model. Task analysis.

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

A Web vem experimentando um aumento expressivo na última década. Atualmente, publicar informações na Web não é mais um privilégio de especialistas em desenvolvimento de *websites*, pois existem diversas ferramentas que exigem apenas conhecimentos básicos de informática para o seu uso. Assim, muitas aplicações são publicadas na Web sem qualquer revisão ou avaliação de qualidade, resultando, muitas vezes, em interfaces pouco amigáveis que dificultam o acesso dos usuários aos serviços disponíveis.

Devido a essa facilidade de criação e publicação de aplicações Web, muitas vezes são oferecidas diversas aplicações para o mesmo negócio e o resultado disso são usuários mais impacientes e intolerantes. Nielsen e Loranger (2007) afirmam que *"Há dez anos a Web era algo diferente para as pessoas. Hoje ela é uma rotina, é uma ferramenta. Se for de fácil acesso, elas a utilizarão, do contrário, não."* Nielsen e Loranger complementam dizendo que, na Web, os usuários estão menos tolerantes a sites complexos, portanto um projeto falho de interface do usuário equivale a negócios perdidos.

Portanto, na era da Internet, a usabilidade assumiu uma importância muito maior e tem conduzido as regras no comércio eletrônico e na relação entre pessoas e empresas pelo simples fato de que, por exemplo, se o cliente não encontra o produto que está procurando em um *website*, ele não o compra naquele momento, mas provavelmente irá encontrá-lo em outro *website*.

Utilizando sistemas de busca em *websites*, o usuário pode encontrar facilmente diversas fontes de uma única informação e, consequentemente, diversas possibilidades para realizar uma compra, por exemplo. Se o usuário não obtém respostas imediatas na sua busca por uma informação em um *website*, ele simplesmente parte para outro endereço, valorizando o seu tempo.

Uma das razões do crescimento da Web é a disponibilidade de ferramentas voltadas para a criação e publicação de conteúdo. Mais que simplesmente facilitar a criação de *home pages*, essas ferramentas permitem que profissionais sem capacitação adequada no

desenvolvimento de aplicações possam criar e disponibilizar essas aplicações na Web.

As estatísticas mostram que mais de 7,8 milhões de *websites* no mundo utilizam um entre os CMS (*Content Management System*) mais utilizados, que são: WordPress [1], Joomla! [2], Drupal [3], Website Tonight [4] e Blogger [5]. Essas ferramentas permitem que os internautas criem e publiquem aplicações Web facilmente (BUILTWITH, 2012).

Neste cenário de diversas ferramentas para publicação de conteúdo na Web, princípios de *design* de interfaces são raramente considerados e testes de usabilidade tradicionais, baseados no uso de avaliadores especialistas e realizados em ambientes controlados, são impraticáveis. Como resultado, são produzidas aplicações que tendem a apresentar problemas de usabilidade.

Para corrigir e/ou aprimorar este cenário, são importantes as pesquisas em avaliação remota e automática ou semi-automática de usabilidade, pois, de acordo com Andreasen et al. (2007), neste tipo de avaliação os usuários e avaliadores estão separados no espaço e no tempo, sendo desnecessário realizar a avaliação em tempo real ou no mesmo local dos usuários, o que permite avaliar um grande número de usuários com um custo baixo. Além disso, a avaliação automática pode ajudar fornecendo informações úteis aos desenvolvedores ou avaliadores inexperientes para identificar problemas nas interfaces.

Por avaliação semi-automática entende-se que a coleta, a análise ou a identificação de problemas de usabilidade não é automatizada. Já na avaliação automática, todas as etapas da avaliação são automatizadas. A forma semi-automática traz benefícios, porém, considerando a rapidez para o desenvolvimento de aplicações atualmente, é necessário haver uma forma rápida e eficaz para se avaliar a usabilidade das interfaces.

As abordagens mais utilizadas para realizar avaliação remota de usabilidade são baseadas no uso de *logs*, os quais podem capturar eventos no servidor ou no cliente (IVORY; HEARST, 2001). Capturar eventos no servidor é mais simples, porém permite registrar apenas a sequência de páginas visitadas, o que não traz informações detalhadas sobre o uso das aplicações. Abordagens mais modernas são baseadas na captura de eventos no cliente, geralmente utilizando aplicações adicionais que executam em segundo plano no navegador (BALBO et al., 2005; ATTERER, 2006; SANTANA; BARANAUSKAS, 2010; VARGAS; WEFFERS; ROCHA, 2010).

[1] http://wordpress.org/
[2] http://www.joomla.org/
[3] http://drupal.org/
[4] http://websitetonight.com/
[5] http://www.blogger.com/

Os *logs* capturados no cliente podem ser processados de diferentes maneiras. Uma abordagem que tem se mostrado eficaz na identificação de problemas de usabilidade consiste em analisar os eventos capturados com base em um modelo de tarefas pré-estabelecido para o domínio de uma aplicação Web.

A comparação entre a sequência de eventos realizada pelo usuário na execução de uma tarefa e a sequência de eventos definida pelo avaliador é capaz de indicar eventuais problemas de usabilidade. Entre os trabalhos que utilizam essa abordagem podem ser destacados WebRemUSINE (PAGANELLI; PATERNò, 2002) e AWUSA (TIEDTKE; MARTIN; GERTH, 2002). Esses trabalhos, entretanto, adotam procedimentos que dificultam sua utilização na avaliação de usabilidade de aplicações Web grandes e dinâmicas, principalmente no que diz respeito à definição das tarefas que devem ser avaliadas.

No sentido de facilitar a avaliação remota de usabilidade de aplicações Web dinâmicas e de grande porte, este trabalho apresenta o modelo de interface COP, que se baseia em três conceitos: *Container*, Objeto e Página. Este modelo de interface é a principal contribuição deste trabalho, pois, a partir dele, é proposta uma abordagem intuitiva para a definição de tarefas e é possível analisar tarefas reais da aplicação Web e identificar pontualmente os elementos da interface que estão relacionados a problemas de usabilidade. Complementando o COP é apresentado o USABILICS, um sistema para avaliação remota e automática de usabilidade de aplicações Web que implementa e estende o modelo de interface proposto.

Visando suprir as deficiências dos métodos de definição e análise de tarefas reportados na literatura, o modelo de interface proposto permite ao avaliador (possivelmente o desenvolvedor da aplicação) definir tarefas de forma simples e intuitiva, capturando suas próprias ações sobre os elementos da interface Web. Além disso, o modelo proposto leva em conta a similaridade dos possíveis caminhos de uma tarefa, permitindo uma abordagem genérica para a definição de caminhos similares, o que poupa um tempo considerável na definição das tarefas de grandes aplicações Web.

A avaliação da usabilidade das tarefas definidas é feita por meio da comparação dos *logs* de eventos produzidos pelos usuários. Esses eventos são capturados por um módulo do sistema USABILICS que é executado no navegador do cliente. Com base no modelo de interface, o USABILICS executa um algoritmo que busca nos dados obtidos as sequências de eventos que correspondam à tentativa de execução de uma tarefa por parte do usuário.

Tendo identificado uma possível tarefa, o algoritmo faz a análise da mesma utilizando a similaridade de caminhos e sequências de eventos. Com base nesse algoritmo e no modelo

de interface, o USABILICS provê uma métrica (índice de usabilidade) que permite avaliar a eficiência e a eficácia de cada tarefa e, consequentemente, da aplicação Web como um todo.

Além do índice de usabilidade, o USABILICS realiza a análise automática de tarefas com o objetivo de identificar problemas de usabilidade e sugerir recomendações pontuais ao desenvolvedor, a fim de que ele possa readaptar a interface da aplicação. Isso é possível devido à flexibilidade do modelo de interface COP. Para identificar problemas de usabilidade, o módulo de análise de tarefas do USABILICS detecta as ações erradas (do inglês, *wrong actions*) definidas por Vermeeren et al. (2008). Assim, o sistema identifica especificamente quais passos da tarefa apresentam problemas de usabilidade.

A validação e os resultados apresentados neste trabalho a partir da avaliação de aplicações Web reais sugerem que o modelo de interface COP e o sistema USABILICS se mostram eficazes na identificação de problemas de usabilidade e na recomendação de sugestões para solucioná-los.

1.1 Objetivos

A fim de contribuir para a pesquisa em avaliação remota e automática de usabilidade e também suprir as deficiências das abordagens reportadas na literatura, este trabalho objetiva desenvolver um sistema para avaliação remota e automática de usabilidade de aplicações Web que forneça informações relevantes para os desenvolvedores de aplicações quanto à usabilidade.

No sentido de alcançar este objetivo geral, os seguintes objetivos específicos foram delineados:

1. permitir a definição de tarefas de forma simples e intuitiva diretamente na interface da aplicação Web, eliminando notações e métodos formais;
2. permitir aos avaliadores/desenvolvedores a fácil manipulação de tarefas de suas aplicações Web;
3. criar uma medida de usabilidade que reflita os aspectos não subjetivos da norma ISO 9241-11 (ISO/IEC, 1998), que são: eficiência e eficácia;
4. coletar minuciosamente os dados das interações dos usuários para permitir a análise de tarefas.

5. identificar problemas de usabilidade em tarefas reais de aplicações Web e sugerir recomendações para a readaptação da interface;

1.2 Metodologia

Considerando as diversas motivações para a avaliação remota de usabilidade, como a redução de custo e a possibilidade de participação de milhares de usuários; e analisando ainda as pesquisas desenvolvidas na área de avaliação remota e automática/semi-automática de usabilidade (BALBO et al., 2005; ATTERER, 2006; SANTANA; BARANAUSKAS, 2010; VARGAS; WEFFERS; ROCHA, 2010; PAGANELLI; PATERNò, 2002; TIEDTKE; MARTIN; GERTH, 2002), constatou-se que há a necessidade de uma abordagem que facilite a avaliação de usabilidade por avaliadores inexperientes e que forneça recomendações aos desenvolvedores para a readaptação da interface, a fim de aprimorar os aspectos da usabilidade da aplicação.

No intuito de suprir essa necessidade, são descritos nesta seção os métodos utilizados para alcançar os objetivos específicos delineados neste trabalho.

Para alcançar o objetivo 1, foi desenvolvido um modelo de interface, o COP, que permite a definição e a análise de tarefas de aplicações Web, baseando-se em três conceitos: *Container*, Objeto e Página. Através desse modelo, foi possível implementar um método que permite ao avaliador definir tarefas diretamente na interface da aplicação, eliminando o uso de notações. Além disso, o método implementado otimiza o tempo e reduz o esforço necessário para definição de tarefas devido à possibilidade de tratar genericamente os elementos da interface.

A fim de contemplar o objetivo 2, foi implementado um módulo no USABILICS denominado UsaTasker, que complementa a definição de tarefas permitindo que o avaliador gerencie as aplicações Web que deseja avaliar e suas respectivas tarefas. O UsaTasker permite ao avaliador adicionar e excluir tarefas, além de permitir editar o caminho ótimo de cada tarefa criada para a aplicação Web.

A fim de alcançar o objetivo 3 e fornecer aos avaliadores inexperientes uma maneira fácil de acompanhar a avaliação de usabilidade de aplicações Web, foi criada uma métrica que reflete a eficiência e a eficácia da usabilidade de interfaces. Essa métrica denomina-se índice de usabilidade e é gerada através da análise de tarefas baseada na comparação entre os eventos das interações dos usuários e os eventos do caminho ótimo da tarefa definida pelo avaliador.

Para suportar a análise de tarefas e atingir o objetivo 4, foi criado um módulo para a coleta automática de dados das interações dos usuários, também baseado no modelo de interface COP. Dessa forma, são coletados todos os atributos HTML (*HyperText Markup Language*) e CSS (*Cascading Style Sheets*) dos elementos da interface. Isso facilita a identificação dos elementos da interface para as recomendações de readaptação a partir da análise de tarefas.

Por fim, para alcançar o objetivo 5, foi inserido um algoritmo na análise de tarefas que identifica as ações erradas (VERMEEREN et al., 2008) das interações dos usuários em relação ao caminho ótimo da tarefa, permitindo que sejam identificados alguns padrões de comportamento em cada passo da tarefa. Além disso, baseado no modelo de interface COP, o algoritmo identifica os elementos da interface que estão relacionados aos problemas de usabilidade identificados, portanto é possível sugerir recomendações específicas para solucioná-los.

O USABILICS, baseado no modelo de interface COP, incorpora os resultados obtidos nos objetivos especificados na Seção 1.1, permitindo a avaliação remota e automática de usabilidade de aplicações Web.

1.3 Estrutura da dissertação

A dissertação está estruturada da seguinte forma:

O Capítulo 2 apresenta um histórico da avaliação de software, os aspectos da qualidade de software e os métodos para a avaliação de usabilidade.

No Capítulo 3, são apresentados os conceitos de análise de tarefas, alguns trabalhos relacionados que usam o método de avaliação baseada em modelo, e o modelo de interface COP, que é a principal contribuição deste trabalho. Este capítulo apresenta também a utilização do COP na definição de tarefas.

O sistema USABILICS é apresentado no Capítulo 4, detalhando sua estrutura e seu funcionamento. Este capítulo apresenta também os trabalhos relacionados ao USABILICS.

O Capítulo 5 apresenta a análise de tarefas do USABILICS, a proposta de um índice que reflete a eficácia e a eficiência da usabilidade e os experimentos realizados para sua validação.

No Capítulo 6, é abordado o método implementado no USABILICS para identificação de problemas de usabilidade e para recomendações automáticas baseadas na análise de

tarefas.

Finalmente, o Capítulo 7 apresenta as considerações finais sobre a dissertação, as principais contribuições do trabalho e sugestões para trabalhos futuros.

2 Avaliação de Usabilidade

A usabilidade é um dos aspectos da qualidade de *software*, portanto é também uma característica essencial para um produto de *software*. Isto é, as interfaces de usuário precisam ser bem projetadas a fim de garantir uma experiência agradável aos usuários e, consequentemente, permitir que eles cumpram seus objetivos durante a interação.

Este capítulo está organizado da seguinte maneira: a Seção 2.1 aborda o histórico da discussão sobre usabilidade na área de *software* e as características da qualidade de *software*; a Seção 2.2 apresenta as principais definições de usabilidade e o desafio de se avaliar a usabilidade de maneira automatizada; a Seção 2.3 aborda os principais métodos de avaliação de usabilidade: avaliação empírica, avaliação formal, avaliação informal e avaliação automática, sendo que a empírica e a automática são de particular interesse deste trabalho. Por fim, a Seção 2.4 apresenta a definição e as características da avaliação remota de usabilidade.

2.1 Avaliação de Software

O *software* tornou-se um componente decisivo para muitos produtos e, consequentemente, para os negócios dos dias atuais, de forma que a qualidade de um *software* é um fator determinante do sucesso ou fracasso de uma atividade de negócio. Mesmo que essa afirmação seja, hoje, um senso comum entre os profissionais da área de tecnologia da informação, tal pensamento foi concebido ainda na década de 80, quando o comitê técnico da ISO/IEC começou a trabalhar para a padronização das características da qualidade de *software* (ABNT, 1996).

Naquela época, a indústria de *software* estava entrando em um período de relativa maturidade e o *software* já estava se tornando decisivo para os negócios. Além disso, começava a surgir uma nova demanda global por segurança e qualidade, gerando a necessidade de acordos internacionais sobre procedimentos para julgamento da qualidade de

software.

Para assegurar a qualidade de um produto, existem basicamente duas soluções: (1) garantir o processo pelo qual o produto é desenvolvido e (2) avaliar a qualidade do produto final. Ambos caminhos são importantes e necessitam de um sistema para gestão da qualidade, estabelecendo políticas e regras para a validação da qualidade de um produto. Para isso, o comitê técnico da ISO/IEC desenvolveu a norma ISO/IEC 9126 (ISO/IEC, 1996), que possui a sua versão brasileira definida pela norma NBR 13596 (ABNT, 1996).

O desenvolvimento da norma ISO/IEC 9126 foi motivado pela ausência de padronização entre os critérios para o julgamento da qualidade de um produto de *software*. Desde 1976, muito trabalho já havia sido desenvolvido para definir estruturas da qualidade de *software*, tais como o Modelo de McCall (PRESSMAN, 2009) e o Modelo de Boehm (BOEHM, 1988); e estes trabalhos foram amplamente utilizados durante anos. Porém, os usuários (entenda-se desenvolvedores e não usuários finais) e consumidores de produtos de *software* ainda encontravam dificuldades para entender ou comparar a qualidade de *software*.

A norma ISO/IEC 9126 começou a ser desenvolvida em 1985, definindo seis características da qualidade de *software*, e foi revisada nos anos seguintes a fim de estabelecer subcaracterísticas em um nível abaixo das seis características definidas no documento inicial.

A partir da norma ISO 9126, a ABNT criou a norma NBR 13596 (ABNT, 1996) motivada, no contexto brasileiro, pela necessidade da indústria de informática do país se atualizar aceleradamente, para sobreviver à passagem da reserva de mercado à livre competição em escala mundial. A NBR 13596 também propiciou a difusão dos conceitos de qualidade de *software*, ampliando a terminologia específica e a própria cultura dessa área entre os profissionais.

Atualmente, as seis características e suas subcaracterísticas são:

- Funcionalidade (Adequação, Acurácia, Interoperabilidade, Conformidade, Segurança de acesso)
- Confiabilidade (Maturidade, Tolerância a falhas, Recuperabilidade)
- Usabilidade (Inteligibilidade, Apreensibilidade, Operacionalidade)
- Eficiência (Comportamento em relação ao tempo, Comportamento em relação aos recursos)

- Manutenibilidade (Analisabilidade, Modificabilidade, Estabilidade, Testabilidade)
- Portabilidade (Adaptabilidade, Capacidade de ser instalado, Conformidade, Capacidade para substituir)

O modelo proposto pela ISO/IEC 9126 (NBR 13596) tem por objetivo servir de referência básica na avaliação de produto de *software*. Além da relevância por ser uma norma internacional, ela cobre os aspectos mais importantes para qualquer produto de *software*. Portanto, todas as características devem ser consideradas para prover a qualidade de *software*.

No entanto, a avaliação da qualidade de *software* segundo as características da norma pode ser suscetível à interpretação de quem irá realizá-la. Por exemplo, como se define especificamente a "Inteligibilidade" de um *software*? Ou, como se pode verificar a "Estabilidade" de um produto de *software*? Para isso, foi agregada à qualidade de *software* uma área de estudo denominada Métricas de *Software*, que se dedica a derivar um valor numérico para algum atributo de um produto ou processo de *software* (SOMMERVILLE, 2010).

Quanto à avaliação da qualidade de *software*, Sommerville (2010) afirma que o ideal é realizar avaliações automatizadas, porém a complexidade para realizar as medições é o principal fator que desfavorece esse tipo de avaliação. Sommerville também afirma que, geralmente, é impossível medir os atributos da qualidade de *software* diretamente, pois alguns atributos como, por exemplo, facilidade de manutenção, facilidade de compreensão e facilidade de uso são atributos externos que se relacionam a como os desenvolvedores e os usuários vêem o *software*. Isto é, esses atributos são influenciados por muitos fatores e não existe uma maneira simples de medi-los. Em vez disso, o autor afirma que é possível medir algum atributo interno de *software* (como seu tamanho, por exemplo) e presumir que há um relacionamento com algum atributo externo. No entanto, a dificuldade nessa abordagem é a validação do relacionamento entre os atributos internos e externos, pois é necessário detectar atributos internos que realmente representem os atributos externos, mesmo que parcialmente.

Neste trabalho, é abordada a característica "usabilidade" da qualidade de *software*, considerando o notório desafio de mensurar a usabilidade de uma aplicação por meio de uma avaliação automatizada. Essa característica está relacionada à interação do usuário final com um produto de *software*.

2.2 Usabilidade

Além das motivações para a criação da ISO 9126 apresentadas na própria norma, a inclusão da usabilidade como característica da qualidade de *software* deve-se ao cenário de desenvolvimento nas décadas de 60 e 70, em que os programas de computador tinham um número reduzido de usuários, mas com um elevado conhecimento técnico, e estes deveriam se adaptar aos sistemas desenvolvidos, o que mostra que nem sempre os desenvolvedores consideraram a usabilidade em suas soluções (MARTINS, 2004).

A norma ISO 9126 foi a primeira a usar o termo "usabilidade", sendo que posteriormente esse termo foi explorado na norma ISO 9241 - parte 11 (ISO/IEC, 1998), que aborda as orientações sobre usabilidade (do inglês, *Guidance on usability*). Na década de 1980, o termo "amigável para o usuário" (do inglês, *user-friendly*) passou a ser adotado na indústria de *software*, porém foi sendo substituído aos poucos pelo termo "usabilidade", porque cada desenvolvedor ou empresa tinha uma visão diferente sobre aquele termo.

A norma ISO 9241 - parte 11 define usabilidade como a capacidade de um produto ser usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso. Essa definição envolve cinco itens:

1. **Usuário:** pessoa que interage com o sistema;
2. **Contexto de uso:** usuários, tarefas, equipamentos (*hardware*, *software* e materiais), ambiente físico e social em que o sistema é usado;
3. **Eficácia:** abrange a precisão e a completude com as quais os usuários atingem os objetivos;
4. **Eficiência:** é a relação entre a eficácia e os recursos gastos para alcançar os objetivos;
5. **Satisfação:** corresponde ao conforto e à aceitação do sistema pelo usuário.

Já Nielsen (2000a) define usabilidade como *"o conceito que busca definir as características da utilização, do desempenho na interação e leitura das e nas interfaces computacionais pelo usuário. As pesquisas em usabilidade sugerem ser esta uma característica fundamental das interfaces desenvolvidas para a internet e utilizadas com muita frequência nas interfaces de intranets, de sites de escolas ou de cursos a distância".*

Nielsen (1993) associa o conceito de usabilidade a cinco atributos que podem ser medidos e estão correlacionados às subcaracterísticas definidas pela ISO 9126: facilidade de aprendizado, eficiência de uso, facilidade de memorização, baixa taxa de erros e satisfação subjetiva.

Sommerville (2010) enfatiza que o projeto de interface de usuário é uma etapa fundamental para o sucesso de um produto de *software*, pois, se um sistema de *software* deve atingir todo o seu potencial, é essencial que sua interface de usuário seja projetada para combinar as habilidades, experiências e expectativas dos futuros usuários. Um bom projeto de interface de usuário é crítico para a confiabilidade do sistema, porque muitos dos chamados "erros do usuário" são causados pelo fato de que as interfaces de usuário não consideram as habilidades dos usuários reais.

Considerando a qualidade de *software* e a necessidade de medir suas características, em especial a usabilidade, a comunidade de desenvolvimento de interface, a partir da metade da década de 80, e enfaticamente nos anos 90, desenvolveu e aplicou métodos de usabilidade para projetar e testar *softwares* e sistemas quanto à facilidade de uso, facilidade de aprendizagem, facilidade de memorização, isenção de erros e satisfação do usuário.

Segundo Bevan, Kirakowski e Maissel (1991), a usabilidade pode ser medida em termos dos atributos ergonômicos dos produtos, ou em função do esforço mental e da atitude do usuário, ou ainda pela observação de como o usuário interage com o produto, enfatizando a facilidade do uso e a aceitabilidade, que representa o desempenho do usuário.

Para Shneiderman e Plaisant (2004), os testes e laboratórios de usabilidade que surgiram desde o início dos anos 80 indicam uma dedicação da atenção dos desenvolvedores às necessidades dos usuários. Porém, os gerentes e desenvolvedores tradicionais resistiram inicialmente, justificando que os recursos e o tempo para o desenvolvimento eram insuficientes para testar a usabilidade. Portanto, observa-se que essa característica da qualidade de *software* tornou-se menos importante que outras características como funcionalidade e eficiência, por exemplo.

No entanto, os desenvolvedores começaram a disputar os recursos escassos dos laboratórios de usabilidade à medida que surgiam projetos bem sucedidos que acreditavam no processo de teste de usabilidade.

Segundo Badre (2002), no fim da década de 90, a usabilidade na Web tornou-se um enfoque específico da comunidade IHC, pelo fato de que os desenvolvedores Web

estavam projetando inadequadamente as interfaces de suas aplicações, novamente devido à inexperiência e ao curto tempo para desenvolvimento.

Devido à importância da usabilidade para a engenharia de *software* e, consequentemente, para o desenvolvimento de *software*, torna-se necessário o projeto de interface centrado no usuário e também torna-se essencial a avaliação da interface projetada. Para isso, existem diversos métodos de avaliação de usabilidade, que estão detalhados na Seção 2.3.

2.3 Métodos de Avaliação de Usabilidade

Segundo Nielsen (1995), existem quatro formas para avaliação de interfaces de usuário, que podem ser utilizadas individualmente ou em conjunto:

1. **Empírica:** realiza testes com usuários reais;
2. **Formal:** usa modelos exatos e fórmulas para calcular as medidas de usabilidade;
3. **Informal:** baseada em "regras de ouro" e na habilidade e experiência dos avaliadores;
4. **Automática:** mede a usabilidade através de algum programa de computador.

Segundo Jordan (1998), cada método para avaliação de interfaces de usuário possui vantagens e desvantagens, portanto é necessário considerar alguns aspectos para a escolha do método adequado, como, por exemplo, o tempo, o esforço, o nível de habilidade e conhecimento para a utilização do método, os recursos necessários e o número mínimo de participantes.

Nas subseções a seguir, são apresentados os métodos mais conhecidos e utilizados de cada forma de avaliação de interfaces.

2.3.1 Métodos de Avaliação Empírica

Os métodos empíricos são a principal maneira de avaliar interfaces de usuário, sendo que são provavelmente a maneira mais usada. No entanto, pode ser difícil e/ou custoso recrutar usuários em número suficiente para testar todos os aspectos de todas as versões de um projeto. Os métodos empíricos mais conhecidos são: Arranjo de cartões (*card sorting*) (FRISONI; STEIL, 2005), Avaliação cooperativa (*cooperative evaluation*) (MIRANDA;

MORAES, 2003), Co-descoberta (*co-discovery*), Grupo focal (*focus group*), Listas de verificação de características (*feature checklists*), Observações de campo (*field observations*), Protocolo "pensar alto" (*think aloud protocol*) e Registro de uso (*logging use*) (JORDAN, 1998).

2.3.2 Métodos de Avaliação Formal

Segundo Dix (1995), as técnicas formais permitem ao projetista de *software* descrever o comportamento externo de um sistema sem especificar a implementação interna. Tais métodos utilizam matemática discreta e notações que objetivam criar ferramentas de uso geral. Porém, segundo Nielsen (1995), os métodos formais são muito difíceis de aplicar e geralmente não se adequam a grandes interfaces. As abordagens mais conhecidas são: Abordagem orientada à propriedade e Abordagem orientada a modelo.

A abordagem orientada a modelo baseia-se na especificação da interface de usuário a partir de estruturas matemáticas como conjuntos e grafos, por exemplo. Os métodos orientados a modelos são usados para construir um modelo comportamental do sistema. Na área de Interação Humano-Computador, os modelos relevantes são: modelo de tarefa, modelo cognitivo, modelo do usuário, modelo de domínio, modelo de contexto, modelo de apresentação e modelo de diálogo (JORDAN, 1998).

2.3.3 Métodos de Avaliação Informal ou Métodos de Inspeção de Usabilidade

Nielsen (1995) aborda os métodos informais como métodos de inspeção de usabilidade, que se mostram muito eficientes na identificação de problemas de usabilidade negligenciados pelos testes empíricos. Porém, os testes empíricos também encontram muitos problemas desconsiderados pelos métodos de inspeção, significando que melhores resultados podem ser encontrados com a combinação de vários métodos. Os métodos de inspeção mais conhecidos são: Avaliação heurística (*heuristic evaluation*) (NIELSEN; MOLICH, 1990), Percurso cognitivo (*cognitive walkthroughs*) (LEWIS et al., 1990), Inspeção formal de usabilidade (*formal usability inspection*) (KAHN; PRAIL, 1994) e Inspeção baseada em padrões (*standards inspection*) (BIAS, 1994).

Dentre os métodos de inspeção de usabilidade, a avaliação heurística é aquele que tem despertado maior interesse por parte dos desenvolvedores, devido à facilidade de aplicação em relação aos outros métodos e à quantidade de trabalhos já desenvolvidos e

bem conhecidos (NIELSEN, 1993; DIAS, 2001; SHNEIDERMAN; PLAISANT, 2004; BASTIEN; SCAPIN, 1993).

2.3.4 Métodos de Avaliação Automática

Em 1995, Nielsen afirmou que não havia métodos de avaliação automática que funcionassem para a identificação de problemas de usabilidade até o momento (NIELSEN, 1995).

Já em 2001, Ivory e Hearst (2001) apresentaram diversos trabalhos desenvolvidos na área de avaliação automática de usabilidade e propuseram uma taxonomia de avaliação automática de usabilidade, considerando que alguns aspectos da avaliação podem ser automatizados. São eles:

- **Captura**: envolve coleta de dados das interações dos usuários em relação ao tempo de conclusão da tarefa, erros, violações de diretrizes e taxas subjetivas;
- **Análise**: abrange a interpretação dos dados para identificar problemas de usabilidade na interface;
- **Crítica**: abrange a sugestão de soluções ou melhorias para solucionar ou minimizar os problemas encontrados.

Ivory e Hearst (2001) afirmam que existem muitas vantagens dos métodos de avaliação automática em relação aos demais métodos, tais como:

- Redução do tempo e, consequentemente, do custo da avaliação, devido à coleta automática de dados por meio de um *software* ao invés da coleta manual.
- Aumento da consistência dos erros descobertos, devido à possibilidade de desenvolver modelos de tarefas em uma interface, a fim de comparar com as interações dos usuários e identificar desvios na realização da tarefa. Além disso, é possível identificar padrões de interação.
- Redução da necessidade de *expertise* entre os avaliadores das interfaces. Isso é possível pela automatização das atividades de análise e crítica, objetivando auxiliar os desenvolvedores das aplicações.

- Aumento da abrangência de características avaliadas. Devido ao tempo, ao custo e às restrições de recursos, nem sempre é possível avaliar todos os aspectos das interfaces pelos métodos não-automáticos.

- Possibilidade de comparação entre projetos alternativos. Com a redução de tempo e custo, é possível avaliar interfaces em um espaço de tempo menor, permitindo a reavaliação após a readaptação de uma interface, por exemplo.

É importante destacar que a avaliação automática não ignora os outros métodos, mas os complementa, com o objetivo de reduzir, principalmente, o tempo e o custo de avaliação.

Devido à possibilidade de automatização de todo o processo de avaliação ou de parte dele somente, Balbo (1995) propôs uma taxonomia que diferencia quatro aspectos para automatização dos métodos de avaliação de usabilidade:

- **Não-automática**: métodos "realizados por especialistas em IHC".

- **Captura automática**: métodos que realizam por meio de *software* a captura dos dados das interações dos usuários, tais como: fala, teclado, mouse, etc.

- **Análise automática**: métodos que identificam problemas de usabilidade automaticamente.

- **Crítica automática**: métodos que não apenas encontram problemas de usabilidade, mas também propõem melhorias.

A proposta de Ivory e Hearst (2001) complementa a proposta de Balbo e agrupa os métodos de avaliação de usabilidade em quatro dimensões:

- **Classe do método:** descreve o tipo de avaliação.

- **Tipo do método:** descreve como a avaliação é conduzida dentro do método, assim como o protocolo "pensar alto" na classe de teste de usabilidade, por exemplo. Os tipos de métodos estão detalhados na publicação dos autores (IVORY; HEARST, 2001).

- **Tipo de automatização:** descreve qual aspecto da avaliação é automatizado (captura, análise ou crítica);

- **Nível de esforço:** descreve o tipo de esforço necessário para executar o método (desenvolvimento de modelo ou uso da interface, por exemplo).

É importante destacar que, dos 57 trabalhos selecionados por Ivory e Hearst (2001), apenas um permite a crítica automática, sendo que o mesmo não permite a captura automática de informações. Isto é, de todos os trabalhos selecionados pelos autores, nenhum abrange os três níveis de automatização: captura, análise e crítica, o que torna notório o desafio de se desenvolver uma abordagem completamente automatizada. O sistema USABILICS, proposto neste trabalho, abrange os três níveis de automatização. A estrutura, o funcionamento e a classificação do sistema em relação às taxonomias apresentadas estão detalhados no Capítulo 4.

Mesmo com a automatização da captura, da análise e da crítica de um método de avaliação de usabilidade, outra característica desejável para a avaliação automática de interfaces é a realização de testes remotos, em que o avaliador e o usuário estão separados no tempo (FIDAS et al., 2007). A avaliação de aplicações *desktop*, que geralmente são construídas para um contexto empresarial específico e não sofrem mudanças frequentes, pode ser feita por métodos tradicionais como os testes em laboratório. Porém, quando se trata de aplicações Web, os métodos tradicionais tornam-se inviáveis.

Segundo Fidas et al. (2007), os *frameworks* de aplicações Web ricas enriquecem ainda mais a interatividade e aceleram o tempo de desenvolvimento, de forma que as mudanças ocorrem com frequência, muitas vezes sem a oportunidade de reavaliar todo o site (BARESI; GARZOTTO; PAOLINI, 2000). Além disso, o público potencial de uma aplicação Web é geograficamente disperso ou abrange uma ampla gama de grupos demográficos. Portanto, os métodos tradicionais de avaliação de usabilidade tendem a ser menos relevantes e financeiramente inviáveis. Dessa forma, a avaliação remota de usabilidade é uma área emergente dentro da área de IHC (FIDAS et al., 2007).

2.4 Avaliação remota de usabilidade

De acordo com Fidas et al. (2007), a avaliação remota de usabilidade é um paradigma eficiente e financeiramente viável para avaliação de sistemas interativos. De maneira geral, as diferenças entre os testes de usabilidade em laboratório e a abordagem de avaliação remota são: (1) o avaliador e os participantes estão em locais diferentes; (2) o avaliador deve usar uma ferramenta de *software* para observar e analisar as interações dos participantes; e (3) a comunicação entre o avaliador e os participantes é mediada por uma ferramenta

de *software*.

Os benefícios da avaliação remota são: (1) facilita o alcance de participantes em diversas áreas geográficas; (2) é útil para avaliar sistemas projetados para grupos descentralizados de usuários; (3) é financeiramente viável devido à redução dos custos de viagens dos avaliadores e participantes; (4) é possível alcançar participantes mundialmente; (5) os participantes podem interagir com a interface no ambiente real; e (6) permite avaliar toda a população de usuários de uma aplicação.

As limitações e desvantagens na avaliação remota são relacionadas à separação entre avaliador e participantes no espaço/tempo: (1) a dificuldade na comunicação devido à limitação dos meios para isso; (2) a dificuldade para capturar expressões faciais dos participantes; e (3) os contextos social e cultural de um grupo internacional de participantes podem influenciar os resultados.

Petrie et al. (2006) citam um conjunto de técnicas de avaliação remota de usabilidade que utilizam desde questionários remotos até *softwares* para coleta automática de dados, passando por conexões de vídeo conferência.

Já Fidas et al. (2007) afirmam que existem duas categorias de técnicas de avaliação remota: a moderada e a automatizada. Essas categorias são classificadas de acordo com a co-presença dos avaliadores e participantes, os canais de comunicação utilizados e as metodologias de usabilidade suportadas.

A categoria moderada exige a comunicação síncrona entre avaliador e participantes, a observação das estações de trabalhos dos participantes, registro da tela, do áudio e dos dados das interações dos participantes.

A categoria automatizada não requer que os participantes e avaliadores estejam no mesmo espaço/tempo, e possuem dois tipos de técnicas de avaliação: testes empíricos e avaliação baseada em modelo. Os testes empíricos se baseiam nas execuções de tarefas reais da interface pelos participantes. A avaliação baseada em modelo é comumente usada para prever certos aspectos do desempenho do usuário, tais como o tempo para completar a tarefa ou a dificuldade para aprender a sequência da tarefa. Para a avaliação remota automatizada, a sequência de ações definida pelo avaliador é comparada à sequência de ações dos participantes, a fim de identificar as diferenças entre as interações (FIDAS et al., 2007).

Essa forma de avaliação é utilizada pelo USABILICS, o que o classifica como uma ferramenta da categoria de avaliação remota automatizada. No entanto, o USABILICS

utiliza tanto testes empíricos quanto a avaliação baseada em modelo. Quanto à classificação de Petrie et al. (2006), o USABILICS utiliza a técnica de avaliação remota de controle com observação assíncrona e a avaliação remota instrumental.

Além das classificações anteriores, os trabalhos publicados por Vargas, Weffers e Rocha (2010) e Andreasen et al. (2007) permitem classificar os métodos de avaliação remota em semi-automáticos e automáticos. Os métodos automáticos realizam a análise e a crítica de forma automática por meio de ferramentas de *software*, enquanto os métodos semi-automáticos necessitam da intervenção humana nessas fases. O USABILICS caracteriza-se como um sistema de avaliação remota e automática de usabilidade, conforme os detalhes apresentados no Capítulo 4.

2.5 Considerações finais

Neste capítulo, foram apresentadas as principais definições de usabilidade após uma breve introdução sobre a avaliação de *software*, e a importância da usabiliade para a qualidade de *software*. Dentro da Engenharia de Software, é possível perceber que há um grande desafio no estudo de métricas de *software*, e esse desafio é ampliado quando se trata dos aspectos que dependem da participação de seres humanos, como é o caso da usabilidade.

No intuito de propor uma alternativa para mensurar a usabilidade, foi desenvolvido um modelo de interface, o COP, que foi implementado e estendido pelo sistema USABILICS. O COP foi criado a partir da observação da estrutura das páginas Web e, portanto, considera os elementos das interfaces de aplicações Web reais.

A implementação do COP no USABILICS permite a realização da avaliação remota e automática de usabilidade de aplicações Web, considerando os conceitos contexto de uso, eficiência e eficácia da ISO 9241:11 (ISO/IEC, 1998), englobando os atributos eficiência de uso e baixa taxa de erros citados por Nielsen (1993). Além disso, dentre as formas de medição da usabilidade citadas por Bevan, Kirakowski e Maissel (1991), o USABILICS utiliza a visão de desempenho do usuário, examinando como o usuário interage com a aplicação Web por meio de *logs* coletados no navegador da máquina cliente.

A fim de definir e analisar tarefas para a avaliação remota e automática de usabilidade, o modelo de interface COP permite a generalização de elementos da interface de uma aplicação Web, facilitando a definição e a análise de tarefas. O Capítulo 3 apresenta o modelo de interface COP, no qual se baseiam todos os módulos do sistema USABILICS.

3 O modelo COP

Este capítulo apresenta o modelo de interface COP, que é utilizado na coleta de dados e na definição e análise de tarefas no USABILICS. Este modelo permite definir tarefas de uma forma mais intuitiva que as formas reportadas na literatura. Além disso, devido à forma como o modelo trata os elementos da interface, é possível definir caminhos alternativos automaticamente para uma mesma tarefa, reduzindo o tempo e o esforço para a definição de tarefas em aplicações Web.

Para compreender a importância do modelo de interface para a avaliação de usabilidade, são tratadas as diferentes técnicas de análise de tarefas e são apresentados alguns trabalhos que utilizam avaliação baseada em modelo. No entanto, observa-se que os trabalhos reportados na literatura geralmente utilizam notações ou métodos formais para a definição de tarefas, que exigem um esforço maior para a aprendizagem pelo avaliador.

O capítulo está organizado da seguinte maneira: a Seção 3.1 apresenta a definição de análise de tarefas e alguns trabalhos da literatura que utilizam avaliação baseada em modelo. A Seção 3.2 descreve o modelo de interface proposto neste trabalho e suas opções de generalização que podem ser utilizadas na definição e na análise de tarefas.

3.1 Análise de tarefas baseada em modelo

Tarefas (do 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 (do 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.

A análise de tarefas é uma técnica usada para descrever e avaliar as atividades requeridas para alcançar um objetivo em um ambiente interativo (ALBERS, 1998). Na análise e no projeto de websites, as tarefas geralmente são associadas a um ou mais perfis de usuários, que representam o público-alvo do website (BOLCHINI; MYLOPOULOS, 2003). Por exemplo, as tarefas "cadastrar uma promoção" e "alterar o status do pedido" podem ser tarefas do usuário administrador de um site de *e-commerce*, enquanto "comprar um produto" pode ser a tarefa de um usuário cliente do mesmo site.

No estudo de avaliação de usabilidade, a análise de tarefas tem sido utilizada na forma de avaliação baseada em modelo, sendo que a abordagem mais comum é a utilização de modelos de tarefas (do inglês, *task models*), conforme constatam os trabalhos desenvolvidos por Lecerof e Paternò (1998), Ivory e Hearst (2001), Paganelli e Paternò (2003), Bolchini e Mylopoulos (2003), Crystal e Ellington (2004), Jokela et al. (2006) e Caffiau et al. (2010).

Caffiau et al. (2010) cita algumas ferramentas baseadas em modelos de tarefas, são elas: *ConcurTaskTree Environment* (CTTE), baseada na notação CTT (PATERNò, 2000); TAMOT, baseada no método Diane+ (TARBY; BARTHET, 1996); EUTERPE, baseada no método GTA (VEER; LENTING; BERGEVOET, 1996); e um ambiente de análise de tarefas denominado AMBOSS (AMBOSS, 2008), que pode ser usado para projetar modelos de tarefas.

3.1.1 ConcurTaskTree Environment CTTE

A *ConcurTaskTree* (CTT) é definida pelos autores como uma notação para especificação de modelos de tarefas para suprir as limitações de notações usadas para criar aplicações interativas. Seu principal objetivo é fornecer uma notação facilmente utilizável que representa diferentes relações temporais no mesmo nível de abstração e que pode ser usada por pessoas inexperientes na modelagem de tarefas. A notação CTT é baseada em uma estrutura hierárquica de tarefas representadas por uma estrutura de árvore. Cada tarefa é apresentada em um formato gráfico de árvore. O CTTE é a ferramenta associada à notação que permite definir tarefas, manipulando dois conceitos: tarefas e objetos. A ferramenta permite ao avaliador vincular os passos das tarefas aos objetos da interface.

3.1.2 TAMOT

A ferramenta TAMOT é associada à metodologia Diane+ e manipula apenas os conceitos de tarefas. Diane+ é uma metodologia completa que tenta sanar a diferença entre engenharia de software e fatores humanos. Assim como a notação CTT, Diane+ centra-se na concepção de aplicações interativas, integrando algumas características de ergonomia e cognitivas, intervindo no desenvolvimento do software. A metodologia Diane+ é composta de um conjunto de operadores e, diferentemente da CTT, as sub-tarefas são representadas por caixas dentro das tarefas de níveis superiores.

3.1.3 EUTERPE

Groupware Task Analysis (GTA) é um método de análise de tarefas destinado para a modelagem de ambientes complexos, nos quais muitas pessoas interagem com os sistemas. Duas ferramentas são associadas a esse método: *GTA Tool* e Euterpe. Esta última ferramenta não inclui qualquer função de programação pré-definida. O método GTA manipula cinco conceitos: tarefas, funções, agentes, objetos e eventos.

3.1.4 AMBOSS

A ferramentas AMBOSS não possui trabalhos publicados quanto ao modelo de tarefas utilizado. Essa ferramenta objetiva auxiliar os projetistas na concepção de seus modelos de tarefas especialmente para sistemas críticos de segurança, fornecendo representações dedicadas a aspectos específicos. Essa ferramenta manipula quatro conceitos: tarefas, objetos, papéis e barreiras. As barreiras são introduzidas para expressar o mecanismo de proteção das tarefas. Por exemplo, para completar uma tarefa de identificação, uma barreira pode representar a condição sobre a validade da senha informada. Durante a simulação da tarefa, o projetista indica se a barreira é desencadeada ou não.

Analisando as notações e os métodos citados anteriormente em conjunto com as definições do Capítulo 2, especialmente da Seção 2.4, é possível refletir sobre possíveis problemas na utilização de tais métodos para a análise de tarefas:

1. Qual o tempo de aprendizagem das notações para avaliadores inexperientes?

2. Como relacionar os elementos da interface do usuário com a representação das notações para a avaliação automática de usabilidade?

3. Em se tratando de aplicações Web grandes e dinâmicas, como é possível vincular elementos da interface localizados em dezenas ou centenas de páginas à representação das notações?

Essas perguntas sugerem um grande desafio para a implementação das notações para a avaliação de aplicações Web, pois é necessário relacionar, com a representação das notações, os dados coletados em termos de elementos da interface (por exemplo *links*, botões, imagens, etc.), além disso, é necessário analisar as tarefas para reconhecer possíveis problemas de usabilidade.

Para Crystal e Ellington (2004), a análise da tarefa é especialmente valiosa no contexto de IHC. As interfaces de usuário devem ser especificadas em um nível extremamente baixo (por exemplo, em termos de estilos de interação) mapeando para tarefas de alto nível. No entanto, interfaces de computador são muitas vezes altamente inflexíveis (quando comparadas à interação com um ambiente físico). Esta inflexibilidade aumenta o impacto dos problemas de *design* de interface, tornando crucial a integração da estrutura de tarefas e o suporte da interface.

Nielsen (1993) observou que muitas notações e métodos formais para análise de tarefas são difíceis para muitos desenvolvedores aprenderem e aplicarem em seus projetos práticos. Portanto, considerando a necessidade de permitir aos desenvolvedores e aos avaliadores inexperientes uma forma mais fácil e intuitiva para a análise de tarefas, é proposto neste trabalho um modelo de interface criado a partir da observação das estruturas de páginas Web. Este modelo permite ao avaliador definir tarefas em termos de elementos da interface da aplicação Web, eliminando o uso de notações e métodos formais. Além disso, o modelo proposto permite tratar genericamente os elementos da interface, a fim de viabilizar a definição de tarefas em aplicações Web grandes e dinâmicas.

3.2 COP: o modelo de interface

O cenário atual da Web permite que qualquer pessoa que tenha conhecimentos básicos de informática seja capaz de criar e publicar aplicações Web. Sendo assim, dificilmente essas pessoas se preocupam com a avaliação da usabilidade das aplicações que criam, e isso provavelmente gera problemas de usabilidade. Neste contexto, acreditar que essas pessoas irão se dedicar ao estudo de IHC parece utópico, e isso torna pouco provável o uso dos métodos tradicionais de avaliação de usabilidade, recrutando usuários, preparando ambientes, realizando os testes e analisando os resultados.

Considerando que os testes empíricos, baseados na observação do comportamento do usuário durante a realização de tarefas, são eficazes para a identificação de problemas de usabilidade (NIELSEN, 1995), permitir que usuários inexperientes realizem esses testes remotamente parece um rumo promissor para a avaliação de usabilidade.

No entanto, os trabalhos reportados na literatura relacionados à avaliação de usabilidade de interfaces não apresentam uma maneira rápida e intuitiva para a definição e análise de tarefas, e isso motivou o desenvolvimento do modelo de interface COP, que se baseia nos elementos de interfaces Web. O COP considera que qualquer elemento da interface pode fazer parte do caminho necessário para o cumprimento de uma tarefa e, além disso, o modelo permite generalizar os elementos, a fim de facilitar a definição de tarefas pelo avaliador.

Definir tarefas para que sejam avaliadas de forma semi-automática ou automática é desafiador, devido ao fato de que uma tarefa pode ser realizada por diversos caminhos diferentes pelo usuário. Com isso, a definição de tarefas pode se tornar um trabalho exaustivo em grandes aplicações Web e isso pode se agravar naquelas que são dinâmicas, em que muitos elementos são compartilhados entre diversas páginas.

Por exemplo, definir a tarefa *Comprar um produto* em um Web site de *e-commerce* que possui 10.000 produtos torna-se inviável se for necessário especificar individualmente todos os possíveis caminhos. Através do COP, é possível definir caminhos alternativos de forma automática, o que ocasiona a redução do tempo e do esforço para a definição de tarefas em uma aplicação. Portanto, o modelo COP supre, por exemplo, a necessidade de definir diversos caminhos para a tarefa *Comprar um produto* exemplificada anteriormente.

Devido à estrutura não-linear de interfaces Web baseada no hipertexto, uma aplicação Web é composta por várias páginas que, por sua vez, são formadas por elementos como *links*, tabelas, formulários, imagens, etc. É comum que esses elementos sejam compartilhados entre duas ou mais páginas, o que ocorre principalmente em aplicações Web dinâmicas, em que é definida uma estrutura para a interface e os conteúdos são carregados dinamicamente.

A existência de elementos comuns em diversas páginas é intensificada pela utilização de ferramentas de gerenciamento de conteúdo (do inglês, *Content Management Systems - CMS*) e modelos de estrutura (do inglês, *templates*) durante o desenvolvimento de uma aplicação Web, pois essas ferramentas dispõem os elementos nas páginas de forma pré-definida para facilitar o desenvolvimento através da publicação de componentes plugáveis.

Conforme a especificação da linguagem HTML 4.01 do *World Wide Web Consortium* (W3C) (RAGGETT; HORS; JACOBS, 1999), os elementos de uma página podem ser agrupados pelos elementos DIV e SPAN, que podem ser usados em conjunto com folhas de estilo. Além disso, a especificação ainda define que os atributos *id* e *class* são identificadores de elementos, de forma que o *id* deve ser um nome único em uma página.

Considerando a especificação do W3C e analisando as estruturas das aplicações Web, foi desenvolvido um modelo de interface, o COP, composto por três conceitos: ***C**ontainer*, **O**bjeto e **P**ágina, detalhados a seguir.

- **Objeto:** qualquer elemento de página que sofre uma ação direta do usuário, tais como: *links*, caixas de texto, imagens e botões de ação, caixa de seleção e outros elementos de formulário.
- ***Container:*** qualquer elemento de página que contém um ou mais objetos, tais como células de tabelas, divisões do *layout* da página, listas e formulários.
- **Página**: uma interface que possui um ou mais *containers*, ou seja, é um documento que compõe uma aplicação Web.

Em uma página, um Objeto pode ser único (através de um identificador - atributo *id* da especificação HTML) ou semelhante a outros em relação à formatação (ex.: borda, fonte, cor) ou em relação ao conteúdo (ex.: textos e imagens). Um *Container* pode ser identificado de forma única ou ser semelhante a outros *containers* quanto à formatação (ex.: borda, fundo). Além disso, os objetos e *containers* podem existir em uma ou mais páginas de uma aplicação Web.

A Figura 1 apresenta a estrutura do modelo de interface proposto. As linhas que ligam os elementos gráficos representam uma possível relação e não a obrigatoriedade de associação entre eles. Um *objeto único* é um objeto identificado de forma única, por meio de um identificador. Assim como um objeto único, os objetos semelhantes quanto à formatação ou ao conteúdo podem estar contidos em um *container* também identificado de forma única por atributos HTML (*Container único*) ou em *containers* semelhantes em relação à formatação (*Container similar*).

Na figura, ainda é possível perceber que um *container* pode pertencer a uma única página ou a várias páginas da aplicação Web. Dessa forma, o modelo abrange desde um *link* existente em um *container* específico na página inicial de uma aplicação Web a todos

os links que permitem acessar os detalhes dos produtos de um *website* de *e-commerce*, por exemplo.

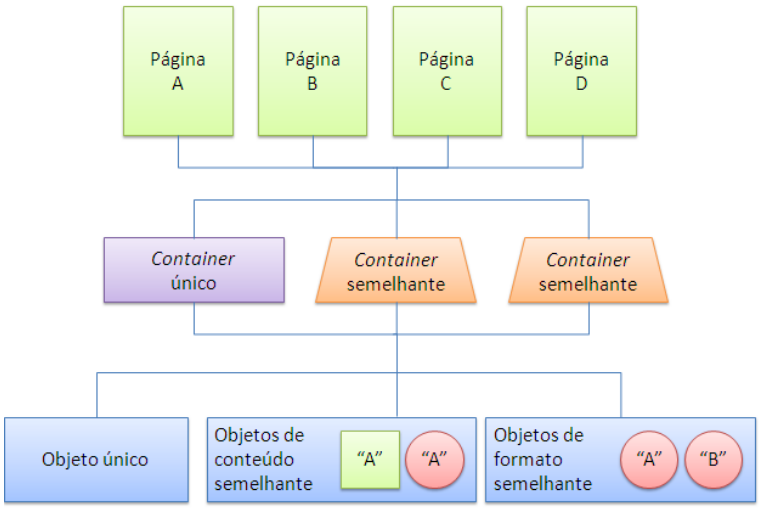

Figura 1: Modelo de interface para definição e análise de tarefas

Comparando os padrões do W3C para construção de documentos baseados em hipertexto (RAGGETT; HORS; JACOBS, 1999) com os conceitos do COP, é possível afirmar que a aplicação do modelo permite identificar qualquer elemento de uma interface Web durante a definição de tarefas, de forma que é possível detectar, por meio de uma linguagem de *script* executada no cliente, os eventos realizados sobre os elementos da interface e, consequentemente, incluí-los no caminho ótimo de uma tarefa.

Assim, o COP permite a análise automática das tarefas definidas pelo avaliador, já que é possível comparar as interações dos usuários com as ações pré-definidas pelo avaliador, conforme discutido na Seção 4.4.

A Tabela 1 resume as definições do modelo de interface proposto. Essas definições estão separadas de acordo com os três conceitos que compõem o modelo. A coluna da direita apresenta as opções aplicáveis a cada conceito, que permitem tratá-los genericamente a fim de facilitar a definição dos diversos caminhos possíveis para a realização de uma tarefa.

Com o desenvolvimento do COP e, consequentemente, as opções definidas na Tabela 1, é possível definir tarefas reais de forma intuitiva em qualquer aplicação Web, pois as tarefas são definidas por ações físicas nos termos dos elementos da interface. Assim, o tempo de aprendizagem para a definição de tarefas é mínimo, correspondente apenas à compreensão das opções do COP.

Analisando diversas interfaces de aplicações Web modernas, tais como comércio vir-

Tabela 1: Conceitos do modelo de interface

Conceito	**Opções**
Objeto	Um objeto individual
	Objetos com o mesmo conteúdo
	Objetos com a mesma formatação
	Quaisquer objetos
Container	Um *container* individual
	Containers com a mesma formatação
	Quaisquer *containers*
Página	Uma página individual
	Quaisquer páginas

tual, portais de notícias, aplicações de *webmail* e ambientes de aprendizagem, com o objetivo de validar a abrangência do modelo de interface para tarefas reais, foi selecionado um ambiente virtual de aprendizagem (AVA), denominado 4Learn [1] , para exemplificar a aplicação das opções do COP.

O AVA 4Learn é uma ferramenta que pode ser usada tanto para a educação a distância como para o apoio ao ensino presencial. Nesse ambiente, professores e alunos interagem através de ferramentas de colaboração e, principalmente, para a publicação e correção de atividades. Portanto, foi selecionada a tarefa "Corrigir uma atividade" devido a sua importância para os usuários do ambiente. Essa tarefa é realizada por um professor e é uma das principais atividades do AVA. A seguir, são listados os passos do caminho ótimo da tarefa, a fim de exemplificar o uso do modelo de interface COP na definição de tarefas.

3.2.1 Exemplo de aplicação do COP na definição de tarefas

Após o acesso do professor ao sistema, as funcionalidades disponíveis ao seu perfil são exibidas em um menu posicionado à esquerda da tela. Para corrigir as atividades, o professor deve clicar no *link* "Corrigir atividades" do menu, conforme destaca a Figura 2. Para incluir essa ação no caminho ótimo da tarefa, o avaliador deve selecionar o *link* "Corrigir atividades" e definir as opções do COP apresentadas na Figura 2. Isto é, apenas o objeto "Corrigir atividades" no *container* "menu" da página "index.asp" será incluído no caminho ótimo da tarefa.

Após a execução do passo anterior, são exibidas as disciplinas e turmas nas quais o professor leciona, conforme apresentado na Figura 3. Porém, como a tarefa "Corrigir uma atividade" pode ser realizada para alunos de qualquer turma ou disciplina, seria inviável

[1] http://www.4learn.pro.br/

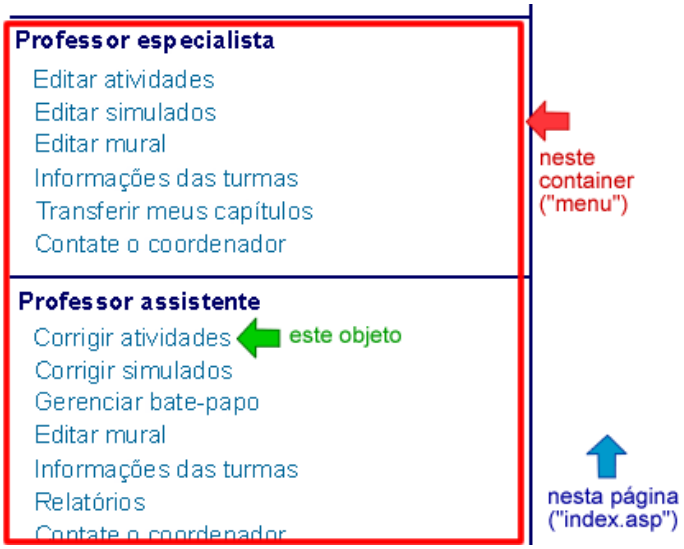

Figura 2: 4Learn: opções do menu de professores

definir cada possível caminho para completar essa tarefa, tendo em vista que podem existir muitas turmas, muitos alunos e muitas atividades para correção. Portanto, utilizando as opções de generalização do COP, o avaliador pode definir o caminho uma única vez e, automaticamente, os caminhos alternativos são detectados. Neste passo da definição da tarefa, conforme destaca a Figura 3, o avaliador pode selecionar o *link* "Corrigir" da disciplina "CTIG - Informática Aplicada", da turma "1A - Mecânica", e definir as opções do COP como: objetos com mesmo conteúdo que este neste container ("conteudo"). Assim, serão incluídos no caminho ótimo da tarefa todos os *links* que possuem o texto "Corrigir" e estão dentro do *container* único denominado "conteudo" na página "turmas.asp".

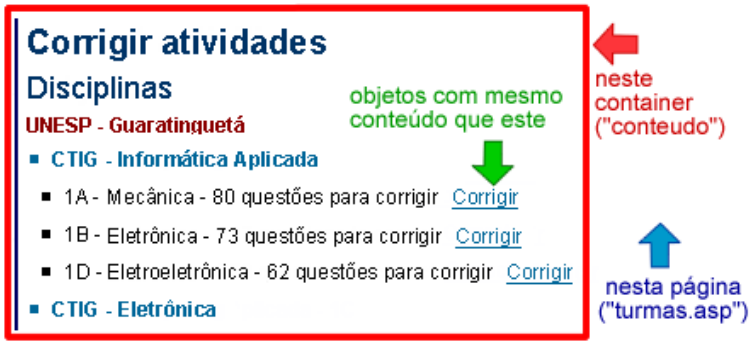

Figura 3: 4Learn: lista de disciplinas e turmas para correção de atividades

Após o professor selecionar uma turma de uma disciplina, é carregada uma página com todas as atividades da turma selecionada, conforme mostra a Figura 4. É possível observar que cada atividade possui 2 *links*: "Corrigir" e "Finalizar". A figura exibe duas questões de alunos diferentes (questões 11 e 12), o que sugere que podem existir muitas atividades para correção em uma turma. Portanto, assim como no passo anterior, é

inviável definir manualmente cada possível caminho para completar a tarefa. Para incluir automaticamente qualquer uma das atividades no caminho ótimo da tarefa, o avaliador pode selecionar o *link* "Corrigir" da atividade de qualquer aluno e definir as opções do COP conforme mostra a figura. Isto é, como o *container* "divAtividade11" da atividade 11 possui a mesma formatação dos *containers* das atividades 12, 13, etc., é possível incluir no caminho ótimo da tarefa quaisquer *links* com o texto "Corrigir" de todas as atividades exibidas na página. Além disso, definindo a opção "em várias páginas", serão consideradas todas as páginas de atividades de todas as turmas.

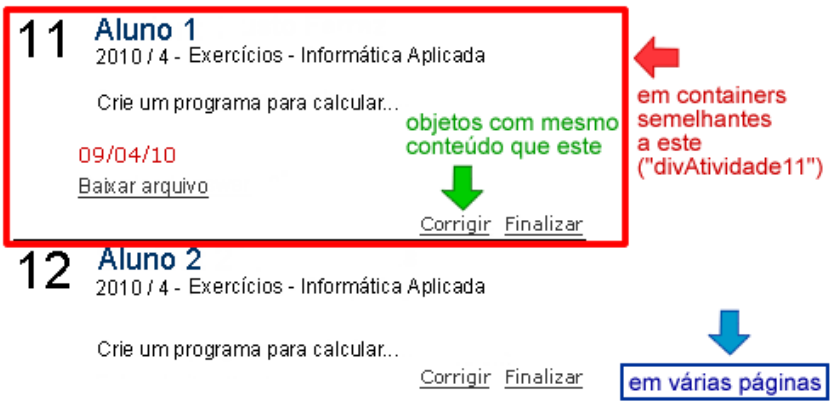

Figura 4: 4Learn: lista de atividades da disciplina selecionada pelo professor

Após a execução do passo anterior, o professor deve informar um texto e confirmar a correção da atividade, conforme mostra a Figura 5. Para incluir a digitação do texto no caminho ótimo da tarefa, o avaliador deve selecionar a área de texto da correção e definir as opções conforme a Figura 5, considerando que a área de texto "resposta" existe somente dentro do *container* denominado "formatividade". Em seguida, o avaliador deve selecionar o botão "Corrigir" e definir as opções apresentadas na figura, considerando que o botão, assim como a área de texto, existe somente dentro daquele *container*. Além disso, a opção "em várias páginas" permite que sejam consideradas todas as páginas que contenham o formulário "formatividade".

A tarefa "Corrigir uma atividade" destaca a aplicação das diferentes opções do modelo de interface COP, que permitem que o avaliador defina uma única vez o caminho ótimo da tarefa, considerando automaticamente todas as atividades de todos os alunos de todas as turmas e disciplinas disponíveis no ambiente.

Assim, o COP se mostra eficiente para suportar a definição de tarefas reais das aplicações Web.

Além do uso do COP na definição de tarefas, os três conceitos que compõem o modelo

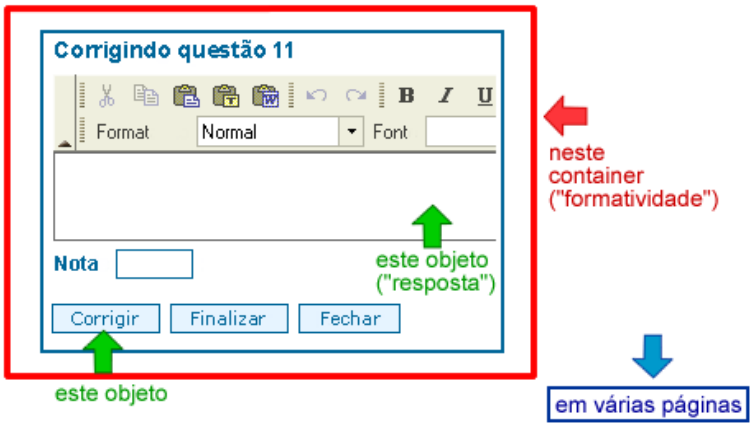

Figura 5: 4Learn: Formulário para correção de atividades

viabilizam a comparação entre o caminho ótimo criado pelo avaliador e as interações dos usuários, a fim de executar a análise de tarefas. Na análise de tarefas, a sequência de eventos do caminho ótimo de uma tarefa é comparada às sequências de eventos das interações dos usuários, objetivando identificar as diferenças entre o que é esperado pelo avaliador e o que realmente é executado pelos usuários finais.

Considerando a similaridade de objetos e *containers* no COP, a comparação entre os eventos do caminho ótimo e os eventos dos usuários possibilita saber qual o grau de similaridade entre dois eventos, ao invés de saber apenas se os eventos são iguais ou não. Isto é, dois eventos que ocorreram no mesmo *container*, na mesma página, mas em objetos diferentes são mais semelhantes que dois eventos que ocorreram na mesma página, mas em *containers* e objetos diferentes. Essa capacidade de calcular a similaridade entre dois eventos é oferecida pelo modelo de interface COP.

Reconhecer a similaridade entre eventos é a principal característica para a análise de tarefas, pois, a partir disso, podem ser identificados, em cada passo da tarefa, os erros em relação ao caminho ótimo, e ainda quais elementos da interface estão diretamente relacionados a esses erros. Consequentemente, é possível propor recomendações para readaptação da interface, baseadas na análise de tarefas.

Para isso, o modelo de interface COP foi estendido e implementado pelo sistema USABILICS, que é apresentado no Capítulo 4.

3.3 Considerações finais

Este Capítulo apresentou os conceitos relacionados à análise de tarefas e algumas abordagens utilizadas sobre o assunto para a avaliação de usabilidade.

Além disso, foi apresentado o modelo de interface COP, que é a principal contribuição deste trabalho. Esse modelo permite tratar elementos da interface de maneira genérica, facilitando a definição e a análise de tarefas em aplicações Web modernas. O exemplo de tarefa apresentado neste capítulo permite compreender a aplicação das opções de generalização do COP.

Definir tarefas nos termos de elementos da interface elimina a necessidade de se aprender notações, possibilitando implementar soluções para a análise de tarefas com o objetivo de identificar problemas de usabilidade pontualmente na interface, já que os elementos da interface compõem as tarefas.

Considerando a definição de tarefas a partir do modelo de interface COP, é possível destacar as seguintes vantagens:

1. A definição de tarefas ocorre por meio de uma simples navegação pela interface e em termos visuais dos elementos da aplicação Web;
2. Redução do tempo e do esforço para a definição de vários caminhos de uma tarefa;
3. Flexibilidade para definição de tarefas em qualquer interface Web, tanto em relação à estrutura de páginas quanto às áreas restritas de aplicações Web que necessitam de autenticação de usuários;
4. Possibilidade de definição de tarefas em interfaces ricas que utilizam tecnologias como AJAX, por exemplo;
5. Definição desde tarefas simples com caminhos lineares até tarefas complexas com ações opcionais e passos que podem se repetir.

No intuito de avaliar remota e automaticamente a usabilidade de aplicações Web, o modelo de interface COP foi estendido e implementado pelo sistema USABILICS. O Capítulo 4 detalha a estrutura e o funcionamento do USABILICS, explicando a coleta automática de dados, a definição e a análise de tarefas, sendo que todos os módulos do sistema se baseiam no modelo de interface COP.

4 USABILICS

Este capítulo aborda a estrutura e o funcionamento do USABILICS, detalhando os módulos que o compõem, as informações recebidas e geradas pelo sistema, e apresenta ainda os trabalhos relacionados mais relevantes sobre avaliação remota de usabilidade. A Seção 4.1 apresenta de uma forma breve a estrutura e o funcionamento do USABILICS. As Seções 4.2, 4.3 e 4.4 detalham o funcionamento dos módulos do sistema, e a Seção 4.5 apresenta os trabalhos relacionados mais relevantes da área de avaliação remota de usabilidade.

4.1 Estrutura e funcionamento

O USABILICS é um sistema Web para avaliação remota e automática de usabilidade de aplicações Web que estende e implementa o modelo de interface COP. Este sistema é composto por três módulos: um módulo para coleta automática de dados das interações dos usuários; um módulo para definição de tarefas pelo avaliador; e um módulo para análise de tarefas. O COP tem fundamental importância em todo o funcionamento do USABILICS, pois a coleta de dados, a definição e a análise de tarefas se baseiam nas opções de generalização do modelo de interface, considerando os conceitos *Container*, Objeto e Página. Assim, a análise de tarefas ocorre pela comparação entre as sequências de ações dos usuários e a sequência definida pelo avaliador, baseando-se nas opções de generalização do COP. A Figura 6 resume o funcionamento e os módulos do USABILICS. Na figura, é possível observar que o módulo de coleta de dados é uma aplicação que executa na máquina cliente, suportada pelo navegador do usuário, e que o módulo de análise de tarefas é responsável por gerar o índice de usabilidade e as recomendações relacionadas aos problemas de usabilidade identificados. Para a execução da análise de tarefas, os dados das interações dos usuários são coletados no navegador do usuário e, em seguida, são comprimidos e enviados ao servidor. Paralelamente, o avaliador pode definir as tarefas da aplicação Web através do módulo denominado UsaTasker. No servidor,

os dados são processados e é executado um algoritmo para comparar as sequências de eventos dos usuários com a sequência de eventos do caminho ótimo de cada tarefa. Como resultado, são gerados o índice de usabilidade e as recomendações pertinentes ao problemas de usabilidade de cada tarefa.

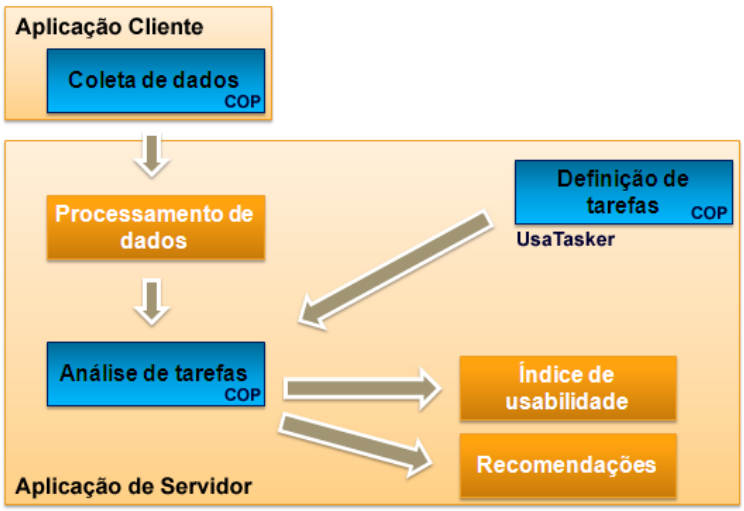

Figura 6: Estrutura dos módulos do USABILICS

Considerando a taxonomia proposta por Balbo (1995), descrita na Seção 2.3.4, o USABILICS abrange os três níveis de automatização: captura, análise e crítica. Em relação à taxonomia proposta por Ivory e Hearst (2001), o USABILICS classifica-se da seguinte maneira:

- **Classe do método:** Teste, pois observa a interação dos usuários, mesmo que remota e automaticamente.
- **Tipo de método:** Análise de log e teste remoto.
- **Tipo de automatização:** captura, análise e crítica.
- **Nível de esforço:** definição das tarefas da aplicação Web pelo avaliador; todas as outras etapas são automatizadas.

Por automatizar as três fases da avaliação (captura, análise e crítica) pode-se afirmar que o USABILICS é um sistema de avaliação automática. Devido à separação entre participantes e avaliadores no espaço/tempo devido à realização de teste remoto, pode-se afirmar que o sistema é também de avaliação remota. Além disso, o USABILICS utiliza os dois métodos citados por Fidas et al. (2007) para avaliação remota de usabilidade em relação à técnica automatizada: os testes empíricos e a avaliação baseada em modelo. O

sistema usa testes empíricos do ponto de vista em que são analisadas as interações dos usuários, e utiliza da avaliação baseada em modelo por centrar-se no modelo de interface COP.

As seções seguintes detalham os módulos de coleta de dados, definição e análise de tarefas do USABILICS.

4.2 Coleta automática de dados

No USABILICS, a coleta automática de dados das interações dos usuários é realizada através da comunicação entre uma aplicação cliente e uma aplicação de servidor. A aplicação cliente foi implementada na linguagem JavaScript, a fim de capturar todos os dados sobre os eventos realizados pelo usuário final no navegador. Assim, a coleta de dados ocorre durante a interação do usuário e, em tempo real, os dados são comprimidos e enviados à aplicação de servidor, implementada na linguagem *Java Server Pages* (JSP). Em seguida, os dados são descomprimidos e armazenados em um banco de dados implementado no sistema gerenciador de banco de dados Postgres. A Figura 7 apresenta o funcionamento do módulo de coleta de dados do USABILICS.

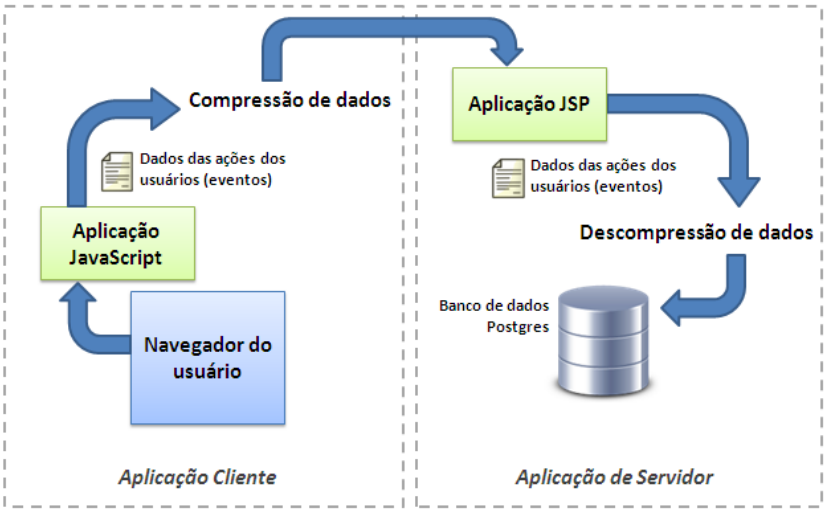

Figura 7: Coleta de dados no USABILICS

Para a coleta dos dados da interação dos usuários, o USABILICS é semelhante a WELFIT (SANTANA; BARANAUSKAS, 2010) e WAUTT (RIVOLLI; MARINHO; PANSANATO, 2008), os quais utilizam uma aplicação no navegador do cliente, desenvolvida na linguagem JavaScript, que é inserida nas páginas das aplicações Web para capturar movimentos de mouse, movimentos de barra de rolagem, redimensionamento de janelas, outros eventos

resultantes da interação do usuário com a interface e eventos gerados pelas páginas das aplicações Web (por exemplo, *load* e *unload*).

No entanto, o módulo de coleta de dados do USABILICS também captura atributos CSS (*Cascading StyleSheet*) dos elementos da interface com os quais os usuários interagem, tendo em vista que o posicionamento, as cores, as fontes e outras características referentes à forma dos elementos interferem diretamente na interação dos usuários. Além disso, os atributos CSS são utilizados na análise de tarefas em conjunto com o modelo de interface COP, na comparação de objetos e *containers* que possuem a mesma formatação, permitindo tratá-los genericamente na aplicação Web.

No USABILICS, são coletadas também diversas características dos eventos e objetos manipulados durante a interação do usuário, tais como: *timestamp*, posicionamento das barras de rolagem, URL, dimensões da janela, posicionamento do cursor do *mouse*, atributos de identificação dos elementos, conteúdo, posicionamento absoluto (em relação à página) e atributos dos *containers* do objeto manipulado.

Com a expansão do uso da linguagem JavaScript e de tecnologias como AJAX, surgiu a possibilidade de desenvolvimento de interfaces ricas para Web, que comumente permitem o carregamento de conteúdos de forma assíncrona. Para abranger tanto interfaces ricas como interfaces tradicionais, o USABILICS suporta uma granularidade fina na coleta de dados dos eventos, a fim de facilitar a descoberta de problemas de usabilidade. Por exemplo, a coleta do *timestamp* dos eventos permite verificar se houve interrupções na execução de tarefas durante a interação do usuário.

O USABILICS coleta, por exemplo, as dimensões da janela do navegador na ocorrência de todos os eventos, pois o usuário pode redimensionar a janela ocasionando uma modificação da disposição dos elementos de posicionamento relativo, o que pode ocultar uma determinada informação e atrasar a realização de alguma tarefa. Além das dimensões da janela do navegador, são coletadas as dimensões da tela do monitor, informações sobre o uso das teclas Alt, Ctrl e Shift e a URL da página visitada anteriormente no histórico de navegação. De forma complementar aos eventos *load* e *unload*, foi criado um evento chamado *openpage*, que corresponde à abertura de uma página, tendo em vista que o evento *load* só ocorre após o carregamento completo do conteúdo da página. O evento *openpage* permite verificar, por exemplo, se os usuários interagem com a interface antes do carregamento completo de seu conteúdo, de forma que, neste caso, podem ser gerados eventos entre a abertura da página e o seu carregamento completo (do inglês, *load*).

Com o objetivo de identificar um mau funcionamento da interface das aplicações Web

e problemas específicos de usabilidade em alguns navegadores, são coletadas informações detalhadas sobre o navegador utilizado pelo usuário, incluindo os *plug-ins* instalados.

As informações sobre os *plug-ins* são importantes para as verificações relacionadas à exibição do conteúdo, tendo em vista que a não utilização de um *plug-in* pode ocultar um vídeo ou uma animação, prejudicando a interação dos usuários ou até impossibilitando-os de navegar, no caso de aplicações Web que utilizam animações em menus, por exemplo.

Na linguagem HTML 4.01 e em sua sucessora XHTML 1.0, os elementos são identificados por meio do atributo *id* (W3C, 2002), porém alguns desenvolvedores, erroneamente, não utilizam esse atributo para identificar todos os elementos das páginas, gerando a necessidade de criação de uma alternativa para identificação dos elementos das páginas. O W3C definiu uma interface para programação de aplicação chamada *Document Object Model* (DOM), para documentos XML bem formados e documentos HTML válidos (HÉGARET; L.; ROBIE, 2000). No DOM, os documentos possuem uma estrutura lógica em árvore, comumente chamada de árvore DOM (do inglês, *DOM-Tree*).

Na coleta de dados do USABILICS, foi definida uma identificação alternativa nomeada *GeneratedID*, que é uma sequência numérica que representa a profundidade e a largura de um elemento de página na árvore DOM. A profundidade é gerada a partir da relação entre os nós pais e nós filhos, e a largura, pela relação entre os nós irmãos.

Portanto, cada objeto manipulado pelo usuário é identificado de forma única em uma página pelo *GeneratedID*, flexibilizando a aplicação do modelo de interface, a fim de que o desenvolvedor não necessite alterar o código da aplicação Web para satisfazer esse requisito de identificação, exceto no caso específico em que o atributo *id* não é usado e deseja-se avaliar interfaces ricas cuja árvore DOM é alterada dinamicamente (por meio da tecnologia AJAX, por exemplo).

Todos os dados coletados pela aplicação cliente são enviados a um servidor e, posteriormente, são pré-processados e armazenados em um banco de dados. Respeitando os requisitos definidos por Atterer e Schmidt (2007) para ferramentas de coleta automática de dados dos usuários, foi aplicada uma técnica de compressão dos dados e foram realizados diversos testes para otimização do espaço de memória e do uso de CPU da aplicação cliente, de modo que a ferramenta não interfira de maneira perceptível na navegação dos usuários.

Devido às limitações da linguagem JavaScript para a manipulação de bits, foi utilizada uma técnica simples de substituição de símbolos, através de uma tabela de conversão, a

fim de otimizar a quantidade de caracteres enviados ao servidor. Portanto, todo atributo HTML e CSS coletado em cada evento da interação do usuário é convertido em um símbolo. Isso gera a necessidade de descompressão na aplicação que executa no lado servidor.

Para otimizar a compressão dos dados e reduzir o tráfefo entre a aplicação cliente e a aplicação servidora, é realizada uma comparação entre os atributos do evento atual com o evento coletado anteriormente, a fim de eliminar os atributos que possuam o mesmo valor. Assim, quanto mais semelhantes os eventos forem, menor será a cadeia de caracteres enviada ao servidor.

Observando o funcionamento do módulo de coleta de dados, é possível perceber que há uma grande quantidade de detalhes sobre os eventos das interações dos usuários, o que possibilita a identificação dos elementos da interface que estão relacionados a problemas de usabilidade identificados.

A seção seguinte detalha o UsaTasker, o módulo do USABILICS para definição de tarefas.

4.3 UsaTasker: definição de tarefas

A definição de tarefas é fundamental para a avaliação remota e automática de usabilidade, pois permite a utilização de tarefas reais das aplicações simulando as avaliações que são realizadas por testes tradicionais em laboratório. Assim, é possível avaliar as interações dos usuários durante a execução de uma tarefa.

Avaliar remota e automaticamente a usabilidade de aplicações Web modernas é desafiador por causa da dificuldade existente na definição de objetivos e tarefas. Devido à complexidade de algumas funcionalidades oferecidas pelas aplicações Web, definir tarefas que atendam a um determinado objetivo torna-se difícil por causa da grande quantidade de caminhos possíveis nas interfaces dessas aplicações.

As abordagens para definição de tarefas reportadas na literatura geralmente utilizam notações (PAGANELLI; PATERNò, 2002; TIEDTKE; MARTIN; GERTH, 2002) , que se baseiam em modelos genéricos de tarefas. No entanto, o uso de notações não se mostra eficaz para a avaliação de aplicações Web, conforme discutido na Seção 3.1.

Ainda em relação a essas abordagens, geralmente a captura das interações exige ações específicas dos usuários para informar o cumprimento de cada passo da tarefa. Isso exige

um esforço de treinamento do usuário para a participação nos testes de usabilidade. Porém, treinar usuários para a participação em testes de usabilidade remotos contraria uma das vantagens da avaliação remota de usabilidade, que é a possibilidade de ter um grande número de participantes nos testes, e ainda dispersos geograficamente.

A fim de suprir essas deficiências na avaliação da usabilidade de aplicações Web, foi criado o UsaTasker, uma ferramenta que permite, baseando-se no COP, a definição de quaisquer tarefas em interfaces Web, desde tarefas simples com caminhos lineares até tarefas mais complexas que possuem ações opcionais, por exemplo.

Como apresentado na Figura 6, o UsaTasker é uma aplicação de servidor, semelhante a Google Analytics (GOOGLE, 2011) e WELFIT (SANTANA; BARANAUSKAS, 2010), que permite aos avaliadores cadastrarem-se e informarem quais aplicações Web desejam avaliar. O UsaTasker permite ao avaliador o gerenciamento das tarefas que deseja avaliar em uma aplicação Web, sendo que é possível criar novas tarefas, alterá-las e removê-las. As tarefas criadas compõem o conjunto de tarefas da aplicação Web a ser avaliada, sendo que cada tarefa representa ou compõe objetivos dos usuários reais na aplicação. As sequências de eventos que formam uma tarefa podem ser gerenciadas pelo UsaTasker, o qual permite generalizar ou especializar os eventos que as compõem.

Para cada aplicação Web cadastrada, é gerado um código JavaScript que deve ser inserido pelo avaliador nas páginas que serão avaliadas.

Para definir o caminho ótimo de uma tarefa, o avaliador informa os dados de acesso na aplicação servidora, seleciona a aplicação Web e informa o nome e a descrição (opcional) para a tarefa que deseja definir.

Na criação de uma nova tarefa, o avaliador navega na interface da aplicação Web para definir o caminho da tarefa e, automaticamente, o UsaTasker define caminhos alternativos conforme as opções do modelo de interface selecionadas pelo avaliador na execução de cada ação. Durante a criação de uma nova tarefa, o avaliador possui controle sobre o início e a finalização da tarefa.

A Figura 8 mostra as opções do COP disponíveis ao avaliador durante a definição de cada ação da tarefa. Ao selecionar um objeto na interface para definir um evento, o avaliador visualiza as opções de generalização referentes ao objeto, bem como os seus *containers* dentro da página. Os *containers* podem ser tabelas, células, formulários e divs, por exemplo.

Assim, o avaliador tem a possibilidade de selecionar qualquer *container* do objeto, do

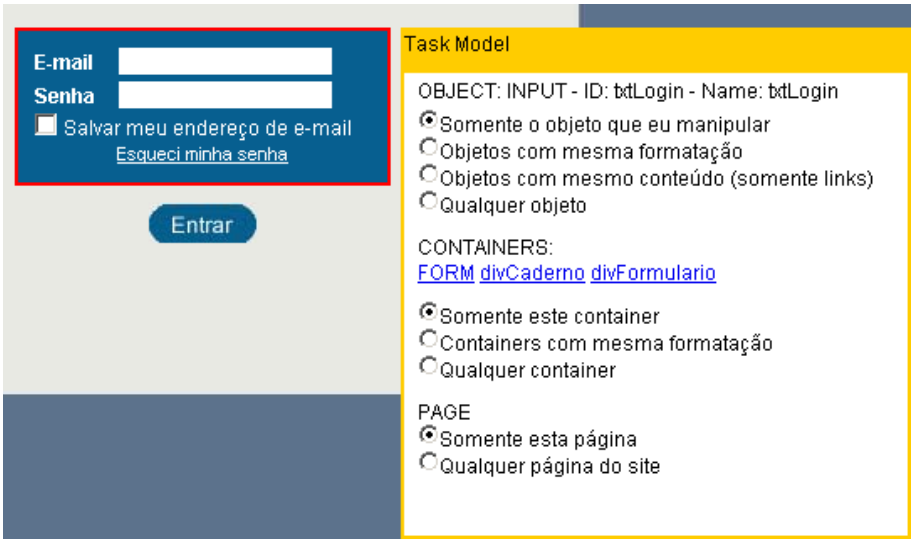

Figura 8: Opções de generalização das ações da tarefa

mais próximo ao mais distante, definindo os objetos que serão considerados no caminho ótimo da tarefa. Por exemplo, se existir um *container* A com alguns *links* e, dentro desse *container*, existir um *container* A1 também com alguns *links*, a seleção do *container* A durante a definição da tarefa, com a opção de considerar objetos semelhantes a um *link*, irá considerar todos os *links* de A e A1, porque o *container* A1 está dentro do *container* A.

Dessa forma, o avaliador tem a flexibilidade de generalizar os objetos e os *containers* conforme a necessidade da tarefa que deseja definir. Assim, é possível considerar em cada passo da tarefa desde os objetos (*links*, por exemplo) que pertençam a um formulário específico até todos os objetos de uma página.

Após o avaliador finalizar a gravação do caminho ótimo de uma tarefa, as ações capturadas são apresentadas graficamente conforme mostra a Figura 9. Apresentar graficamente ao avaliador as ações que compõem o caminho ótimo da tarefa facilita a visualização e a compreensão.

Na Figura 9, cada retângulo corresponde a uma ação da tarefa e as setas indicam a sequência dos eventos que compõem o caminho ótimo da tarefa. Além de visualizar as ações, o avaliador pode excluí-las utilizando o ícone "X" localizado na parte superior de cada retângulo, a fim de eliminar um evento desnecessário para o caminho ótimo da tarefa.

Complementando as opções do modelo de interface COP, o UsaTasker define três novos recursos que permitem a definição de tarefas complexas nas interfaces de aplicações Web modernas. Esses recursos definem que (i) uma ação pode ser opcional na realização

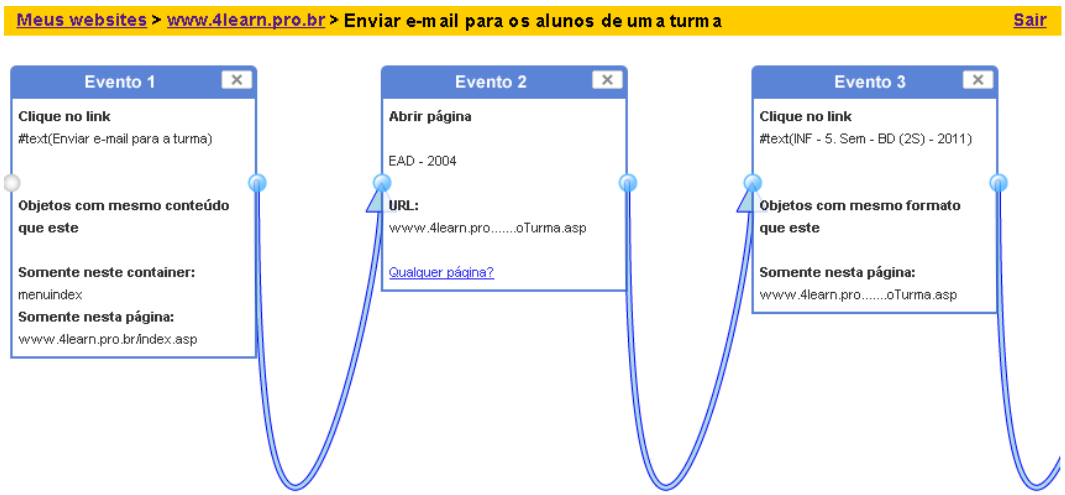

Figura 9: Exibição das ações capturadas para uma tarefa no UsaTasker

de uma tarefa; (ii) uma ação pode ser executada fora de ordem em um conjunto de ações; e (iii) uma ou mais ações podem ser repetidas na realização de uma tarefa. No UsaTasker, podem ser definidos conjuntos de ações que podem ou não se repetir. Além disso, ações consecutivas podem possuir ou não uma relação de precedência.

4.3.1 Ordem entre ações: relação de precedência

Algumas tarefas realizadas nas interfaces de aplicações Web possuem ações que não respeitam uma relação de precedência, portanto são ações que independem da ordem em que são executadas.

Por exemplo, o preenchimento de um formulário geralmente não possui uma ordem definida das caixas de texto e outros elementos. Portanto, se um usuário preenche um formulário da primeira caixa de texto até a última, e outro preenche na ordem inversa, ambos cumprem a tarefa de preencher o formulário.

Por padrão, o UsaTasker define que as ações da tarefa possuem relação de precedência, porém o avaliador pode definir que um ou vários conjuntos de ações não possuem essa relação. Para isso, o avaliador marca as ações consecutivas que não possuem relação de precedência. Cada bloco de ações consecutivas sem relação de precedência é interpretado separadamente dos outros, permitindo que dois conjuntos de ações tenham uma relação de precedência mesmo que não haja relação de precedência entre dos blocos.

A Figura 10 mostra as ações de uma tarefa, sendo que as ações destacadas em amarelo não possuem relação de precedência, ao contrários das ações cujos retângulos possuem fundo branco.

4.3.2 Ações opcionais e obrigatórias

Em diversas aplicações Web, existem ações que podem ou não compor a sequência de ações para o cumprimento de uma tarefa. Usando como exemplo o preenchimento de um formulário, podem existir campos opcionais e campos obrigatórios. Ações opcionais não ocorrem somente no preenchimento de formulários, pois existem tarefas em que os usuários podem ou não clicar em um *link* ou abrir uma página, por exemplo.

Na Figura 10, a ação nomeada "Evento 2" é uma ação opcional, identificada pela linha tracejada. Além disso, observa-se que uma ação pode ser opcional e, simultaneamente, pertencer a um bloco de ações sem relação de precedência. Isso aumenta a flexibilidade para definição de tarefas complexas.

4.3.3 Repetição de ações

Muitas tarefas de aplicações Web modernas possuem ações que podem ou devem ser repetidas um determinado número de vezes dentro da sequência de ações da tarefa. Por exemplo, em uma aplicação de comércio virtual, a ação de selecionar um produto e adicioná-lo ao carrinho de compras pode ser repetida várias vezes, de acordo com a quantidade de produtos que o cliente deseja comprar.

No sentido de atender essa necessidade, o UsaTasker permite ao avaliador a definição de conjuntos de ações que podem se repetir na tarefa. Para isso, o avaliador cria uma seta para conectar a ação final à ação inicial do conjunto de ações que podem se repetir. Além disso, o avaliador pode remover há qualquer momento as setas que indicam a repetição de ações em uma tarefa.

Na Figura 10, a seta em azul escuro indica que as ações "Evento 2" e "Evento 3" podem se repetir.

Para a validação da definição de tarefas pelo UsaTasker, foi selecionada uma aplicação Web real, a fim de exemplificar as opções do UsaTasker em conjunto com a implementação do modelo de interface COP. A aplicação selecionada foi o *website* Groupon [1].

A Seção 4.3.4 apresenta os detalhes da definição de tarefas da aplicação selecionada.

[1] http://www.groupon.com/

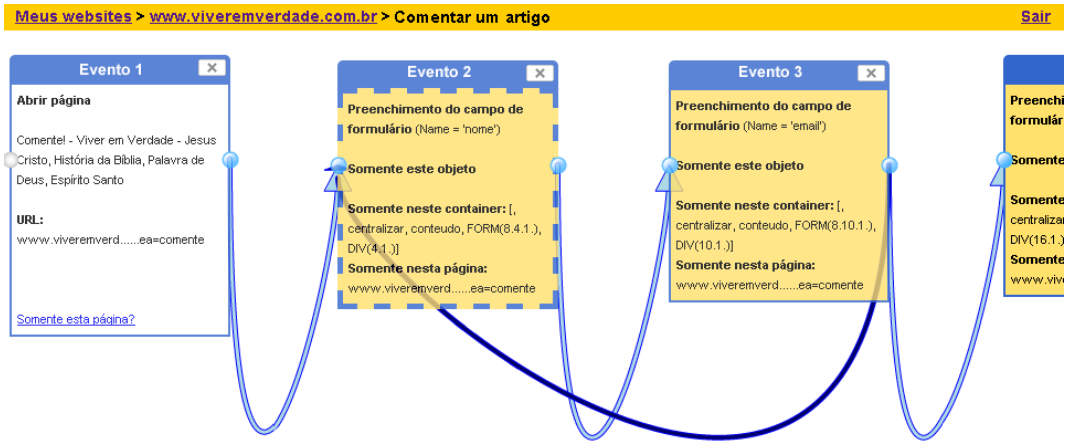

Figura 10: Exibição dos recursos do UsaTasker para a definição de tarefas

4.3.4 Groupon

O Groupon é um serviço de comércio virtual que segue o conceito de compra coletiva. Assim como em qualquer *website* de comércio virtual, a principal tarefa desse *website* é comprar uma oferta. A seguir, a aplicação do UsaTasker para a definição dessa tarefa é exemplificada passo-a-passo.

A Figura 11 destaca o primeiro passo para realizar uma compra no Groupon, que é selecionar uma oferta. Nessa página, quando o usuário posiciona o ponteiro do *mouse* sobre uma oferta, aparece a botão "Ver oferta", como mostra a Figura 11. Para incluir no caminho ótimo da tarefa a ação de clicar no botão "Ver oferta", o avaliador pode selecionar este botão e também o *container* destacado em vermelho na figura, informando que devem ser considerados os botões com o conteúdo "Ver oferta" em *containers* semelhantes ao selecionado, somente nesta página. Portanto, utilizando as opções de generalização do modelo COP, é possível, no UsaTasker, definir essa ação uma única vez para todas as ofertas da página de ofertas.

Após a execução do passo anterior, o usuário visualiza a oferta selecionada e, para continuar a compra, deve clicar no botão "Comprar", destacado na Figura 12. Para incluir essa ação no caminho ótimo da tarefa, o avaliador deve selecionar as opções do COP marcadas na figura, isto é, o objeto botão "Comprar" no *container* destacado em vermelho, em quaisquer páginas. Assim, serão consideradas todas as ofertas do *website*, tendo em vista que todas são carregadas nessa mesma interface.

É importante destacar que podem ser compradas várias ofertas, caso o usuário volte à lista de ofertas e repita as ações anteriores. Isso pode ser definido pelo avaliador no UsaTasker por meio do recurso de ações repetidas, apresentado na Seção 4.3.3.

Figura 11: Página de ofertas do *website* Groupon

Figura 12: Página da oferta selecionada pelo usuário no *website* Groupon

A Figura 13 destaca as opções de desconto que podem ser oferecidas em algumas ofertas do *website*, isto é, as ações nesta página são opcionais e devem ser definidas dessa forma no UsaTasker. Na figura, é possível observar que podem existir várias opções de desconto, assim como a oferta ilustrada na figura possui as opções "Jantar para 2" e "Jantar para 4" em *containers* com mesma formatação. Como esses objetos possuem conteúdos diferentes, o avaliador pode definir no UsaTasker a opção do COP que considera objetos com mesma formatação. Sabendo que cada oferta possui a sua própria página, o avaliador deve definir que a ação de clicar em um desconto pode ocorrer em várias páginas, conforme ilusta a Figura 13.

Caso o usuário do *website* ainda não tenha logado no sistema, a opção destacada na Figura 14 será exibida oferecendo a ele a opção de "Entrar". No UsaTasker, o avaliador pode definir essa ação como opcional, considerando que essa opção não aparecerá para

Figura 13: Opções de desconto da oferta selecionada no *website* Groupon

usuários já logados. Para incluir o botão "Entrar" no caminho ótimo da tarefa, o avaliador deve selecionar as opções do COP conforme ilustra a Figura 14, informando que esse botão existe somente dentro do *container* destacado em vermelho, na página aberta atualmente.

Figura 14: Botão para logar no sistema do *website* Groupon

Os retângulos azuis na Figura 15 destacam os objetos da interface que podem ser manipulados antes da finalização da compra, alterando as opções de compra ou realizando o cadastro de um novo usuário. Portanto, tais ações devem ser definidas pelo avaliador como opcionais no UsaTasker. Além disso, como os objetos e os *containers* são únicos na página, o avaliador deve definir as opções do COP conforme ilustra a figura. Como cada compra possui a sua própria página dentro da aplicação, a opção "em várias páginas" do COP torna-se necessária para que sejam considerados os caminhos alternativos de todas as ofertas do *website*.

A Figura 16 destaca a ação que finaliza a compra de uma oferta, sendo necessário preencher as informações referentes ao cartão de crédito e também o clique no botão *Completar compra*. Todas as ações destacadas pelos retângulo azuis são obrigatórias,

Figura 15: Compra da oferta selecionada - opções de compra no *website* Groupon

exceto a data de expiração do cartão, portanto devem ser definidas dessa maneira no UsaTasker. Como a página de finalização da oferta é única na aplicação, a opção "nesta página" do COP torna-se mais conveniente para incluir estas ações no caminho ótimo da tarefa.

Resumindo a definição dessa tarefa no UsaTasker, tem-se o esquema apresentado na Figura 17, na qual é possível observar as opções de cada ação da tarefa, sendo que as linhas tracejadas definem ações opcionais e a cor de fundo amarela define que as ações não possuem uma relação de precedência, ou seja, podem ser realizadas em qualquer ordem. A linha azul representa um conjunto de ações que podem ser repetidas. Portanto, as ações "Clique no link ENTRAR", "Digitar o e-mail e a senha", "Enviar o formulário para entrar" e "Digitar dados pessoais" são opcionais para usuários já logados no sistema. Na Figura 17, também é possível observar que não é relevante a ordem das ações "Digite as informações da compra" e "Digite as informações do endereço", já que fica a critério do usuário a decisão de quais informações serão preenchidas primeiro. As demais ações são obrigatórias e possuem uma relação de precedência, sendo que a ação "Clique no link COMPLETAR COMPRA" finaliza a tarefa. A Figura 17 destaca ainda que as ações "Abrir

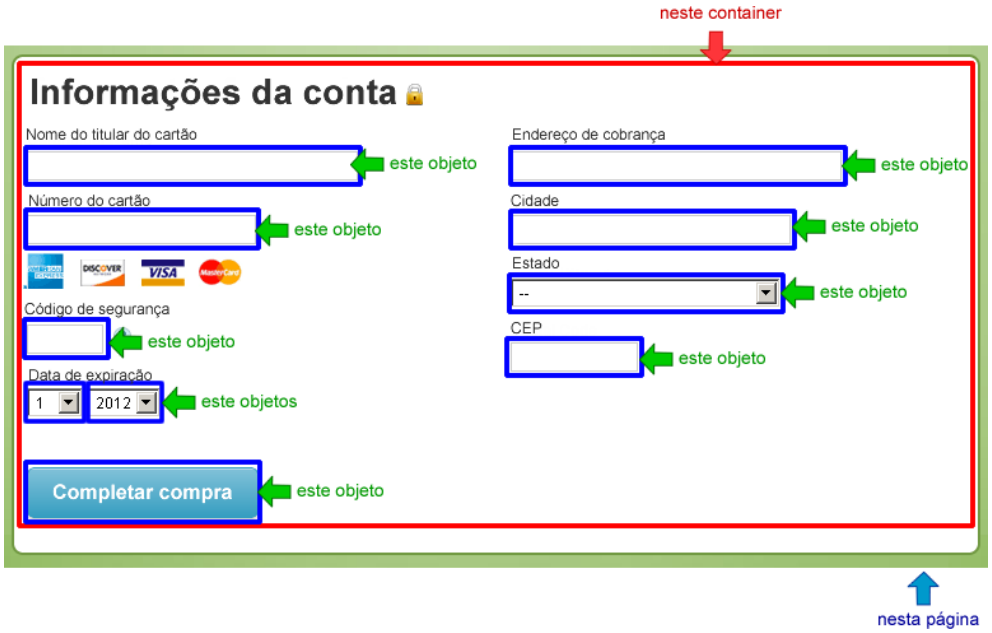

Figura 16: Finalização da compra de uma oferta no *website* Groupon

página OFERTAS", "Clique em uma oferta", "Abrir página OFERTA SELECIONADA" e "Clique no link COMPRAR" podem ser repetidas nesta ordem, caso o usuário deseje comprar várias ofertas no mesmo pedido.

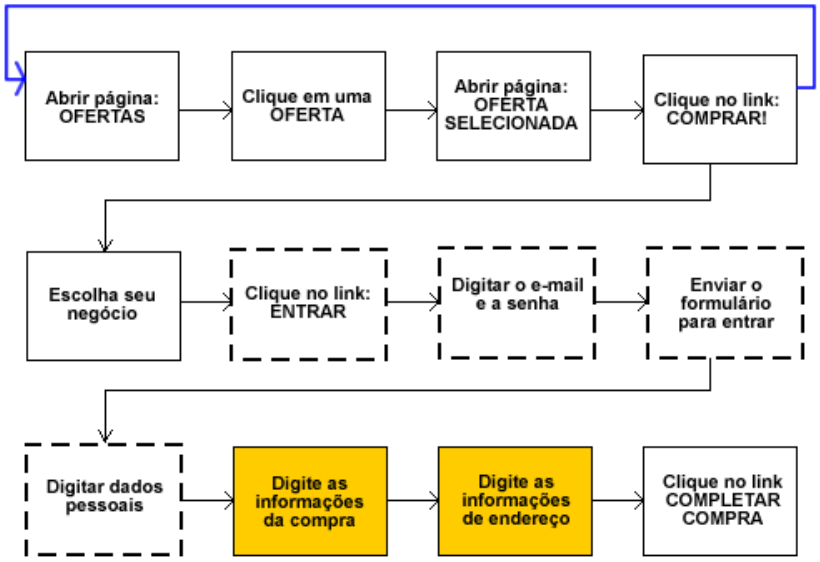

Figura 17: Representação da tarefa *Comprar uma oferta* segundo o UsaTasker

A tarefa exemplificada anteriormente ilustra a conexão entre as opções do modelo de interface COP e os recursos definidos no UsaTasker, que flexibiliza a definição de tarefas em aplicações Web modernas. Os recursos do UsaTasker estendem o COP para que seja possível definir automaticamente outros caminhos alternativos relacionados ao fluxo da interação do usuário final. Sem os recursos de ordem, ações opcionais e repetição do

UsaTasker, seria possível definir apenas tarefas de caminhos lineares, mas isso não reflete a maioria das aplicações Web reais.

Complementando a coleta automática e a definição de tarefas, a Seção 4.4 apresenta uma breve descrição do método utilizado no USABILICS para a análise de tarefas, sendo que os testes e a validação do método são apresentados no Capítulo 5

4.4 Análise de tarefas

Considerando a definição de usabilidade da ISO 9241-11 (ISO/IEC, 1998) e os cinco itens que a compõem (usuário, contexto de uso, eficácia, eficiência e satisfação), além dos conceitos envolvidos na avaliação remota e automática de usabilidade, é proposto no USABILICS um método para avaliar automaticamente a eficácia e a eficiência de interfaces de aplicações Web.

Para isso, o indicador de eficácia adotado é a completude de tarefas e os indicadores de eficiência são a taxa de erros, a completude e o tempo de realização das tarefas, assim como definem Frokjaer, Hertzum e Hornbaek (2000). Além disso, o contexto de uso é inferido a partir dos dados coletados sobre o sistema operacional, o navegador e a velocidade de transmissão utilizados pelo usuário.

A análise de tarefas dos usuários é feita encontrando-se a similaridade entre a sequência de eventos da tarefa definida pelo avaliador e as sequências de eventos das interações dos usuários. Para a comparação entre as sequências de eventos, a similaridade é calculada para cada subsequência de eventos da interação do usuário que iniciou a tarefa, pois o mesmo usuário pode ter executado a tarefa várias vezes durante a interação com a aplicação Web.

A identificação das subsequências de eventos que correspondem a uma tarefa é realizada pela comparação das opções de generalização do COP aplicadas aos eventos da interação do usuário e aos eventos que compõem as tarefas definidas para a aplicação Web analisada. Assim, é possível identificar quando um usuário iniciou uma determinada tarefa em sua interação com a interface, considerando os eventos que possivelmente iniciam o caminho ótimo definido pelo avaliador.

Todas as subsequências são comparadas à sequência de eventos da tarefa definida pelo avaliador, a fim de:

1. Gerar um índice de usabilidade que represente a eficiência e a eficácia da interface;

2. Identificar possíveis problemas de usabilidade na interface e, consequentemente, sugerir recomendações específicas ao avaliador.

O Capítulo 5 detalha a análise de tarefas e discute o índice de usabilidade proposto no USABILICS, e o Capítulo 6 apresenta o método usado pelo sistema para a identificação de problemas de usabilidade e, consequentemente, para as recomendações automáticas de possíveis soluções.

A Seção 4.5 compara o USABILICS aos diversos trabalhos desenvolvidos na área de avaliação remota e automática ou semi-automática de usabilidade. O USABILICS não se baseia em modelos pré-existentes reportados na literatura devido ao fato de não ter sido encontrado um modelo que permita uma abordagem genérica para a definição de tarefas sem a utilização de notações.

4.5 Trabalhos relacionados

Muitas pesquisas têm sido desenvolvidas na área de avaliação remota e automática ou semi-automática de usabilidade, em decorrência das diversas motivações existentes para isso, tais como: redução do custo da avaliação de usabilidade devido à redução do tempo de avaliação através de algoritmos e ferramentas, aumento da consistência dos erros descobertos principalmente na utilização de modelos de tarefas, redução da necessidade de avaliação por um especialista e aumento da cobertura de características

Para a avaliação remota de usabilidade, os trabalhos mais recentes utilizam *logs* para captura dos dados no navegador a fim de obter detalhes sobre os eventos gerados pelas ações do usuário durante a interação. WELFIT (SANTANA; BARANAUSKAS, 2010), WAUTT (RIVOLLI; MARINHO; PANSANATO, 2008), WebHint (VARGAS; WEFFERS; ROCHA, 2010), WAUTER (BALBO et al., 2005) e UsaProxy (ATTERER, 2006) são ferramentas que realizam esse modo de coleta de dados.

WAUTER necessita instalar um *software* no computador do cliente, mas isso impede o uso transparente da aplicação por parte do usuário. As ferramentas WELFIT, WebHint e WAUTT não capturam atributos CSS e outros atributos HTML específicos dos objetos, o que impossibilita a análise de tarefas em termos visuais dos elementos das páginas.

UsaProxy é um sistema baseado em *proxy* cuja captura dos dados é limitada à configuração do navegador do usuário ou de um *proxy*. Este sistema apresenta limitações associadas à identificação de elementos manipulados durante a interação do usuário por

utilizar apenas o atributo HTML *id*, característica que não permite tratar elementos genericamente para a análise de tarefas.

As ferramentas WELFIT, WebHint e WAUTT utilizam métodos para análise dos *logs* capturados. WELFIT utiliza um algoritmo sobre grafos para identificar padrões nas ações do usuário em uma página individualmente, porém, comparando com a abordagem adotada neste trabalho, essa ferramenta não considera o caminho entre as páginas de uma aplicação Web. Isso é uma limitação relevante tendo em vista que as aplicações Web geralmente são estruturadas em diversas páginas.

WAUTER e WebHint analisam *logs* baseando-se na análise de tarefas. WAUTER utiliza uma notação para definição de tarefas e heurísticas pré-definidas para comparação entre as ações realizadas pelos usuários e o modelo de tarefas da aplicação Web. Entretanto, diferentemente do USABILICS, WAUTER não oferece uma forma intuitiva para definição de tarefas e oferece limitações para a definição de possíveis caminhos de uma tarefa em aplicações Web dinâmicas.

Assim como UsaProxy, a ferramenta WebHint não utiliza notações para a definição de tarefas, permitindo que uma tarefa seja definida pelo avaliador apenas navegando na interface. O trabalho publicado sobre esta ferramenta cita o uso de um modelo de tarefas (VARGAS; WEFFERS; ROCHA, 2010), o qual, entretanto, não é suficientemente detalhado para que possa ser comparado com nossa abordagem.

Cybis (2009) realiza análise de *logs* orientada a transações, a fim de descobrir erros de execução evidentes, porém o artigo publicado sobre a pesquisa informa que os *logs* são gerados manualmente, o que impossibilita a realização dos testes de forma automática.

AWUSA (TIEDTKE; MARTIN; GERTH, 2002) também compara sequências de eventos a fim de encontrar as diferenças entre a ação do usuário e uma tarefa definida pelo avaliador, porém utiliza *logs* de servidor, que não possuem uma granularidade fina em relação às ações dos usuários. Para definir sequências de eventos como referência para comparação com as ações dos usuários, AWUSA utiliza recursos lógicos como páginas e *links* identificando-os a partir da análise do código HTML. Entretanto, essa abordagem dificulta a associação de diversos caminhos alternativos a uma única tarefa, ocasionando uma dificuldade na análise de tarefas

WebRemUSINE (PAGANELLI; PATERNò, 2002) é uma ferramenta de avaliação remota que faz captura e análise automáticas de *logs* de interação em aplicações Web para detectar problemas de usabilidade, baseando-se na comparação entre o modelo de tarefas da

aplicação Web e os caminhos realizados pelos usuários. Para definir o modelo de tarefas da aplicação Web, o avaliador deve utilizar o editor ConcurTaskTrees (PATERNò; MANCINI; MENICONI, 1997), bem como uma tabela específica de mapeamento entre as entradas de *log* e o modelo de tarefas. As desvantagens em relação ao USABILICS são: o usuário deve selecionar as tarefas que está realizando para que os eventos sejam associados à tarefa; a definição de tarefas utiliza notações, o que força o avaliador, possivelmente o desenvolvedor da aplicação, a conhecê-las para definir as tarefas em uma interface; a necessidade de um *plug-in* Java para salvar os arquivos de *log* das interações dos usuários em um servidor; e a redução do espaço útil da tela para exibir as tarefas disponíveis.

4.6 Considerações finais

Este Capítulo apresentou os módulos do USABILICS, que são baseados no modelo de interface COP e, de forma integrada, permitem a avaliação remota e automática de usabilidade.

O UsaTasker, módulo para definição de tarefas, permite definir tarefas de forma intuitiva na própria interface da aplicação. Além disso, os recursos de ações opcionais, ações repetidas e ações sem relação de precedência do UsaTasker permitem a definição de tarefas além daquelas que possuem caminho linear.

Foram apresentados também os trabalhos relacionados ao USABILICS, a fim de compará-los a proposta deste trabalho, e foi possível verificar que o USABILICS traz contribuições significativas na forma de definição de tarefas e no nível de detalhes da coleta automática.

Na análise de tarefas do USABILICS, é proposta uma medida de similaridade entre as sequências de eventos dos usuários e a sequência de eventos definida pelo avaliador. Essa medida resulta das opções do modelo de interface COP recebeu o nome de "índice de usabilidade", por refletir a eficácia e a eficiência na execução da tarefa. Para a validação da análise de tarefas e do índice de usabilidade proposto, foram desenvolvidos diversos experimentos, que são apresentados no Capítulo 5.

Aliada ao índice de usabilidade e à comparação das sequências de eventos, está a identificação de problemas de usabilidade. Essa identificação se baseia no conceito de ações erradas (do inglês, *wrong actions*) e permite detectar os passos mais e menos críticos das tarefas. A partir de padrões de interação identificados pelo USABILICS, são apresentadas algumas recomendações automáticas em relação aos elementos da interface. O Capítulo 6

apresenta o método de identificação de problemas de usabilidade e as recomendações, além de descrever os experimentos realizados em aplicações reais para a validação da proposta.

5 Análise de tarefas

Considerando o cenário atual da Web, já discutido anteriormente, e o curto espaço de tempo entre a concepção e a implementação de aplicaçoes Web, surge a necessidade de permitir que avaliadores inexperientes (possivelmente os desenvolvedores) consigam avaliar a usabilidade das interfaces rapidamente. Para isso, não é viável que a análise de tarefas resulte em grandes volumes de informações que exijam um alto nível de conhecimento dos avaliadores. No intuito de prover uma maneira rápida de acompanhar a avaliação de usabilidade das interfaces, é proposto no USABILICS um "índice de usabilidade", derivado do modelo de interface COP, que reflete a eficácia e a eficiência das tarefas em relação às interações dos usuários. Assim, os avaliadores possuem uma única informação que resume o estado atual do "nível de usabilidade" da aplicação, permitindo que sejam comparadas diferentes aplicações e também verificar a usabilidade de aplicações após a readaptação da interface, por exemplo.

Para isso, foram realizados diversos experimentos a fim de validar o método proposto no sistema. Todos os experimentos foram realizados em aplicações Web reais e, para cada uma delas, foram selecionadas as tarefas mais importantes, de acordo com o seu domínio.

Portanto, este capítulo apresenta o desenvolvimento do índice de usabilidade, os detalhes dos experimentos realizados e os resultados obtidos. A Seção 5.1 mostra como o índice de usabilidade foi derivado a partir do modelo de interface COP e apresenta a validação do índice como uma semi-métrica. A Seção 5.2 detalha os experimentos realizados para validação do sistema como um todo, enfatizando, porém, a validade do índice de usabilidade proposto.

5.1 Índice de usabilidade

Para quantificar a eficácia e a eficiência das interfaces Web, foi desenvolvida uma medida de usabilidade para as tarefas. Tal medida considera a similaridade entre os eventos realizados pelo usuário e os eventos definidos pelo avaliador (o caminho ótimo).

Para a definição de tarefas, eventos como *mouseover*, *mouseout*, *mousemove*, entre outros, não são considerados, porque eles não correspondem a decisões do usuário final, porque são ações que não alteram o estado da aplicação. Entretanto, esses eventos necessitam ser capturados durante a interação do usuário, pois são importantes para a identificação de problemas de usabilidade. A Figura 18 exemplifica os eventos coletados durante a interação do usuário e os eventos coletados na definição do caminho ótimo de uma tarefa. Os eventos representados pelos círculos brancos são aqueles que não existem no caminho ótimo da tarefa e que podem, portanto, representar ações erradas durante a execução de uma tarefa.

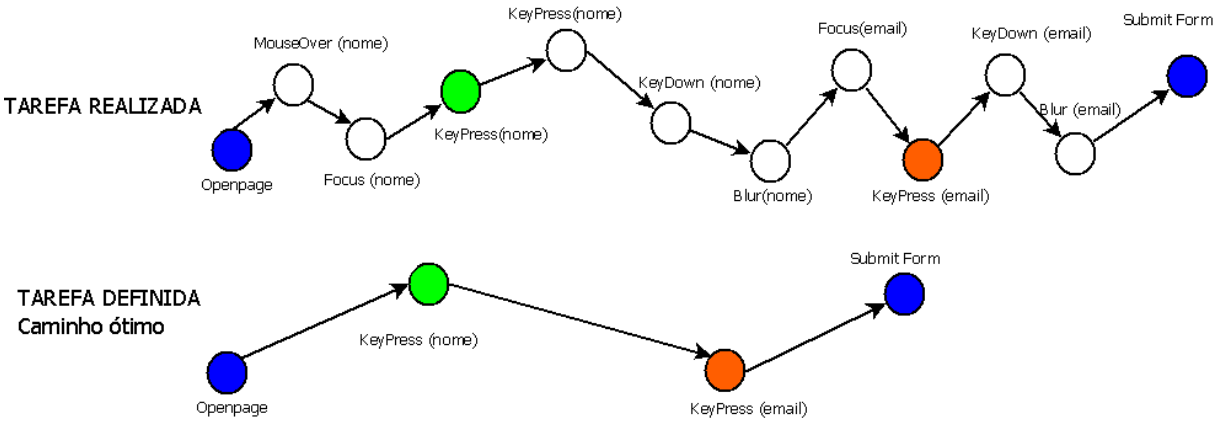

Figura 18: Comparação entre os eventos realizados pelo usuário e os eventos capturados na definição de uma tarefa

Portanto, considerando que (i) o conjunto de informações coletadas que necessitam ser comparadas é ligeiramente diferente, e (ii) o processo de comparação leva em conta as opções de generalização do modelo de interface COP, são usados quatro fatores principais para calcular a similaridade entre dois eventos:

1. **Evento:** o tipo do evento;
2. **Objeto:** o objeto no qual o evento ocorreu;
3. ***Container*:** o *container* do objeto no qual o evento ocorreu; e
4. **Página:** a página na qual o evento ocorreu.

Esses fatores são importantes, por exemplo, para comparar um evento *mouseover* relacionado à interação do usuário e um evento *click* existente no caminho ótimo da tarefa. Apesar de não serem idênticos, esses eventos são similares, pois o posicionamento do *mouse* sobre um objeto pode indicar o interesse do usuário em clicar no objeto. Esse

exemplo mostra que fornecer uma resposta binária (verdadeiro ou falso) para a comparação entre dois eventos não é apropriada.

Portanto, quando é comparado um evento realizado pelo usuário a um evento correspondente definido pelo avaliador, é produzido um valor de similaridade entre 0 e 1. No USABILICS, esse valor é baseado na importância de cada conceito do modelo COP associado ao evento. Portanto, foram aplicados pesos para esses conceitos, como segue: 0.1 para a página, 0.3 para o *container* e 0.5 para o objeto. Foi aplicado ainda o peso 0.1 para o evento realizado. Assim, se o evento é realizado na página correta, é computado o valor 0.1; se for realizado no *container* correto, soma-se 0.3; se for realizado no objeto correto, soma-se 0.5; e, se o tipo do evento for correto (um clique, por exemplo), soma-se 0.1. Neste caso o resultado é 1, isto é, o evento foi realizado corretamente.

É importante notar que os pesos relacionados aos *containers* e aos objetos têm os maiores valores, porque é importante distinguir claramente os eventos realizados próximos ao objeto alvo e aqueles realizados em *containers* ou em outros objetos longe do alvo (considerados não semelhantes aos eventos definidos).

A similaridade entre dois eventos resulta em um valor entre 0 e 1, 0 quando nenhum dos fatores é equivalente e 1 quando todos os fatores são equivalentes.

Na comparação entre a sequência de eventos do usuário e a sequência de eventos da tarefa definida pelo avaliador, quando a similaridade entre dois eventos resulta em 1, o contador de completude da tarefa é incrementado.

No USABILICS, a métrica da eficácia da usabilidade é a porcentagem de completude da tarefa, ou seja, quantos passos do caminho ótimo da tarefa foram completados pelo usuário. Já a eficiência é medida pela similaridade entre os eventos do caminho ótimo e os eventos da interação do usuário, porque esse método considera a taxa de erros da interação do usuário, conforme sugere Frokjaer, Hertzum e Hornbaek (2000).

Generalizando a similaridade entre eventos, tem-se uma medida de similaridade entre a sequência de eventos realizados pelo usuário e a sequência de eventos definidos na tarefa. Essa similaridade entre sequências de eventos é dada pela fórmula:

$sim(Si, Sj) = (SS/QE) * PC$

Onde:

- Si é a sequência de eventos da interação do usuário;
- Sj é a sequência de eventos da tarefa definida pelo avaliador;

- SS é a soma da similaridade entre os eventos de Si e os eventos de Sj;
- QE é a quantidade de eventos de Si; e
- PC é a completude da tarefa.

O cálculo de $sim(Si, Sj)$ resulta em um valor entre 0 e 1, pois, como a similaridade entre dois eventos retorna um valor entre 0 e 1, o valor mínimo para a soma da similaridade dos eventos (SS) é 0 e o valor máximo é igual a quantidade de eventos (QE) realizados pelo usuário em uma interação. Além disso, PC é um valor percentual que mantém os limites mínimo e máximo de *sim(Si, Sj)*.

Assim como a medida de similaridade proposta por Oh e Kim (2004), é possível converter a medida proposta em dissimilaridade com uma simples transformação:

$$Dissim(Si, Sj) = 1 - sim(Si, Sj)$$

Sendo assim, verifica-se que a dissimilaridade entre as sequências Si e Sj satisfaz as condições de uma semi-métrica, que é uma forma de definir uma distância entre dados (MOEN, 2000).

De acordo com Moen (2000), seja O um conjunto de dados, e d uma medida de distância entre os dados no conjunto O. A medida d é chamada de semi-métrica, se as seguintes condições forem satisfeitas:

$d(\gamma i, \gamma j) \geq 0,$
$d(\gamma i, \gamma i) = 0 \, ou \, d(\gamma i, \gamma j) = 0 \, se \, e \, somente \, se \, \gamma i = \gamma j$
$d(\gamma i, \gamma j) = d(\gamma j, \gamma i)$

$para \, todo \, \gamma i, \gamma j \, pertencente \, ao \, conjunto \, O.$

Como $0 \leq sim(Si, Sj) \leq 1$, $dissim(Si, Sj)$ é maior que 0 e a primeira condição é satisfeita. Como $sim(Si, Si) = 1$, a segunda condição é satisfeita. A terceira condição também é satisfeita porque $sim(Si, Sj) = sim(Sj, Si)$. Portanto, a medida de similaridade proposta satisfaz condições de uma semi-métrica.

A partir da similaridade entre sequências ($sim(Si, Sj)$), é possível deduzir o índice de usabilidade de uma tarefa por:

$$IndiceDeUsabilidade = (\sum sim(Si, Sj))/QI$$

Onde: QI é a quantidade de tarefas iniciadas pelos usuários.

A seção seguinte apresenta os experimentos realizados para a validação do índice de usabilidade proposto no USABILICS.

5.2 Validação do índice de usabilidade

Testes de usabilidade em laboratório vem sendo amplamente aceitos como uma forma eficiente para a descoberta de problemas de usabilidade em aplicações Web (TULLIS et al., 2002). Um dos motivos dessa aceitação é o pequeno número de usuários exigido por testes que realizam um conjunto de tarefas usando uma aplicação ou um protótipo (NIELSEN, 2000b). Para a avaliação, um ou mais observadores anotam o tempo que o usuário levou para completar cada tarefa, se ele obteve sucesso e quaisquer comentários relevantes, além do retorno subjetivo de cada usuário.

Portanto, para validar o índice de usabilidade proposto, foi realizado um teste tradicional em laboratório e, simultaneamente, um teste remoto através do USABILICS. O objetivo do uso desse método foi comparar os resultados do teste em laboratório com os resultados fornecidos pelo sistema.

Para a validação, foram selecionados 7 *websites* de diversas áreas de negócios, nos quais foi implantada a ferramenta para coleta automática de dados. Os *websites* selecionados foram:

- (a) um *blog* pessoal com assuntos sobre tecnologia da informação (http://www.4learn.pro.br/guarino/blog);
- (b) um ambiente virtual de aprendizagem a distância (http://www.4learn.pro.br/);
- (c) um *blog* pessoal com poesias (http://www.semprealegria.com);
- (d) um *website* de publicação de estudos (http://www.viveremverdade.com.br);
- (e) um institucional de um clube (http://www.itaguara.com);
- (f) uma página pessoal (http://www.minhavictoria.net); e
- (g) um portal de uma revista regional (http://www.anunciame.com.br).

Tabela 2: Tarefas selecionadas para a validação nos *websites* estudados

Website	Tarefas	Depth
(a)	Comentar qualquer post do *blog*	6
(b)	Corrigir uma atividade de qualquer aluno	5
	Enviar um e-mail aos alunos de uma turma	5
(c)	Encontrar uma poesia específica	1
	Copiar o selo digital com a marca do *blog*	2
(d)	Enviar um comentário aos autores	5
	Cadastrar-se para receber os estudos	5
(e)	Encontrar um evento específico	2
(f)	Acessar as fotos de uma viagem específica	2
(g)	Acessar a página de um dos autores	3

Todos são *websites* dinâmicos. Além disso, foram desenvolvidos utilizando alguma ferramenta CMS, exceto o ambiente virtual de aprendizagem e o *website* de publicação de estudos.

Consequentemente, foram identificadas tarefas importantes nos *websites* estudados, as quais foram definidas no USABILICS, formando o modelo de tarefas das aplicações. A Tabela 2 apresenta as tarefas selecionadas com seus respectivos níveis de profundidade, de acordo com o número de ações esperadas no caminho ótimo para completude da tarefa, baseado na definição de Melguizo, Oostendorp e Juvina (2007). Na tabela, é possível observar que foram definidas duas tarefas para os *websites* (b), (c) e (d).

Em seguida, foram escolhidos dois *websites* entre aqueles que possuem tarefas cujo nível de profundidade é maior: o *blog* pessoal com assuntos sobre tecnologia da informação (a) e o ambiente virtual de aprendizagem a distância (b). Os passos da tarefa *Corrigir uma atividade* do *website* (b) foram apresentados no exemplo da Seção 3.2.1.

Satisfazendo os requisitos para testes de usabilidade em laboratório (NIELSEN, 2000b), foram selecionados 14 participantes cujas idades variam entre 19 e 42 anos, sendo 6 estudantes, 5 professores, 1 técnico industrial, 1 analista de suporte e 1 estagiário em informática.

Para a realização dos testes, os participantes receberam um questionário com questões sobre os seus perfis e suas experiências com recursos de informática.

Todos os participantes julgaram ter experiência boa, ótima ou excelente com *softwares* e Internet, e 70% informaram que acessam o computador principalmente para navegar na Web. Cerca de 90% dos participantes acessam a Internet diariamente sendo que 60% acessam sites de pesquisa e apenas 21% acessam *blogs* e notícias frequentemente. Dos 14

participantes do teste, 5 participantes foram selecionados aleatoriamente para realizarem as tarefas do *website* (a) e 5 para o *website* (b), sendo que 4 informaram que nunca acessaram anteriormente o *website* que utilizaram no teste, 2 costumavam acessar três vezes por semana, 2 costumavam acessar uma vez por semana e 2 raramente acessavam.

Os testes foram realizados em um computador colocado em uma sala isolada e as interações dos usuários foram gravadas por um *software* livre específico para gravação de tela. Além disso, foi habilitado o microfone do computador para capturar os comentários dos usuários durante a navegação, a fim de reconhecer comentários relevantes. Esses procedimentos foram comunicados aos participantes da pesquisa antes de sua realização.

Após a realização do teste, cada usuário avaliou a facilidade de encontrar informações nos *websites* estudados, sendo que 50% julgaram muito fácil, 40% julgaram fácil e 1 participante (10%) avaliou a interface do *website* (b) como difícil para navegar, comentando que *'poderia ser mais fácil de encontrar informações, os links poderiam estar mais destacados'*.

Após a realização dos testes, foram analisados os vídeos e os áudios gravados pelo *software* durante a navegação dos usuários e verificou-se que, dos 5 participantes que avaliaram o *website* (a), apenas 1 participante não completou a tarefa de enviar um comentário.

No Web site (b), todos os participantes obtiveram sucesso na tarefa *Enviar um e-mail aos alunos de uma turma*, porém 2 participantes não conseguiram cumprir a tarefa *Corrigir uma atividade*, de forma que um usuário não encontrou o *link* que direcionava à página de correção de atividades e outro usuário não enviou o texto da correção.

No *website* (a), a principal ação dos usuários que impediu a realização da tarefa com maior eficiência foi a rolagem da página em busca do formulário de comentários do post. Já no *website* (b), as principais ações foram cliques em *links* incorretos e, consequentemente, a abertura de páginas que não estavam relacionadas às tarefas.

Simultaneamente aos testes em laboratório, o sistema USABILICS foi habilitado para capturar todos os eventos das interações dos usuários com os *websites*, a fim de verificar a representatividade dos resultados da análise de tarefas baseada no modelo de interface proposto.

Todos os participantes utilizaram o browser Firefox 3.6 para navegar nos *websites*, porém o funcionamento do sistema foi validado por meio de outros testes nos browsers Internet Explorer 6 e 7, Firefox 2+, Opera 9+, Google Chrome 9+ e Safari 3.

A análise de tarefas através do algoritmo de similaridade de sequência de eventos implementado no USABILICS, apresentou os seguintes resultados:

Para o *blog* sobre assuntos de tecnologia da informação:

- **Tarefa:** Comentar qualquer post do *blog*
- **Índice de usabilidade:** 0,7993
- **Observações:** As principais ações erradas (52%) ocorreram entre a abertura da página do post e o preenchimento da primeira caixa de texto, sendo que 20% dos eventos referem-se à rolagem da página. Verificando a estrutura das páginas dos posts, constatou-se que o formulário de comentário localiza-se no rodapé da página.

Para o AVA 4Learn:

- **Tarefa:** Corrigir uma atividade
- **Índice de usabilidade:** 0,1348
- **Observações:** Dois usuários não completaram a tarefa, sendo que o índice de similaridade das interações foram 0,0202 e 0,0915 e as taxas de completude da tarefa foram 14% e 85%, respectivamente. As principais ações erradas ocorreram entre a abertura da página que contém as atividades dos alunos e o clique no *link* "Corrigir" (53%), e entre a abertura da página do sistema e o clique no *link* que direciona à página de correção de atividades (28%), sendo que 2% referem-se a cliques e, consequentemente, à abertura de páginas não relacionadas à tarefa.

- **Tarefa:** Enviar e-mail aos alunos de uma turma
- **Índice de usabilidade:** 0,6844
- **Observações:** As principais ações erradas (62%) ocorreram entre a seleção dos alunos que receberão o e-mail e o preenchimento da primeira caixa de texto, sendo que a maior parte dos eventos ocorridos referem-se à rolagem de página. Analisando a página em que ocorre esse passo da tarefa, detectou-se que o formulário de envio de e-mail está localizado no rodapé da página.

Observando os resultados da tarefa *Comentar qualquer post do blog* do *website* (a), constata-se que a execução dessa tarefa por parte dos usuários é realizada de forma satisfatória, sendo que a similaridade com o caminho ótimo definido na tarefa é de aproximadamente 80%. Esse resultado corresponde aos resultados apresentados pelo teste em

laboratório, no qual verificou-se que todos os usuários encontraram facilmente os posts do *blog* e os *links* que direcionavam ao formulário de comentário.

O índice de usabilidade apresentado para a tarefa *Corrigir uma atividade* do *website* (b) demonstra que essa tarefa é realizada de maneira insatisfatória, pois a similaridade entre as interações dos usuários e o caminho ótimo definido pelo avaliador é de aproximadamente 13%. Esse índice sugere que há muitos problemas de usabilidade na interface e, além disso, as ações erradas identificadas pelo USABILICS apontam para as dificuldades detectadas no teste em laboratório, que ocorreram na procura pelo *link* que direcionava ao formulário de correção e na procura pelo *link* que direcionava à página de correção de atividades, sendo que um usuário não completou a tarefa por este motivo.

Os resultados apresentados pelo sistema para a tarefa *Enviar um e-mail aos alunos de uma turma* do *website* (b) demonstram que essa tarefa é realizada de forma satisfatória pelos usuários. Tal resultado foi comprovado na análise das interações do teste em laboratório, sendo que os usuários não encontraram dificuldades para cumprir a tarefa. Porém, o USABILICS ainda aponta uma possibilidade de melhoria na interface para otimizar a execução da tarefa, identificando a ocorrência de uma grande quantidade de eventos de barra de rolagem entre a abertura da página que contém a lista de alunos e o início do preenchimento do formulário.

Comparando os resultados do USABILICS e do teste em laboratório, é possível verificar que o índice de usabilidade proposto fornece um bom indicador da eficácia e da eficiência das interfaces das aplicações Web, refletindo o comportamento dos usuários na realização das tarefas.

Além disso, a análise dos eventos realizados entre os passos do caminho ótimo das tarefas permite identificar fatores que influenciam de forma negativa a navegação dos usuários, impedindo-os de realizarem as tarefas de forma mais eficiente. A identificação do carregamento de páginas não relacionadas à tarefa sugere, por exemplo, que os *links* não estão identificados de forma clara e a detecção de uma grande quantidade de eventos de rolagem de página sugere que o posicionamento dos elementos da interface pode ser otimizado.

5.3 Considerações finais

Este capítulo apresentou o método utilizado pelo USABILICS para a análise de tarefas, que se baseia na similaridade entre o caminho ótimo da tarefa e os caminhos realizados

pelos usuários. Essa similaridade gera um índice de usabilidade que se baseia nos conceitos do modelo COP e reflete a eficácia e a eficiência das tarefas. Também foram apresentados os experimentos realizados para a validação da análise de tarefas e do índice de usabilidade em aplicações Web reais, constatando que o método proposto é eficiente para a avaliação remota de usabilidade, pois beneficia avaliadores inexperientes na compreensão do "nível de usabilidade" de suas aplicações.

Além disso, os experimentos apresentados neste capítulo mostram a validade do índice de usabilidade proposto no USABILICS. Dessa forma, tem-se uma medida confiável da eficácia e da eficiência da usabilidade das tarefas de aplicações Web.

As discussões e os resultados apresentados neste capítulo foram publicados em um artigo completo no Simpósio Brasileiro de Fatores Humanos em Sistemas Computacionais - IHC 2011 (VASCONCELOS; BALDOCHI JR., 2011).

Mesmo que o índice de usabilidade forneça uma maneira rápida e confiável para acompanhar a usabilidade de aplicações Web, essa informação não é suficiente para explicitar os problemas de usabilidade identificados. Além disso, descrever os problemas de usabilidade para avaliadores inexperientes pode ser pouco eficaz. Portanto, utilizando a medida de similaridade e o método para identificação de problemas de usabilidade apresentados neste capítulo, o USABILICS foi estendido para fornecer recomendações pontuais aos avaliadores, a fim de propor soluções para a readaptação manual da interface.

O Capítulo 6 apresenta as recomendações como resultado da análise de tarefas, bem como os experimentos realizados para validá-las.

6 Recomendações automáticas

Para a avaliação remota e automática de usabilidade, as três fases da avaliação descritas por Balbo (1995) devem ser automatizadas: a coleta, a análise e a crítica. Os capítulos anteriores apresentaram a automatização da coleta de dados e da análise de tarefas, e este capítulo apresenta o método implementado no USABILICS para a identificação de problemas de usabilidade e, consequentemente, para as recomendações automáticas que sugerem soluções para a readaptação manual da interface, automatizando a crítica da avaliação de usabilidade.

A Seção 6.1 apresenta o conceito de ações erradas proposto por Vermeeren et al. (2008) e como esse conceito auxilia a identificação de problemas de usabilidade pelo USABILICS. Essa seção apresenta também as recomendações derivadas a partir de padrões identificados na análise de tarefas, e a Seção 6.2 detalha os experimentos realizados e os resultados obtidos após a readaptação das interfaces das aplicações estudadas.

6.1 Identificação de problemas de usabilidade

O índice de usabilidade proposto neste trabalho é uma informação valiosa para desenvolvedores Web, porque aponta as tarefas que apresentam problemas de usabilidade e, mais importante ainda, a extensão destes problemas.

No entanto, para desenvolvedores inexperientes em avaliação de usabilidade, apontar um problema pode não ser suficiente, uma vez que esses desenvolvedores podem não conseguir interpretar corretamente a informação e executar melhorias para resolverem o problema. A fim de sanar essa necessidade, o USABILICS permite, através da análise de tarefas, a identificação de problemas de usabilidade e sugere medidas para enfrentar os problemas apontados.

Como apontado por Vermeeren et al. (2008), quando são comparadas as sequências de eventos que compõem uma tarefa, é possível encontrar três diferentes situações que

indicam a ocorrência de ações erradas (do inglês, *wrong actions*):

1. uma ação que não pertence à sequência correta de ações (caminho ótimo);
2. uma ação é omitida da sequência; e
3. uma ação dentro da sequência é substituída por outra ação.

Considerando as definições de Vermeeren et al. (2008), o USABILICS identifica as três situações na análise de tarefas. Para tanto, funcionalidades adicionais foram adicionadas ao sistema a fim de manter a informação detalhada sobre interações abortadas e para registar o tempo necessário para executar cada subsequência de uma determinada tarefa.

Com estas extensões, é possível identificar as subsequências de uma tarefa em que os usuários finais encontraram problemas. Além disso, é possível identificar objetos e *containers* associados a cada problema e, portanto, fornecer recomendações detalhadas a fim de suprimir ou, pelo menos, minimizar o problema.

Para a identificação dos problemas de usabilidade, o algoritmo de análise de tarefas contabiliza as ações erradas em cada passo do caminho ótimo da tarefa, associando-as aos tipos de eventos executados, aos objetos e *containers* manipulados e ao tempo gasto para completar cada passo da tarefa. Assim, é possível obter um valor numérico que reflete a quantidade de ações erradas, bem como o tempo utilizado pelos usuários, em cada passo. Quanto maiores o tempo e a quantidade de ações erradas, provavelmente maiores foram as dificuldades encontradas pelos usuários.

No intuito de recomendar possíveis soluções para os avaliadores, foram testadas aplicações Web reais, comparando as sequências coletadas com as respectivas tarefas definidas. Então, observou-se uma correlação entre o baixo índice de usabilidade e a presença de ações erradas. Portanto, o USABILICS identificou padrões de erros associados aos cliques em *links*, à abertura de novas páginas, à rolagem das páginas e à interação com formulários. Esses padrões foram classificados em seis categorias:

1. Cliques em *links* que não pertencem ao caminho ótimo da tarefa;
2. Abertura de páginas que não pertencem ao caminho ótimo da tarefa, antes de acessar um *link*;
3. Abertura de páginas que não pertencem ao caminho ótimo da tarefa, antes de acessar um formulário;

4. Uso excessivo da barra de rolagem antes de preencher um formulário;
5. Uso excessivo da barra de rolagem antes de acessar um link;
6. Intervalo de tempo grande entre a requisição de uma página e o evento que indica o carregamento completo da mesma.

Sempre que um desses padrões é encontrado, o algoritmo de análise de tarefas do USABILICS identifca o elemento da interface que mais provavelmente está envolvido com o problema correspondente. Observando os padrões de erro detectados pelo algoritmo e os passos das tarefas testadas, foram constatados problemas na identificação e no posicionamento dos objetos na interface. Como resultado dessa observação, foi proposta uma recomendação para cada padrão de erro detectado.

As recomendações propostas estão enumeradas de R1 a R6, de maneira que R1 é a recomendação para o padrão 1, R2 para o padrão 2, e assim sucessivamente.

- **(R1)** Destacar o link X de outros links na página P. Use ícones.
- **(R2)** Mudar a localização do link X na página P. Coloque-o em uma página que é frequentemente acessada pelos usuários.
- **(R3)** Mudar a localização do formulário F na página P. Coloque-o em uma página que é frequentemente acessada pelos usuários.
- **(R4)** Posicionar o formulário F no topo da página P. Identifique o formulário com um título adequado.
- **(R5)** Destacar o link X na página P. Tente posicionar o link no topo da página.
- **(R6)** Otimizar o carregamento da página P.

A recomendação R1 derivou da ocorrência do padrão 1 devido à existência de diferentes *links* na mesma página do *link* alvo, confundindo os usuários pela má formatação do objeto ou pela terminologia utilizada.

As recomendações R2 e R3 derivaram dos padrões 2 e 3, respectivamente, devido à ocorrência de cliques em *links* que abriram páginas irrelevantes para a tarefa realizada.

As recomendações R4 e R5 derivaram dos padrões 4 e 5, respectivamente, por causa da excessiva execução de eventos de rolagem de página antes do acesso ao objeto alvo.

A recomendação R6 derivou do padrão 6 a partir da observação do tempo contabilizado pelo algoritmo de análise de tarefas.

A fim de reconhecer a validade das recomendações propostas, foram realizados experimentos com duas aplicações Web. Em seguida, foi solicitado aos desenvolvedores das aplicações que readaptassem a interface conforme as recomendações, objetivando medir o impacto das recomendações na usabilidade das aplicações. Os experimentos e resultados são apresentados na Seção 6.2.

6.2 Validação das recomendações automáticas

Com objetivo de identificar a relação entre as ações erradas e os conceitos do modelo COP, foram realizados experimentos com duas aplicações Web: (I) o AVA 4Learn e (II) um *website* para publicação de artigos sobre tecnologia da informação. Durante 52 dias, foram monitoradas quatro tarefas, duas para cada aplicação. O resultado da coleta de dados foi:

- **Aplicação I:** 1.019 tentativas de realização de tarefas com 258.802 entradas de *log*;
- **Aplicação II:** 1.605 tentativas de realização de tarefas com 474.437 entradas de *log*;

As Seções 6.2.1 e 6.2.2 apresentam os resultados dos experimentos para cada aplicação.

6.2.1 Ambiente Virtual de Aprendizagem a Distância

No AVA 4Learn (aplicação I), foram selecionadas as seguintes tarefas para os experimentos:

1. Corrigir a atividade de um aluno (usuário: Docente). Esta tarefa está detalhada na Seção 3.2.1.
   (a) Selecionar a opção *Corrigir* no menu principal
   (b) Selecionar uma turma para corrigir as atividades e clicar no link *Corrigir*
   (c) Escolher a atividade de um aluno e clicar no link *Corrigir*
   (d) Atribuir um nota de avaliação e preencher a resposta ao aluno

(e) Clicar no botão *Corrigir* para enviar a nota e o comentário ao aluno

2. Enviar um e-mail para uma turma de alunos (usuário: Docente)

   (a) Selecionar a opção *Informações das turmas* no menu principal

   (b) Selecionar uma turma

   (c) Marcar os alunos que receberam a mensagem

   (d) Preencher as caixas de texto *assunto* e *mensagem*

   (e) Clicar no botão *Enviar e-mail*

Para a tarefa *Corrigir a atividade de um aluno*, foi detectado que apenas 16% dos usuários que iniciaram a tarefa conseguiram completá-la, resultando em um índice de usabilidade baixo (0,1033).

Além disso, o USABILICS apontou que 36% dos usuários abortaram a tarefa no passo (b) e a mesma porcentagem de usuários abortou no passo (c). Em ambos os casos, o sistema identificou que as ações erradas realizadas pelos usuários relacionavam-se à rolagem da página, o que indica que os usuários podem ter encontrado dificuldades para encontrar um elemento da interface.

Portanto, o sistema escolheu a recomendação R5: *Destacar o link "Corrigir" na página "index.php". Tente posicionar o link no topo da página.*

Para a tarefa *Enviar um e-mail para uma turma de alunos* do AVA, o sistema identificou que usuários levaram cerca de 11 segundos para clicar no *link* que inicia a tarefa, apontando que o *link Informações das turmas* não está destacado dos outros *links*, portanto a recomendação R1 foi selecionada. Além disso, no passo (b) desta tarefa, notou-se uma quantidade excessiva de eventos de rolagem de página associados ao preenchimento do formulário. Como resultado, a recomendação R4 também foi selecionada. O índice de usabilidade desta tarefa foi 0,3757, sendo que apenas 35% dos usuários completaram-na.

Após os testes, foi solicitado aos desenvolvedores do ambiente que readaptassem a interface conforme as recomendações. Então, os dados foram coletados novamente pelo USABILICS e foi executada a análise de tarefas, gerando os seguintes resultados.

- **Tarefa 1:** Corrigir a atividade de um aluno
- **Índice anterior:** 0,1033
- **Novo índice:** 0,4679

- **Observações:** As ações erradas encontradas anteriormente não apareceram nesta vez. Entretanto, o sistema identificou um novo problema relacionado ao carregamento da página que mostra a lista de disciplinas do docente, e recomendou R6.

- **Tarefa 2:** Enviar um e-mail para uma turma de alunos
- **Índice anterior:** 0,3757
- **Novo índice:** 0,5759
- **Observações:** O sistema identificou que foram reduzidas 99% das ações erradas relacionadas à rolagem da página, como resultado da recomendação R4. Semelhantemente, a implementação da recomendação R1 também solucionou o problema relacionado às ações erradas realizadas antes do clique no *link Informações das turmas*. Na realidade, a quantidade de ações erradas detectadas nesse passo da tarefa reduziu 62%.

6.2.2 Website de Publicação de Artigos

Para o *website* de artigos (aplicação II), foram definidas as seguintes tarefas nos experimentos:

1. Cadastrar-se para receber os artigos (usuário: Visitante)
   (a) Clicar no link *Cadastre-se para receber os artigos*
   (b) Preencher as caixas de texto *nome* e *e-mail*
   (c) Clicar no botão *Cadastrar*
2. Enviar um comentário para o autor de um artigo (usuário: Visitante)
   (a) Selecionar a opção *Comente* no menu principal
   (b) Preencher as caixas de texto *nome*, *e-mail*, *assunto* e *mensagem*
   (c) Clicar no botão *Enviar e-mail*

O índice de usabilidade calculado para a tarefa *Cadastrar-se para receber os artigos* do *website* de artigos foi 0,5359. Para esta tarefa, a taxa de completude foi 100%, entretanto foram identificadas ações erradas relacionadas a cliques e à rolagem de página no passo (a), e outras relacionadas à rolagem de página no passo (b).

Baseado nesses resultados, foram recomendadas:

- **R5:** *Destacar o link "Cadastre-se para receber os artigos" na página "artigo20.php". Tente colocar o link no topo da página*

- **R4:** *Posicione o formulário "Cadastre" no topo da página "artigo20.php". Identifique o formulário com um título adequado*

Em relação à tarefa *Enviar um comentário para o autor de um artigo*, apenas 50% dos usuários completaram-na. Como resultado, esta tarefa apresentou um índice de usabilidade de 0,3217. O baixo valor do índice é devido à grande quantidade de ações erradas encontradas no passo (b) da tarefa, principalmente relacionadas com o preenchimento do formulário. Da análise dessa tarefa, 47% das ações erradas estão relacionadas à rolagem de página e 31%, a cliques em objetos que não compõem o caminho ótimo da tarefa.

Baseado nesses resultados, o sistema recomendou:

- **R1:** *Destacar o link "Comente" dos outros links na página "index.php". Use ícones.*

- **R4:** *Posicione o formulário "FormComente" no topo da página "index.php?area=comente". Identifique o formulário com um título adequado*

Após a análise de tarefas deste *website*, o desenvolvedor foi contatado para readaptar a interface conforme as recomendações. Depois das atualizações, foram realizados novamente os mesmos experimentos. Os resultados obtidos na segunda rodada de testes são estes:

- **Tarefa 1:** Cadastrar-se para receber os artigos

- **Índice anterior:** 0,5359

- **Novo índice:** 0,6807

- **Observações:** A atualização realizada na interface reduziu 88% das ações erradas. Além dos bons resultados no índice e da redução das ações erradas, foi observado que o números de usuários cadastrados também aumentou após a readaptação da interface.

- **Tarefa 2:** Enviar um comentário para o autor de um artigo

- **Índice anterior:** 0,3217

- **Novo índice:** 0,6180
- **Observações:** As ações relacionadas a cliques em *links* fora do caminho ótimo não foram detectadas nesta vez. Além disso, a média do tempo para o início da tarefa reduziu em 3 segundos, validando, portanto, a eficácia da recomendação R1. As atualizações relacionadas ao formulário de comentário, como sugerida na recomendação R4, também foi eficaz para reduzir as ações associadas à barra de rolagem. Com as melhorias implementadas, o tempo para realizar a tarefa reduziu 25%. Apesar dos bons resultados, 20% dos usuários não foram capazes de cumprir plenamente a tarefa, o que explica o índice de 0,6180.

6.3 Considerações finais

Este capítulo apresentou as recomendações automáticas sugeridas pelo USABILICS a partir da análise de tarefas baseada nos conceitos do modelo de interface COP. No USABILICS, o algoritmo de análise de tarefas contabiliza as ações erradas da interação do usuário comparada ao caminho ótimo da tarefa. Assim, é possível identificar os passos da tarefa que são mais e menos críticos, ou seja, os passos em que os usuários encontraram mais e menos dificuldade.

O capítulo apresenta seis padrões de erro identificados pelo USABILICS na análise de tarefas, que dão origem a seis recomendações como possíveis soluções para problemas de usabilidade.

De forma complementar, os experimentos realizados para a validação das recomendações apontam que estas foram satisfatórias. No entanto, também é possível observar que muitas outras recomendações podem ser sugeridas, já que a coleta de dados possui muitos detalhes sobre os elementos da interface.

As discussões e os resultados apresentados neste capítulo foram publicados em um artigo completo no Simpósio de Computação Aplicada da ACM - ACM SAC 2012 (VASCONCELOS; BALDOCHI JR., 2012).

7 Conclusão

A usabilidade no cenário da Web vem ganhando importância, visto que uma interface atrativa e funcional pode representar a diferença entre o sucesso e o fracasso de uma aplicação. No entanto, avaliar a usabilidade de interfaces de forma rápida e eficaz ainda é um grande desafio. Por isso, no sentido de prover uma ferramenta mais robusta, capaz de oferecer maior flexibilidade para a avaliação de grandes aplicações Web, foi desenvolvido o USABILICS, um sistema para avaliação remota e automática de usabilidade de aplicações Web baseada em um modelo de interface.

As diversas ferramentas reportadas na literatura para avaliação remota e automática ou semi-automática de usabilidade apresentam limitações, as quais foram apresentadas e discutidas na Seção 4.5. Essas limitações tornam-se cruciais quando se trata de aplicações Web grandes e dinâmicas.

A essência do USABILICS é seu modelo de interface, o qual permite que o avaliador, possivelmente o desenvolvedor da aplicação, possa definir as tarefas que deseja avaliar de maneira simples e intuitiva, interagindo com a interface gráfica da aplicação.

Para permitir a definição das tarefas pelos desenvolvedores e a posterior captura dessas tarefas, quando realizadas pelos usuários, o USABILICS provê uma ferramenta para captura das interações que considera os atributos CSS dos elementos das páginas, permitindo tratá-los genericamente, o que facilita a definição de diversos caminhos alternativos para uma mesma tarefa.

Finalmente, para analisar os eventos coletados, é proposto um algoritmo que avalia a similaridade de sequências de eventos, resultando em uma métrica da eficiência e da eficácia da usabilidade em interfaces Web. Essa métrica é chamada de índice de usabilidade (VASCONCELOS; BALDOCHI JR., 2011).

Analisando os resultados dos experimentos realizados com diferentes aplicações Web, é possível concluir que a abordagem proposta neste trabalho contribui de forma eficiente para a avaliação remota e automática de usabilidade dessas aplicações, tendo em vista

que os índices de usabilidade resultantes refletem os resultados dos testes de usabilidade realizado em laboratório.

Além disso, os experimentos mostram que os problemas de usabilidade identificados são válidos, e as recomendações propostas para a solução dos mesmos colaboram efetivamente no sentido de solucionar ou, ao menos, minimizar os problemas apontados (VASCONCELOS; BALDOCHI JR., 2012).

Pelos resultados apresentados, o USABILICS mostra-se um sistema inovador e com potencial para ser adotado para avaliação de usabilidade em grandes aplicações comerciais. Espera-se também que as contribuições produzidas pelo em seu desenvolvimento possam fomentar o avanço das pesquisas na área de avaliação remota e automática de aplicações Web.

7.1 Resultados obtidos

O desenvolvimento do USABILICS gerou contribuições relevantes para pesquisas relacionadas à avaliação remota e automática de usabilidade, destacando-se entre elas:

- O modelo de interface COP, que permite definir tarefas reais de uma aplicação Web de uma maneira mais intuitiva que abordagens anteriores. Acredita-se que o COP possa ser utilizado por modelos de tarefas para relacionar os elementos da interface aos conceitos das notações, facilitando a utilização de modelos de tarefas na avaliação da usabilidade de aplicações Web.
- O UsaTasker, que permite ao avaliador definir e gerenciar as ações das tarefas da aplicações Web.
- O índice de usabilidade que oferece uma forma simples e direta de mensurar a eficiência e a eficácia da usabilidade de interfaces Web a partir da análise de tarefas.
- A automatização da fase crítica de avaliação de interfaces, identificando possíveis problemas de usabilidade através da comparação das interações dos usuários com o caminho ótimo definido pelo avaliador nas tarefas.

Finalmente, cabe destacar os extensos experimentos realizados durante o trabalho no sentido de verificar a efetividade do índice de usabilidade, do sistema de recomendações automáticas e do meconismo de definição de tarefas. As partes essenciais do sistema foram

amplamente validadas através desses experimentos, o que reforça o valor das contribuições produzidas (VASCONCELOS; BALDOCHI JR., 2011, 2012).

7.2 Trabalhos futuros

No intuito de incentivar a pesquisa na área de avaliação remota e automática de usabilidade, e para contribuir com o desenvolvimento de trabalhos já existentes, são apresentadas as seguintes sugestões para trabalhos futuros:

1. Desenvolvimento de uma API para consulta das informações resultantes da avaliação de usabilidade, a fim de readaptar a interface durante a interação do usuário por ações pré-programadas;
2. Utilizar técnicas de mineração de dados para identificar padrões de comportamento e definir uma ontologia para recomendação baseada nos padrões identificados;
3. Implementar a verificação de heurísticas utilizando o modelo de interface COP;
4. Utilizar o modelo de interface COP para facilitar a implementação de ferramentas que utilizam modelos de tarefas baseados em notações;
5. Avaliar exaustivamente diversas aplicações Web a fim de definir padrões de comportamento de acordo com as áreas de negócios ou funcionalidades das aplicações Web.

Referências

ABNT (1996). NBR 13596 - tecnologia de informação - avaliação de produto de software - características de qualidade e diretrizes para o seu uso. Technical report, ABNT.

Albers, M. J. (1998). Goal-driven task analysis: improving situation awareness for complex problem-solving. In *Proceedings of the 16th annual international conference on Computer documentation*, SIGDOC '98, New York, NY, USA, pp. 234–242. ACM.

AMBOSS (2008). Amboss. `http://wwwcs.uni-paderborn.de/cs/ag-szwillus/lehre/ws05_06/PG/PGAMBOSS`.

Andreasen, M. S., H. V. Nielsen, S. O. Schrøder, and J. Stage (2007). What happened to remote usability testing?: an empirical study of three methods. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, San Jose, California, USA, pp. 1405–1414. ACM.

Atterer, R. (2006). Logging usage of ajax applications with the usaproxy HTTP proxy. In *Proceedings of the WWW 2006 Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection*, WWW '06, Edinburgh, Scotland. ACM.

Atterer, R. and A. Schmidt (2007). Tracking the interaction of users with ajax applications for usability testing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, San Jose, California, USA, pp. 1347–1350. ACM.

Badre, A. N. (2002). *Shaping Web Usability: Interaction Design in Context*. Addison-Wesley Professional.

Balbo, S. (1995). Automatic evaluation of user interface usability: Dream or reality? In *Proceedings of the Queensland Computer-Human Interaction Symposium*, QCHI '95, Queensland, Australia. ACM.

Balbo, S., S. Goschnick, D. Tong, and C. Paris (2005). Leading web usability evaluations to WAUTER. In *Proceedings of the Eleventh Australasian World Wide Web Conference*, AusWeb '05, Gold Coast, Australia. ACM.

Baresi, L., F. Garzotto, and P. Paolini (2000). From web sites to web applications: New issues for conceptual modeling. In *Proceedings of the World Wide Web and Conceptual Modeling: Conceptual Modeling for E-Business and the Web*, ER '00, London, UK, pp. 89–100. Springer-Verlag.

Bastien, C. and D. Scapin (1993). Rt-0156 - ergonomic criteria for the evaluation of human-computer interfaces. rapport technique de l'inria. `http://www.inria.fr/rrrt/rt-0156.html`.

Bevan, N., J. Kirakowski, and J. Maissel (1991). What is usability? `http://www.usability.serco.com/papers/whatis92.pdf`.

Bias, R. G. (1994). The pluralistic usability walkthrough: coordinated empathies. In J. Nielsen and R. L. Mack (Eds.), *Usability inspection methods*, Chapter The pluralistic usability walkthrough: coordinated empathies, pp. 63–76. New York, NY, USA: John Wiley & Sons, Inc.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer 21*(5), 61–72.

Bolchini, D. and J. Mylopoulos (2003). From task-oriented to goal-oriented web requirements analysis. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering*, WISE '03, Washington, DC, USA, pp. 166–175. IEEE Computer Society.

Builtwith, T. U. S. (2012). Top in content management system: most popular content management systems. `http://trends.builtwith.com/cms/top`.

Caffiau, S., D. Scapin, P. Girard, M. Baron, and F. Jambon (2010). Increasing the expressive power of task analysis: Systematic comparison and empirical assessment of tool-supported task models. *Interact. Comput. 22*(6), 569–593.

Crystal, A. and B. Ellington (2004). Task analysis and human-computer interaction: approaches, techniques, and levels of analysis. In *Proceedings of the Tenth Americas Conference on Information Systems*, New York, USA, pp. 391–399. Association for Information Systems.

Cybis, W. A. (2009). Estudo de validação de abordagem de monitoramento de usabilidade baseada na análise de logs orientada a transações. In *Proceeding of the 4th Latin American Conference on Human-Computer Interaction (CLIHC)*, CLIHC '09, Merida Yucatan, Mexico.

de Lourdes Oliveira Martins, M. (2004). O papel da usabilidade no ensino à distância mediado por computador. Master's thesis, Centro Federal de Educação Tecnológica de Minas Gerais.

Dias, C. (2001). Métodos de avaliação de usabilidade no contexto de portais corporativos: um estudo de caso no senado federal. Master's thesis, Universidade de Brasília.

Dix, A. J. (1995). Formal methods: An introduction to and overview of the use of formal methods within hci. In *Perspectives in HCI: Diverse Approaches*, Chapter 2 in Perspectives in HCI: Diverse Approaches, pp. 9–43. London, UK: Academic Press.

Fidas, C., C. Katsanos, E. Papachristos, T. Nikolaos, and N. Avouris (2007). Remote usability evaluation methods and tools: A survey. In *Panhellenic Conference on HumanComputer Interaction PCHCI 2007*, Number 2002 in 1, pp. 151–163. ACM.

Frisoni, B. C. and V. Steil (2005). Como estruturar melhor a área de contato com o usuário? A utilização da técnica de card sorting para desenvolver a estrutura do website do núcleo de inovação em design da cadeia têxtil. In *5º Congresso Internacional de Ergonomia, Usabilidade, Design de Interface e Interação Humano-Computador*, ERGODESIGN '05, Rio de Janeiro, Rio de Janeiro, USA.

Frokjaer, E., M. Hertzum, and K. Hornbaek (2000). Measuring usability: are effectiveness, efficiency, and satisfaction really correlated? In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '00, The Hague, The Netherlands, pp. 345–352. ACM.

Google (2011). Google analytics. `http://www.google.com/analytics`.

Hégaret, P. L., W. L., and J. Robie (2000). What is the document object model? `http://www.w3.org/TR/DOM-Level-2-Core/introduction.html`.

ISO/IEC (1996). 9126 Software engineering - Product quality. Technical report, ISO.

ISO/IEC (1998). 9241-11 Ergonomic requirements for office work with visual display terminals (VDT)s - Part 11 Guidance on usability. Technical report, ISO.

Ivory, M. Y. and M. A. Hearst (2001). The state of the art in automating usability evaluation of user interfaces. *ACM Comput. Surv. 33*, 470–516.

Jokela, T., J. Koivumaa, J. Pirkola, P. Salminen, and N. Kantola (2006). Methods for quantitative usability requirements: a case study on the development of the user interface of a mobile phone. *Personal Ubiquitous Comput. 10*(6), 345–355.

Jordan, P. W. (1998). *An Introduction To Usability*. 0748407626: CRC Press.

Kahn, M. J. and A. Prail (1994). Usability inspection methods. In J. Nielsen and R. L. Mack (Eds.), *Usability inspection methods*, Chapter Formal usability inspections, pp. 141–171. New York, NY, USA: John Wiley & Sons, Inc.

Lecerof, A. and F. Paternò (1998, October). Automatic support for usability evaluation. *IEEE Trans. Softw. Eng. 24*(10), 863–888.

Lewis, C., P. G. Polson, C. Wharton, and J. Rieman (1990). Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, CHI '90, New York, NY, USA, pp. 235–242. ACM.

Melguizo, M. C. P., H. van Oostendorp, and I. Juvina (2007). Predicting and solving web navigation problems. In *Proceedings of the eighteenth conference on Hypertext and hypermedia*, HT '07, pp. 47–48. ACM.

Miranda, F. and A. d. Moraes (2003). Avaliação da interface de um site de comércio eletrônico através da técnica avaliação cooperativa. In *2º Congresso Internacional de Ergonomia, Usabilidade, Design de Interface e Interação Humano-Computador*, ERGODESIGN 03, Rio de Janeiro, Rio de Janeiro, USA.

Moen, P. (2000). *Attribute, Event Sequence, and Event Type Similarity Notions for Data Mining*. Ph. D. thesis, University of Helsinki.

Nielsen, J. (1993). *Usability Engineering*. MA AP Professional Ed.

Nielsen, J. (1995). Usability inspection methods. In *Conference companion on Human factors in computing systems*, CHI '95, New York, NY, USA, pp. 377–378. ACM.

Nielsen, J. (2000a). *Projetando Websites*. Campus.

Nielsen, J. (2000b). Why you only need to test with 5 users. `http://www.useit.com/alertbox/`.

Nielsen, J. and H. Loranger (2007). *Usabilidade na Web: Projetando Websites com Qualidade*. Elsevier Brazil.

Nielsen, J. and R. Molich (1990). Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, CHI '90, New York, NY, USA, pp. 249–256. ACM.

Oh, S. J. and J. Y. Kim (2004). A hierarchical clustering algorithm for categorical sequence data. *Inf. Process. Lett. 91*, 135–140.

Paganelli, L. and F. Paternò (2002). Intelligent analysis of user interactions with web applications. In *Proceedings of the 7th international conference on Intelligent user interfaces*, IUI '02, pp. 111–118. ACM.

Paganelli, L. and F. Paternò (2003). Tools for remote usability evaluation of Web applications through browser logs and task models. *Behavior Research Methods, Instruments & Computers 35*(3), 369–378.

Paternò, F. (2000). *Model-Based Design and Evaluation of Interactive Applications*. Springer.

Paternò, F. (2002). Task models in interactive software systems. In *In Handbook of software engineering and knowledge*. World Scientific Publishing Co.

Paternò, F., C. Mancini, and S. Meniconi (1997). Concurtasktrees: A diagrammatic notation for specifying task models. In *Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, INTERACT 97, pp. 362–369. Chapman & Hall, Ltd.

Petrie, H., F. Hamilton, N. King, and P. Pavan (2006). Remote usability evaluations with disabled people. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, New York, NY, USA, pp. 1133–1141. ACM.

Pressman, R. S. (2009). *Software Engineering: A Practitioner's Approach*. McGraw-Hill.

Raggett, D., A. L. Hors, and I. Jacobs (1999). The global structure of an HTML document. `http://www.w3.org/TR/1999/REC-html401-19991224/struct/global.html`.

Rivolli, A., D. A. Marinho, and L. T. E. Pansanato (2008). Wautt: Uma ferramenta para o rastreamento da interação do usuário com aplicações interativas web. In *Companion Proceedings of the XIV Brazilian Symposium on Multimedia and the Web*, WebMedia '08, pp. 179–181. ACM.

Santana, V. F. and M. C. C. Baranauskas (2010). Summarizing observational client-side data to reveal web usage patterns. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pp. 1219–1223. ACM.

Shneiderman, B. and C. Plaisant (2004). *Designing the user interface: strategies for effective human-computer interaction*. Addison-Wesley Publishing Company.

Sommerville, I. (2010). *Software Engineering*. Addison Wesley.

Tarby, J.-C. and M.-F. Barthet (1996). The DIANE+ method. In *CADUI*, pp. 95–120. Presses Universitaires de Namur.

Tiedtke, T., C. Martin, and N. Gerth (2002). AWUSA: A tool for automated website usability analysis. In *Proceedings of the 9th International Workshop on the Design, Specification and Verification of Interactive Systems*, DSVIS '02, pp. 251–266. Springer.

Tullis, T., S. Fleischman, M. McNulty, C. Cianchette, and M. Bergel (2002). An empirical comparison of lab and remote usability testing of web sites. In *Usability Professionals Association Conference*, UPA '02. IEEE.

Van Der Veer, G. C., B. F. Lenting, and B. A. J. Bergevoet (1996). GTA: Groupware task analysis - modeling complexity. *Acta Psychologica 91*, 297–322.

Vargas, A., H. Weffers, and H. V. da Rocha (2010). A method for remote and semi-automatic usability evaluation of web-based applications through users behavior analysis. In *Proceedings of the 7th International Conference on Methods and Techniques in Behavioral Research*, MB '10, Eindhoven, Netherlands, pp. 19:1–19:5. ACM.

Vasconcelos, L. G. and L. A. Baldochi, Jr. (2011). Usabilics: avaliaćão remota de usabilidade e métricas baseadas na análise de tarefas. In *Proceedings of the 10th Brazilian Symposium on on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction*, IHC+CLIHC '11, Porto Alegre, Brazil, Brazil, pp. 303–312. Brazilian Computer Society.

Vasconcelos, L. G. and L. A. Baldochi, Jr. (2012). Towards an automatic evaluation of web applications. In *SAC '12: Proceedings of the 27th Annual ACM Symposium on Applied Computing*, New York, NY, USA, pp. 709–716. ACM.

Vermeeren, A. P., J. Attema, E. Akar, H. de Ridder, A. J. von Doorn, c. Erbug, A. E. Berkman, and M. C. Maguire (2008). Usability problem reports for comparative studies: Consistency and inspectability. *Human Computer Interaction 23*(4), 329–380.

W3C, H. W. G. (2002). Xhtml^{TM}1.0 the extensible hypertext markup language (second edition). `http://www.w3.org/TR/xhtml1/\#h-4.10`.