Instituto de Computação
Universidade Estadual de Campinas

Um Método para Avaliação Remota e Automatizada de Usabilidade de Aplicações Web

Ariel Vargas

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Ariel Vargas e aprovada pela Banca Examinadora.

Campinas, 10 de maio de 2012.

Profª. Drª. Heloísa Vieira da Rocha
(Orientadora)

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para obtenção do título de Doutor em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA POR
ANA REGINA MACHADO - CRB8/5467
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA - UNICAMP

Vargas, Ariel, 1981-
V426m Um método para avaliação remota e automatizada de usabilidade de aplicações Web / Ariel Vargas. – Campinas, SP : [s.n.], 2012.

Orientador: Heloísa Vieira da Rocha.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Interfaces de usuário (Sistema de computador). 2. Interação homem-máquina - Avaliação. 3. World Wide Web. 4. Internet. 5. Serviços de informação - Estudo de usuários. I. Rocha, Heloísa Vieira da, 1954-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em inglês: A method for remote and automatic usability evaluation of Web applications
Palavras-chave em inglês:
User interfaces (Computer systems)
Human-computer interaction - Evaluation
World Wide Web
Internet
Information services - Use studies
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Heloísa Vieira da Rocha [Orientador]
Anamaria Gomide
Hans Kurt Edmund Liesenberg
Eduardo Hideki Tanaka
Marcelo Morandini
Data de defesa: 10-05-2012
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Tese Defendida e Aprovada em 10 de maio de 2012, pela Banca examinadora composta pelos Professores Doutores:

Dr. Eduardo Hideki Tanaka
Instituto de Pesquisas Eldorado

Prof. Dr. Marcelo Morandini
EACH / USP

Profª. Drª. Anamaria Gomide
IC / UNICAMP

Prof. Dr. Hans Kurt Edmund Liesenberg
IC / UNICAMP

Profª. Drª. Heloisa Vieira da Rocha
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Um Método para Avaliação Remota e Automatizada de Usabilidade de Aplicações Web

Ariel Vargas[*]

Maio de 2012

Banca Examinadora:

- Profª. Drª. Heloísa Viera da Rocha (Orientadora)
Instituto de Computação – UNICAMP

- Profª. Drª. Anamaria Gomide
Instituto de Computação – UNICAMP

- Prof. Dr. Hans Kurt Edmund Liesenberg
Instituto de Computação – UNICAMP

- Prof. Dr. Eduardo Hideki Tanaka
Instituto de Pesquisas Eldorado

- Prof. Dr. Marcelo Morandini
Escola de Artes, Ciências e Humanidades (EACH) – USP

- Profª. Drª. Maria Beatriz Felgar de Toledo (Suplente)
Instituto de Computação – UNICAMP

- Prof. Dr. Celmar Guimarães da Silva (Suplente)
Faculdade de Tecnologia – UNICAMP

- Profª. Drª. Joice Lee Otsuka (Suplente)
Universidade Aberta do Brasil (UAB) - UFSCar

[*] Suporte Financeiro de: Capes (2007), CNPQ (2008 e 2009) e Erasmus Mundus (2010 e 2011).

Resumo

Observar as ações de usuários interagindo com a interface de uma aplicação computacional é a base de métodos tradicionais de avaliação de usabilidade, como os testes com usuários e a observação em campo. Apesar de populares e eficientes na descoberta de problemas, esses métodos tradicionais são considerados bastante caros, devido aos custos de envolver usuários, de preparar a infra-estrutura e especialmente da própria execução dos métodos e análise de seus resultados. A qualidade dos resultados da execução desses métodos depende da escolha dos usuários participantes da avaliação, que precisam ser representativos de todos os perfis de usuários da aplicação avaliada. Na Web, devido à diversidade e característica distribuída dos usuários, a obtenção de usuários representativos dos diversos perfis torna-se bastante difícil. Nos métodos tradicionais de avaliação, especialistas em usabilidade analisam a interação de cada usuário interagindo com a aplicação, para encontrar problemas de usabilidade. Esta análise depende do conhecimento e esforço desses especialistas para encontrar problemas na aplicação avaliada. Desse modo, os métodos tradicionais, se utilizados em grande escala de usuários, tornam-se demasiadamente caros devido a seus custos e tempo de execução. Outra característica importante das aplicações Web é sua constante evolução, o que necessita de avaliações de usabilidade periódicas e constantes, para verificar se alterações não prejudicam sua usabilidade. Da mesma forma, os métodos tradicionais de avaliação de usabilidade tornam-se demasiadamente caros, se utilizados em avaliações constantes da usabilidade de aplicações Web. Nesse contexto, esta tese buscou, na automatização do processo de avaliação de usabilidade, uma alternativa para diminuir as dificuldades mencionadas anteriormente em se avaliar a usabilidade de aplicações Web, especialmente em sua fase de manutenção. Assim, esta tese defende a hipótese de que,

utilizando-se um método de avaliação de usabilidade fundamentado na captura e análise da interação de usuários de maneira remota e automatizada, é possível diminuir as dificuldades existentes nos métodos tradicionais na fase de manutenção de uma aplicação Web, possibilitando ainda: a análise da interação de usuários em campo e em grande escala de usuários; a identificação de padrões de interação dos usuários com a aplicação; uma avaliação com baixo custo de execução, favorecendo a realização de avaliações periódicas de usabilidade de uma aplicação Web. Neste sentido, esta tese apresenta o desenvolvimento do WebHint, um método remoto e automatizado de avaliação de usabilidade para aplicações Web. O método proposto nesta tese foi aplicado em três estudos de usabilidade que validaram o WebHint como uma alternativa viável para a realização de avaliações remotas e automatizadas de usabilidade de aplicações Web.

Abstract

Observing users interacting with user interfaces is the basis of traditional usability evaluation methods as user tests and field observation. In spite of their popularity and efficiency in finding usability problems, these traditional methods are considered expensive, due to the costs of involving users in the evaluation, preparing the infrastructure and specially, due to the costs of executing the methods and analyzing the results. The quality of the results obtained with these methods depends on the users participating of the evaluation, who need to represent all the user's profiles of the evaluated application. In the Web, due to the diversity and distributed location of the users, it is difficult to obtain representative users. In traditional methods, usability experts analyze the behavior of each user interacting with the application in order to detect usability problems. This subjective analysis depends on the knowledge and work of the expert to find the problems. This way, the traditional methods became too expensive when it is necessary to involve high number of users in the evaluation. Another important aspect of Web applications is their frequent evolvement, which requires usability evaluations performed frequently to avoid usability problems caused by changes on the applications. The same way, the traditional methods are too expensive to be performed in a regular basis. In that context, this thesis proposes an automatic method for usability evaluation, as an alternative approach to deal with the difficulties mentioned above. This way, this thesis supports the hypothesis that is possible to decrease the difficulties regarding to the traditional methods through a usability evaluation method based on the automatic capture and analysis of user's interaction. Thus, this thesis, presents the development of the WebHint, a remote and automatic usability evaluation method for Web applications. The proposed method was applied in three usability studies, and their

results validated the WebHint as a feasible alternative to perform remote and automatic usability evaluations of Web applications.

Publicações

Vargas, A., Weffers, H., Rocha, H. V.. 2010.
A Method for Remote and Semi-Automatic Usability Evaluation of Web-based Applications Through Users Behavior Analysis. In *Proceedings of Measuring Behavior 2010*. Measuring Behavior 2010. Eindhoven, The Netherlands 2010 New York: ACM

Vargas, A., Weffers, H., Rocha, H. V.. 2010.
Usability Analysis of User Interaction in Online Applications Based on User Interaction Log Analysis. In *Proceedings of ICCEE 2010 ICCEE 2010 - 3rd International Conference on Computer and Electrical Engineering.* Chengdu, China.

Vargas, A., Weffers, H., Rocha, H. V.. 2011.
Analyzing User Interaction Logs to Evaluate the Usability of Web Applications. In *Proc. of 3rd Symposium on Web Society (SWS) 2011.* Port Elizabeth.

Silva, A. C., Letizio, C., Caceffo, R., Vargas, A., Rocha, H. V.. 2011.
Avaliação da Usabilidade e da Acessibilidade do Ambiente Virtual de Aprendizagem TelEduc: levando o 'aprender'a todos. In *Proc. of Brazilian Symposium on Human Factors in Computing Systems – IHC 2011.* Porto de Galinhas.

Vargas, A., Weffers, H., Rocha, H. V.. 2011.
Discovering and Analyzing Patterns of Usage to Detect Usability Problems in Web Applications. In *Proc. of ISDA 2011 - 11th International Conference on Intelligent Systems Design and Applications ISDA 2011.* Córdoba.

Eu dedico esta tese à minha mãe (in memoriam), pelo amor incondicional, carinho e dedicação que sempre me ofereceu ao longo de toda a minha vida. Que mesmo partindo tão repentinamente, poucos meses antes de eu concluir este trabalho, sempre se fez presente em meu coração com todo o seu amor, me dando a força que precisei para chegar ao final desta etapa tão importante de minha vida. Hoje, compartilho, com você mãe, a realização desse sonho, tendo a certeza de que, em algum lugar, você estará muito feliz por isso. Com todo o meu eterno amor, muito obrigado.

Agradecimentos

A Deus, que mesmo me apresentando um caminho difícil, com obstáculos que pareciam muitas vezes intransponíveis, me deu a coragem para enfrentar todos os desafios e me permitiu concluir mais esta etapa de minha vida, realizando um sonho meu e de tantas pessoas que estiveram comigo durante toda essa jornada. Apesar de difícil, esse caminho foi repleto de momentos felizes e inesquecíveis, de experiências únicas e incomparáveis, tornando-o parte de minha vida para sempre.

Ao meu pai, que sempre me apoiou em todas as decisões de minha vida, nunca medindo esforços para me auxiliar na busca por meus sonhos, ideais, planos e projetos, e principalmente, por todo o amor e carinho que sempre me ofereceu em todos os dias de minha vida.

À tia Luiza, por ter sempre me oferecido o amor e carinho de uma segunda mãe.

À tia Izolete, à tia Marlise, à Vânia e à tia Odete por serem pessoas tão especiais e para as quais terei eterna gratidão.

À Heloísa, minha orientadora e amiga, por ter acreditado em meu potencial e sempre ter me dado a liberdade e o apoio necessários para que eu pudesse crescer sempre como profissional e pessoa.

Ao Harold, meu orientador e grande amigo, que fiz durante o doutorado sanduíche na Holanda, pela generosidade em ter me recebido como parte de sua equipe por quase 2 anos, por ter contribuído tanto e de maneira tão importante com meu trabalho e principalmente, por ter sempre me oferecido sua amizade, compreensão e apoio durante todo o tempo em que trabalhamos juntos.

A toda à equipe do *Laboratory for Quality Software* e funcionários da TU/e, pela amizade e pelo ambiente de trabalho tão agradável que me proporcionaram enquanto estive lá: Joost, Reinier, Ion, Martin, Martijn, Sergey, Willeke, Anneroos, Lutgart e Nelleke.

A toda a equipe do programa EBWII (Euro Brazilian Windows II), em especial à Ana Reis, que me forneceu todo o suporte necessário, sempre de maneira prestativa e me auxiliando em todas às necessidades durante minha participação no programa de doutorado sanduíche.

Aos amigos que fiz durante o doutorado no Instituto de Computação da UNICAMP, com os quais compartilhei intermináveis horas de trabalho e estudo, mas que graças à sua companhia, tornou-se também um período muito feliz: Joana, Jorge, Carlos, Diego, André, Ricardo, Eduardo, Sheila, Nádia, Luiz e Elaine.

Aos amigos da república "J5 e Agregados", com os quais compartilhei tantos momentos felizes em meu dia-a-dia em Campinas: Vini, Camys, Diego, Marco, Jeff, Dani, Naza, Gabi, Teta, Laura, Vivi.

Aos amigos sempre presentes, independente da distancia geográfica: Daniel, Maria Angélica, Marta, Thiago, Gaspar, Dri e Fê.

Aos eternos amigos de infância: Gil, Bruno, Leandro, Lucas, Prim, Dai e Karluza.

Aos amigos que fiz em Eindhoven, que se tornaram rapidamente pessoas tão importantes e queridas, com as quais compartilhei tantos momentos inesquecíveis: Daniel, Davide, Valentino, Danielle, Pedro, Johnny, Zaida, Olga, Jorge, Mark, Dominic, Alex, Fanis, Diana, Ivana, Bogdan, Martin, Tudor, Zlatka, Jonathan, Eirini, Francisco, Rubina, Fons, Vivian, Marcos, Zé Leonardo, Carlota, Thiago e Paulo.

Aos professores e funcionários do Instituto de Computação da UNICAMP, com os quais convivi diariamente durante vários anos e que contribuíram para que este trabalho fosse realizado.

À Capes, ao CNPq e ao Programa Erasmus Mundus, pelo apoio financeiro a este trabalho.

A todos os meus amigos, cada um por sua importância tão grande e particular em minha vida, que hoje, espalhados por todo o mundo, me fazem ter a certeza de serem minha maior conquista e o maior presente que a vida me deu.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

"As interfaces de usuário de software vêm se tornando cada vez mais importantes com o aumento do número de usuários e de software disponíveis" (Nielsen, 1993). Esta observação demonstra que a importância da usabilidade das interfaces de software já vem crescendo há cerca de 20 anos. E sem qualquer dúvida, nos dias de hoje, a importância de uma interface com boa usabilidade em um software pode ser considerada elemento crucial para seu sucesso, aceitação e uso (Sherman, 2006; Lynch & Gilmore, 2006).

Com a popularização dos computadores e da Web, tem-se hoje um número muito maior de usuários e aplicações do que há cerca de 20 anos atrás. O grande número de usuários, aliado à concorrência existente no mercado de software torna a usabilidade ainda mais importante nos dias de hoje, pois, conforme Nielsen já afirmava, "os usuários tendem cada vez mais a tornar-se exigentes e menos tolerantes a enfrentar dificuldades em utilizar um software" (Nielsen, 1993).

Na Web, a usabilidade ganha ainda mais importância, pois os usuários podem facilmente trocar de aplicação se não estiverem satisfeitos (Nielsen 2000). Palmer (2002), Roy et. al (2001) e Casaló et al. (2007) ressaltam a importância da usabilidade em aplicações Web, pois esta tem um papel determinante no ganho e na sustentação da confiança que os usuários têm na aplicação, seja esta uma aplicação de *e-banking*, uma loja virtual, um *site* de relacionamentos, ou qualquer outra.

Calisir et al. (2010), em uma pesquisa realizada com o objetivo de analisar qual a importância de diversos fatores em uma aplicação Web, apontam a usabilidade como o fator mais importante, na opinião dos usuários.

No entanto, uma interface com boa usabilidade não é construída simplesmente porque é desejável, apesar de sua importância. Para garantir o desenvolvimento de uma interface de software com boa usabilidade é necessário levar em conta os aspectos de usabilidade desde o princípio do desenvolvimento e por todo o ciclo de vida do software (Nielsen, 1993).

Apesar do ciclo de vida de um software não terminar com o seu lançamento, frequentemente as companhias de desenvolvimento de software focam mais seus esforços em usabilidade nas fases anteriores ao lançamento do produto, conforme relatam Nichols et al. (2003) e Chilana et al. (2011).

No entanto, mesmo com um bom projeto de design e desenvolvimento, detectando e corrigindo problemas de usabilidade antes do lançamento de um software é comum que diversos problemas sejam apenas descobertos quando os usuários começam efetivamente a utilizá-lo em seu dia a dia (Nichols et al., 2003; Chilana et al., 2011).

Nessa fase, quando os usuários começam efetivamente a utilizar o software, tem-se a possibilidade de analisar como realmente como eles interagem com a aplicação. Assim, é possível detectar as reais necessidades e dificuldades dos usuários, seus padrões de interação e consequentemente, detectar problemas de usabilidade existentes no software.

Analisar como os usuários utilizam uma aplicação é a base de diversos métodos de avaliação de usabilidade. Dentre eles, os mais conhecidos e utilizados são teste com usuários e observação em campo.

No teste com usuários, alguns usuários são selecionados para utilizar a aplicação enquanto são observados por um ou mais avaliadores. Cada usuário participante realiza um conjunto de tarefas pré-definidas, enquanto os avaliadores registram informações sobre seu comportamento, visando identificar os problemas de usabilidade existentes na aplicação (Nielsen, 1993).

Apesar de eficiente na descoberta de problemas, o teste com usuários é um método bastante caro devido aos custos de selecionar os usuários adequados ao teste, preparar o ambiente de utilização do software, preparar a infraestrutura para realização do teste, transportar os usuários até o local de sua realização, etc. Além disso, há também o tempo de execução de um teste e os custos de coleta e interpretação dos dados, que geralmente são altos (López et al., 2007; Baker et al., 2008; Hong et al., 2001).

Em aplicações Web, a realização de testes com usuários torna-se ainda mais difícil pela diversidade e localização distribuída de seus usuários.

Na observação em campo, o avaliador vai até o ambiente de trabalho do usuário e observa-o enquanto este utiliza o software para realizar suas tarefas cotidianas. Por analisar usuários reais em campo, este método é bastante caro, pois requer a observação dos usuários por longos períodos de tempo. Alternativamente, a observação pode ser feita através da gravação de vídeos, onde se monitora usuários por um determinado período de tempo (Nielsen, 1993). Mesmo realizada através da gravação de vídeos, o que pode evitar

o deslocamento de um avaliador para o ambiente de trabalho do usuário, a análise posterior dos vídeos demanda bastante tempo e esforço (no mínimo o mesmo número de horas de vídeo gravadas).

Testes com usuários podem também ser realizados de maneira remota, através da uso de vídeo conferência, por exemplo, eliminando-se a necessidade de se deslocar usuários para um laboratório de testes. Mesmo assim, o tempo de execução de um teste e a análise dos seus resultados permanecem altos (López et al., 2007; Baker et al., 2008; Hong et al., 2001; Dix et al.,2004).

Devido aos custos, tempo e esforço necessários para se realizar uma avaliação de usabilidade utilizando-se os métodos de testes com usuários e observação em campo, é bastante comum que esses métodos sejam executados com um pequeno conjunto de usuários (Thompson et al., 2004; Hong et al., 2001).

Analisar poucos usuários pode refletir apenas a visão de um determinado grupo, assim, a avaliação acaba tendo um caráter mais qualitativo e menos quantitativo, o que pode deixar de revelar problemas que só seriam percebidos em uma análise mais abrangente, com um número maior de usuários (Hong et al., 2001; Obendorf et al., 2004).

Além disso, para avaliar a severidade de problemas de usabilidade é importante analisar o número de usuários e a frequência com a qual estes são afetados pelos problemas. Por exemplo, dado um certo conjunto de problemas de usabilidade detectados em uma avaliação de usabilidade, é importante definir a prioridade com a qual estes precisam ser corrigidos. Saber o número de usuários afetados por um problema e a frequência com a qual esses usuários são afetados é importante na definição da prioridade com a qual o

problema precisa ser corrigido. Assim, problemas mais prioritários podem ser corrigidos antes de problemas com prioridade menor.

Além da diversidade e localização distribuída de seus usuários, as aplicações Web possuem ainda outra importante característica que influencia em sua usabilidade, sua constante evolução (Oreilly, 2007). Por estar *online*, qualquer alteração ou atualização de uma aplicação torna-se imediatamente disponível a todos os seus usuários. Assim, sua evolução é constante, pois não há tempo gasto na distribuição ou instalação de uma nova versão da aplicação, como acontece em aplicações *desktop*. Na Web, os usuários têm acesso à última versão disponível, a cada momento que acessam a aplicação.

Essa característica torna a avaliação de usabilidade uma necessidade constante em aplicações Web, pois alterações podem incluir problemas de usabilidade que dificultam a utilização das aplicações pelos usuários. Segundo Chilana et al. (2011), avaliar a usabilidade de aplicações periodicamente tende a ser uma necessidade crescente e cada vez mais importante para a evolução das aplicações.

Levando em conta que os usuários tendem a se tornar cada vez mais exigentes em relação à usabilidade das aplicações, as empresas têm se preocupado cada vez mais em manter suas aplicações sempre evoluindo em função das necessidades dos usuários. Assim, é cada vez mais importante obter informações sobre a interação dos usuários com a interface das aplicações em campo, ou seja, analisando usuários reais realizando suas tarefas cotidianas (Chilana et al., 2011).

Obter informações sobre usuários interagindo com aplicações Web, ou seja monitorar seu acesso de maneira remota e automática, já é uma prática bastante comum em comércio

eletrônico, *social media*, *e-banking*, ferramentas de email, serviços de busca, entre outros. Em comércio eletrônico, por exemplo, as aplicações geralmente rastreiam as ações de seus usuários visando compor o perfil individual de cada um. As companhias costumam obter informações sobre quais sessões determinado usuário visita com frequência em uma loja virtual, quais produtos busca, que tipo de produtos costuma comprar, etc. Com essas informações as companhias procuram direcionar campanhas de marketing mais específicas para cada usuário, visando ter mais sucesso nas vendas.

Obter informações sobre a interação de usuários com aplicações Web pode também ser uma alternativa para se analisar a usabilidade dessas aplicações. Monitorando automaticamente as ações dos usuários, é possível obter informações sobre a maneira com a qual estes usuários interagem com as aplicações e através dessas informações detectar possíveis problemas usabilidade.

Monitorar a interação de usuários reais interagindo com aplicações Web de maneira automática permite que se obtenha dados de usuários reais, executando suas tarefas reais na interação com as aplicações.

Além disso, se a captura da interação dos usuários for feita também de maneira remota é possível capturar dados de todos os usuários acessando a aplicação. Por fim, a captura remota e automática permite que a obtenção desses dados seja feita de maneira contínua e com baixo custo, favorecendo a obtenção de dados dos usuários a qualquer momento que uma alteração seja realizada na aplicação, evitando-se a necessidade de deslocar usuários para um laboratório, para a realização de um teste, por exemplo. Desse modo, essas informações podem ser analisadas periodicamente, em qualquer momento que se deseje avaliar a usabilidade da aplicação.

Para tornar viável a avaliação da usabilidade de aplicações Web com base nos dados obtidos através da captura remota e automatizada da interação dos usuários é necessário automatizar também o processo de análise desses dados. A captura remota e automatizada dos dados da interação gera uma imensa quantidade de dados, tornando a análise inviável se realizada sem o auxílio de técnicas automatizadas. Desse modo, é necessário tornar a análise desses dados menos onerosa em termos de tempo e custo.

Automatizando também o processo de análise desses dados, torna-se possível a análise de grandes quantidades de usuário, obtendo-se uma visão abrangente da utilização da aplicação. Desse modo, é possível a obtenção de dados quantitativos sobre, por exemplo, a quantidade de usuários afetados por um determinado problema de usabilidade, favorecendo a definição da severidade do problema e sua prioridade em ser corrigido.

Além disso, automatizando o processo de análise torna-se viável a execução de avaliações sucessivas, analisando-se a usabilidade da aplicação de maneira constante, ao longo de sua evolução.

Nesse contexto, a principal hipótese assumida nesta tese é a de que:

Utilizando-se um método de avaliação de usabilidade fundamentado na captura e análise da interação de usuários de maneira remota e automatizada, é possível diminuir as dificuldades existentes nos métodos tradicionais na fase de manutenção de uma aplicação Web, possibilitando ainda: a análise da interação de usuários em campo e em grande escala de usuários; a identificação de padrões de interação dos usuários com a aplicação; uma avaliação com baixo custo de execução, favorecendo a realização de avaliações periódicas de usabilidade de uma aplicação Web.

A partir desta hipótese, algumas questões de pesquisa foram levantadas e nortearam o desenvolvimento desta tese. As questões de pesquisa levantadas são:

- *Como realizar a avaliação de usabilidade de uma aplicação Web, capturando-se a interação de seus usuários em campo, de maneira remota e automatizada, sem interferir na interação dos usuários, possibilitando a obtenção de dados de grandes quantidades de usuários, ou até mesmo de todos os usuários acessando a aplicação avaliada?*

- *Como analisar, de maneira automatizada, os dados obtidos na captura automática da interação dos usuários, de modo a lidar com grandes volumes de informação, sem multiplicar o custo e tempo da análise, para cada usuário envolvido na avaliação?*

- *Como analisar a interação dos usuários de modo a obter informações sobre o comportamento predominante e tendências de uso da aplicação, levando em conta os usuários como um todo?*

- *Como desenvolver a avaliação de modo que esta possa ser repetida quantas vezes for necessário sem possuir um alto custo em sua execução, favorecendo análises periódicas da usabilidade da aplicação?*

Visando responder essas questões esta tese descreve o desenvolvimento do WebHint, um método proposto para avaliação de usabilidade de aplicações Web, fundamentado na captura e análise da interação de usuários de forma remota e automatizada.

O texto desta tese está organizado da seguinte maneira: o Capítulo 2 apresenta o Referencial Teórico e o Estado da Arte. Esse capítulo descreve a revisão da literatura estudada, abordando os assuntos que serviram como base teórica para o desenvolvimento desta tese e fundamentando sua originalidade. O Capítulo 3 apresenta o WebHint, o método para avaliação de usabilidade de aplicações Web proposto nesta tese. O Capítulo 4 é composto pelos artigos publicados, que descrevem os estudos de usabilidade desenvolvidos com o WebHint, para validação do método. O Capítulo 5 apresenta as conclusões deste trabalho e as possibilidades de trabalhos futuros impulsionadas pela pesquisa realizada e apresentada nesta tese.

Capítulo 2

Referencial Teórico e Estado da Arte

Neste capítulo, descreve-se os temas da literatura que serviram como base teórica para o desenvolvimento desta tese e para a composição do método proposto, o WebHint. Na apresentação deste capítulo, segue-se uma ordem cronológica dos temas que foram estudados ao longo do desenvolvimento da pesquisa que originou esta tese. O encadeamento cronológico é também o encadeamento lógico de como os assuntos foram estudados. Cada assunto estudado serviu de base para os assuntos que foram estudados em seguida, levando em conta as necessidades que foram sendo descobertas na composição do método proposto.

Primeiramente, apresenta-se uma visão geral sobre os métodos de avaliação usabilidade existentes. Este estudo foi realizado visando-se identificar as características de cada método e sua adequação em uma análise de usabilidade levando em conta a interação dos usuários no contexto de uso da aplicação e a possibilidade de se realizar a avaliação em grande escala de usuários. Os estudo desses métodos também visou a identificação das possibilidades de automatização do processo de avaliação em cada método.

Após o estudo dos métodos de avaliação de usabilidade, estudou-se as abordagens existentes para se realizar a captura e análise da interação de usuários em aplicações Web de forma remota e automatizada. Este estudo também trouxe a tona o estado da arte no

desenvolvimento deste tipo de análise e fundamenta a originalidade do desenvolvimento do WebHint.

No estudo das abordagens existentes para se realizar a captura da interação dos usuários, visou-se identificar a abordagem que mais se adequasse aos objetivos vislumbrados: realizar a captura de modo remoto e automático, não interferindo na interação dos usuários e também não necessitando da alteração da aplicação analisada.

O estudo das abordagens existentes para análise de dados obtidos na interação dos usuários com aplicações Web teve o objetivo de identificar e eleger um conjunto de técnicas que permitissem a manipulação e análise de grandes quantidades de dados de maneira automatizada, tornado a análise menos trabalhosa e diminuindo o custo e o tempo da avaliação. Além disso, teve-se o objetivo de favorecer uma visão geral da interação dos usuários, evidenciando-se as tendências e padrões de uso da aplicação.

2.1 Métodos de Avaliação de Usabilidade

Avaliação de usabilidade pode ser definida como o ato de mensurar (ou identificar potenciais problemas que afetam) a usabilidade de uma aplicação ou dispositivo, em relação a determinados usuários, executando determinadas tarefas em um determinado contexto (Hilbert & Redmiles, 2000).

Existem diversos métodos para se avaliar usabilidade, que, adaptando a classificação proposta por Ivory & Hearst (2001), podem ser divididos basicamente em 4 grupos, levando em conta a forma como a avaliação é executada: Inspeção, Pesquisa, Análise baseada em Modelos e Simulação e Teste.

2.1.1 Métodos de Inspeção

Os métodos de inspeção são baseados no conhecimento de especialistas em usabilidade, que inspecionam a interface de um software em busca de problemas de usabilidade. Os métodos mais conhecidos e utilizados nessa categoria são a Avaliação Heurística, o Percurso Cognitivo, o Percurso Pluralista e a Inspeção por Funcionalidades (Nielsen, 1994).

As principais vantagens dos métodos de inspeção são o seu baixo custo de execução, pois não necessitam da participação de usuários, e a possibilidade de serem executados em qualquer estágio do desenvolvimento de uma aplicação, especialmente nas fases iniciais, pois podem ser executados utilizando-se protótipos de baixa fidelidade (Nielsen, 1994; Dix et al., 2004).

No entanto, por não utilizarem usuários, os métodos de inspeção não permitem analisar o real uso da aplicação, ou seja, eles baseiam-se na previsão de como os usuários executariam certas tarefas na interface e na habilidade dos avaliadores em detectar possíveis problemas.

2.1.2 Métodos Baseados em Modelos e Simulação

Os métodos baseados em modelos e simulação permitem estimar a performance do usuário na utilização de uma aplicação, detalhando-se cada ação do usuário e estimando-se o tempo necessário para sua realização. Um dos métodos mais conhecidos nessa categoria é o GOMS (*Goals, Operators, Methods, Selection rules*) (Paternò, 2000).

No método GOMS, descreve-se, de uma forma hierárquica, como se executa uma tarefa na aplicação avaliada, ou seja, para um determinado objetivo, qual é a sequência de ações que devem ser executadas. O GOMS é utilizado para avaliar a usabilidade do procedimento realizado na execução de uma tarefa no software, mas não considera erros na execução das tarefas e trabalha apenas com tarefas sequenciais. Além do GOMS tradicional, existem algumas versões mais recentes do modelo como: KLM (*Keystroke-Level Model*), NGOMSL (*Natural GOMS Language*), CPMGOMS (*Critical Path Method GOMS*), que visam lidar com aspectos não abordados no GOMS, como a característica de só trabalhar com tarefas sequenciais e sem erros, por exemplo (Paternò, 2000).

Utilizando-se modelos de execução de tarefas e técnicas como algoritmos genéticos, é possível simular automaticamente o comportamento de usuários interagindo com uma aplicação. A utilização de métodos de simulação permite ao avaliador testar diversos cenários de uso, sem a participação de usuários reais. Desse modo, mesmo que de uma maneira artificial, é possível se obter dados que auxiliam, por exemplo, a analisar diferentes opções de design de uma interface e obter uma projeção de como seria o comportamento dos usuários nesses diferentes cenários (Ivory & Hearst, 2001).

Da mesma forma que nos métodos de inspeção, a maior vantagem dos métodos baseados em modelos e simulação é seu baixo custo, por não envolver usuários na execução do método. No entanto, por não utilizarem usuários, não levam em conta a real utilização da aplicação em sua análise e da mesma forma que os métodos de inspeção, não permitem avaliar a usabilidade da aplicação em seu contexto de uso.

2.1.4 Métodos de Investigação

Os métodos de investigação são compostos por técnicas que visam obter informações sobre os usuários, seu comportamento, suas necessidades e objetivos em relação à realização de suas tarefas em um software (Nielsen, 1993). Nesta categoria, o método de avaliação mais comum é a observação em campo.

Neste método, o avaliador vai até o ambiente de uso do software pelo usuário e observa-o por um determinado período de tempo, visando identificar como o usuário executa suas tarefas e o modelo mental que o usuário tem sobre a aplicação. Esse método geralmente é aplicado após o lançamento do produto, ou seja, quando este já está sendo utilizado no dia-a-dia dos usuários (Nielsen, 1993).

Além da observação em campo, outros métodos compõem esta categoria como: *Feedback* do Usuário, que consiste no relato voluntário de problemas encontrados pelos usuários quando estes utilizam uma aplicação; e os métodos de pesquisa como, Grupos Focais, Entrevistas, Estudos de Campo e Questionários, que visam a prospecção de informações a respeito do perfil dos usuários e requisitos da aplicação (Nielsen, 1993).

A maior vantagem dos métodos observação em campo e *feedback* do usuário é obtenção de informações sobre usuários reais da aplicação nas fases posteriores ao lançamento do produto.

A desvantagem do método de *feedback* do usuário, apesar de obter informações de usuários reais no contexto de uso da aplicação, é que sua execução depende do empenho e habilidade do usuário em reportar problemas. Desse modo, não é possível automatizar a

captura ou o processo de análise dos dados, pois ambas as atividades dependem de uma análise subjetiva, tanto para reportar um problema, quanto para analisar os relatos dos usuários.

O método de observação em campo, permite a observação de usuários reais, no contexto real de uso da aplicação, no entanto, é um método caro, pois demanda bastante tempo de execução e necessita de um especialista para observar a interação dos usuários com a aplicação. Além disso, a observação dos usuários por um especialista pode se tornar invasiva e influenciar no comportamento dos usuários, que por estarem sendo observados podem não apresentar seu comportamento habitual na interação com a aplicação (Nielsen, 1993).

Alternativamente, a observação dos usuários pode ser feita através da gravação de vídeos, onde se monitora o usuário por um determinado período de tempo. Dessa maneira, evita-se o deslocamento de um especialista para o ambiente de trabalho do usuário tornando a observação menos invasiva. No entanto, conforme já mencionado, a análise posterior dos vídeos demanda bastante tempo e esforço de um especialista, possuindo ainda os custos de infraestrutura para realizar a gravação de vídeos (Nielsen 1993). Desse modo, uma análise que leva em conta uma grande quantidade de usuários, pode se tornar inviável com a utilização desse método.

Mesmo que os custos da captura da interação dos usuários em vídeos fossem desconsiderados, a principal dificuldade do método de observação em campo é o processo de análise dos dados, que depende da análise de um especialista para detectar problemas de usabilidade. Desse modo, a análise dos vídeos ou da interação dos usuários na

observação em campo não pode ser automatizada, o que inviabiliza a utilização do método em larga escala de usuários.

2.1.5 Método de Teste com Usuários

Na realização de um teste com usuários são utilizados potenciais usuários que executam um conjunto pré-definido de tarefas na interface do software avaliado, enquanto um grupo de avaliadores observa a interação. Os avaliadores buscam identificar as dificuldades que cada usuário apresenta na realização das tarefas na aplicação e que podem revelar problemas na interface (Nielsen, 1993).

A principal vantagem do método de teste é a possibilidade de analisar a interação dos usuários executando tarefas na aplicação, facilitando a identificação das dificuldades que estes usuários podem encontrar e consequentemente revelar problemas de usabilidade na interface.

Apesar de sua eficiência, o teste com usuários, conforme já mencionado, é um método de avaliação de usabilidade bastante caro, devido aos custos de selecionar os usuários, preparar a infraestrutura, além do tempo de execução e análise dos dados (López et al., 2007; Baker et al., 2008; Hong et al., 2001).

Os resultados de testes com usuários são estritamente relacionados à quão representativos, dos diferentes perfis de usuários da aplicação, são os usuários utilizados nos testes. Abranger todos os perfis de usuários torna-se ainda mais difícil na Web, dada a diversidade e distribuição de usuários.

Além disso, o contexto real de uso do software é difícil de reproduzir em um laboratório de testes, pois diversos fatores relacionados ao contexto de uso podem influenciar no modo com o qual os usuários interagem com a aplicação em seu dia a dia, como a realização de tarefas paralelas, interrupções, etc. Os usuários podem também agir de forma diferente do habitual por estarem sendo observados por um avaliador, dificultando a obtenção de dados reais nos testes (Castillo et al., 2007).

Além das dificuldades mencionadas, que tornam a execução de testes com usuários cara, trabalhosa e demandando bastante tempo, a execução de testes com usuários baseia sua execução no trabalho de especialistas. A coleta de dados durante os testes é realizada através da observação dos usuários por especialistas. Esta observação, por ser feita de maneira subjetiva, não é passível de automatização, pois depende do conhecimento do observador para registrar as informações relevantes sobre o comportamento dos usuários. Além disso, a análise do comportamento dos usuários depende da experiência e conhecimento do especialista para detectar problemas de usabilidade existentes na aplicação.

Desse modo, as dificuldades em se automatizar as atividades dos especialistas nos testes com usuários dificulta a utilização desta técnica quando se deseja analisar a interação de usuários reais, realizando suas tarefas cotidianas em campo. Além disso, a utilização de testes com usuários torna-se demasiadamente cara, em termos de tempo e custo de execução, quando se deseja realizar uma avaliação incluindo um grande número de usuários, visando uma visão mais abrangente da utilização da aplicação.

2.1.6 Considerações sobre os métodos de avaliação estudados

De modo geral, os métodos tradicionais de avaliação de usabilidade apresentados anteriormente possuem algumas limitações, quando se deseja avaliar a usabilidade de uma aplicação Web baseando-se na análise em larga escala da interação de usuários.

Os métodos de inspeção, simulação e os métodos baseados em modelos não analisam dados da interação de usuários, desse modo, não permitem obter informações sobre o real uso da aplicação.

Os métodos de investigação como grupos focais, entrevistas, questionários e estudo de campo, são voltados para a obtenção de requisitos. O método de *feedback* do usuário obtém dados reportados pelos usuários, também não permitindo uma análise de como o usuário realiza suas tarefas na aplicação.

O método de observação em campo, apesar de possibilitar a análise da interação de usuários reais em campo, torna-se demasiadamente caro, em termos de tempo e custo de execução, em uma análise em grande escala de usuários, pois depende da observação destes por especialistas, que não pode ser automatizada.

Os testes com usuários, podem ser executados remotamente utilizando-se recursos de vídeo e áudio conferência, de maneira que os usuários não necessitam se deslocar para um laboratório de testes. No entanto, da mesma forma que a observação em campo os testes com usuários tornam-se demasiadamente caros se executados com vários usuários.

Desse modo, para possibilitar a avaliação de usabilidade, levando em conta a interação de usuários reais em campo e em larga escala de usuários, é necessário viabilizar a automatização do processo de avaliação.

Automatizar o processo de avaliação requer que as duas fases de uma avaliação, a captura e a análise dos dados, sejam automatizadas. Desse modo, apresenta-se a seguir o estudo das abordagens existentes para captura e análise automatizadas da interação de usuários em aplicações Web.

2.2 Captura da Interação dos Usuários

Para capturar, automaticamente e remotamente, a interação dos usuários em aplicações Web utiliza-se técnicas que permitem registrar as ações dos usuários na interface das aplicações e armazená-las em logs (Ivory & Hearst, 2001). Enquanto os usuários utilizam a aplicação, suas ações podem ser capturadas de modo transparente, sem interferir na interação. Assim, obtém-se dados de usuários reais executando suas tarefas na aplicação, em seu contexto real de uso. Além disso, automatizando-se a captura é possível a obtenção de dados da interação de grandes quantidades de usuários, ou até mesmo de todos os usuários acessando a aplicação (West & Lehman, 2006; Balbo et al., 2008; Ivory & Hearst, 2001).

A captura automática da interação dos usuários em aplicações Web pode ser realizada basicamente de duas formas: analisando-se logs de servidores Web ou capturando-se os eventos gerados no *browser* do usuário.

As abordagens para coleta de dados em logs de servidores Web permitem obter a sequência de páginas requisitadas ao servidor Web durante utilização da aplicação pelos usuários. Essas abordagens não interferem na interação dos usuários, pois o log é gerado automaticamente pelo servidor Web da aplicação (Hong et al., 2001). Dentre as abordagens que capturam a interação através de logs de servidores Web, encontram-se, por exemplo, o AWUSA (Tiedtke et. al 2002) e o Ergomonitor (Schwerz et al. 2007).

O AWUSA (*Automated Website Usability Analyzer*) consiste em uma abordagem para se analisar a usabilidade de Web sites, tendo o objetivo de verificar tanto sua estrutura estática, quanto analisar o comportamento do usuário (Tiedtke et. al 2002). As ações do usuário são extraídas automaticamente de arquivos de log gerados pelo servidor Web (Buchholz et al., 2007).

O Ergomonitor, por sua vez, também busca analisar a usabilidade de aplicações com base no log de servidores Web. A ferramenta busca a extração de métricas sobre tarefas executadas pelos usuários, a partir das páginas acessadas por estes e registradas no log do servidor Web da aplicação (Schwerz e at. 2007).

Diversas outras abordagens visam obter informações sobre a interação de usuários utilizando informações de servidores Web, como os trabalhos de Wu et al, (1998), Spiliopoulou & Faulstich (1998), Buchner & Mulvenna (1998), Mobasher et al. (2000), Spiliopoulou (2000), Chi (2002), Shahabi & Banaei-Kashani (2003), Chi et al. (2002), Dai & Mobasher (2005), Mobasher (2007), Domenech & Lorenzo (2007), Pascual & Durstler (2007).

No entanto, logs de servidores Web apenas registram as requisições de páginas da aplicação ao servidor, ou seja, basicamente a sequência de páginas acessadas pelos usuários, sem nenhuma informação sobre sua interação dentro de uma determinada página, por exemplo. Além disso, mesmo essas informações não são precisas, pois se entre o servidor Web e o usuário existir uma *cache,* contendo a página requisitada, a requisição da página é interceptada e não é registrada no log do servidor. O mesmo acontece em aplicações que utilizam tecnologias dinâmicas como AJAX (Garret, 2005), que também não precisam fazer requisições ao servidor a cada interação do usuário.

Outro problema na análise de logs de servidores Web é o compartilhamento de um mesmo endereço IP por diversos usuários em uma rede local, o que dificulta a identificação dos registros individuais de cada um (Hong et al., 2001).

Como alternativa aos logs de servidores há a possibilidade de se capturar a interação no *browser* dos usuários. A vantagem de se capturar a interação desse modo é a possibilidade de se registrar todas as ações dos usuários, desde cada movimento de mouse, cada caractere digitado, cada elemento da interface acessado, até as páginas Web requisitadas ao servidor, gerando informações bastante precisas sobre a interação dos usuários com a aplicação.

Essa alternativa é possível pois, como as aplicações Web são executadas dentro de um *browser,* este serve como plataforma padronizada para a execução das aplicações. Assim, as ações dos usuários dentro de uma aplicação geram eventos no *browser*, que podem ser registrados para monitorar a interação. Por exemplo, é possível registrar se determinado usuário pressionou um botão na interface da aplicação, digitou alguma palavra em um campo de texto, ou qualquer outro tipo de interação.

Outra característica importante que permite a captura das ações dos usuários de maneira automatizada são as linguagens de marcação como HTML, XHTML e XML (W3C, 2011). Quando uma página HTML é executada no *browser* do usuário, sua estrutura é representada como um conjunto hierárquico de objetos organizados em uma árvore conhecida como árvore DOM (*Document Objetct Model*), que constitui-se em uma convenção definida pela W3C para a manipulação de documentos HTML ou XML (W3C, 2011). A Figura 1, ilustra uma árvore DOM que representa um documento HTML.

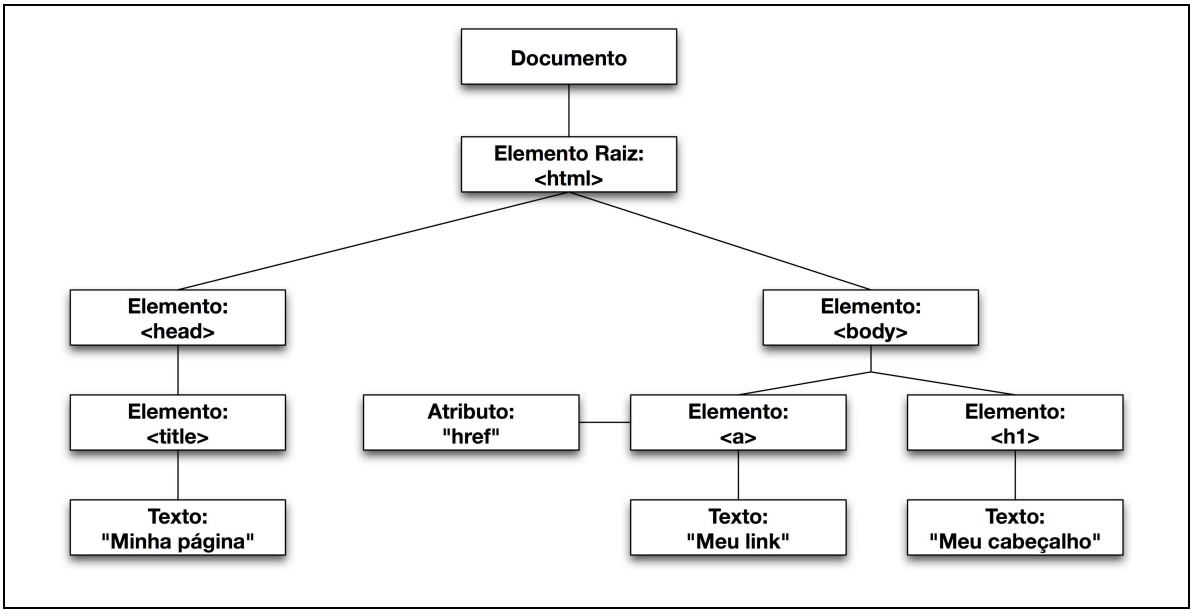

Figura 1. Documento HTML representado na árvore DOM

A representação da página no padrão DOM permite instrumentar a aplicação para que registre os eventos ocorridos dentro da estrutura da árvore, possibilitando registrar, não só o evento ocorrido, mas também identificar com precisão qual o objeto relacionado ao evento. Por exemplo, é possível registrar qual o campo de um formulário foi editado pelo usuário ou qual o botão pressionado na interface da aplicação, como ilustra o trecho de log apresentado na Tabela 1.

Tabela 1. Trecho de um log que representa a captura de eventos no browser do usuário.

IP	*Timestamp*	ID Sessão	Evento	Complemento / Identificador
10.0.0.3	2012-01-29,09:55:47	sid=htnNNAMm2spg	httptraffic	url=http://www.ic.unicamp.br/
10.0.0.3	2012-01-29,09:55:50	sid=htnNNAMm2spg	event=mousedown	link=http://www.ic.unicamp.br/news text=News
10.0.0.3	2012-01-29,09:55:50	sid=htnNNAMm2spg	httptraffic	url=http://www.ic.unicamp.br/news
10.0.0.3	2012-01-29,09:56:53	sid=htnNNAMm2spg	event=mousedown	id=searchGadget
10.0.0.3	2012-01-29,09:56:53	sid=htnNNAMm2spg	event=focus	id=searchGadget
10.0.0.3	2012-01-29,09:56:54	sid=htnNNAMm2spg	event=keypress	key=d
10.0.0.3	2012-01-29,09:56:55	sid=htnNNAMm2spg	event=keypress	key=o
10.0.0.3	2012-01-29,09:56:55	sid=htnNNAMm2spg	event=keypress	key=u
10.0.0.3	2012-01-29,09:56:55	sid=htnNNAMm2spg	event=keypress	key=t
10.0.0.3	2012-01-29,09:56:55	sid=htnNNAMm2spg	event=keypress	key=o
10.0.0.3	2012-01-29,09:56:56	sid=htnNNAMm2spg	event=keypress	key=r
10.0.0.3	2012-01-29,09:56:58	sid=htnNNAMm2spg	event=change	type=text id=searchGadget value=doutor
10.0.0.3	2012-01-29,09:56:58	sid=htnNNAMm2spg	event=mousedown	id=searchbt
10.0.0.3	2012-01-29,09:57:11	sid=htnNNAMm2spg	httptraffic	url=http://www.ic.unicamp.br/search?SearchableText=doutor

No log apresentado na Tabela 1, pode-se observar o registro das seguintes informações:

- IP – endereço IP do usuário;
- *Timestamp* – momento que o evento foi registrado;
- ID Sessão – identificador único da sessão do usuário, para possibilitar a distinção de usuários utilizando um mesmo endereço IP;
- Evento – identificação do evento registrado;
- Complemento / Identificador – especificação do evento registrado para identificação do elemento na árvore DOM ou outros atributos do evento.

Por exemplo, é possível observar na Tabela 1, que o usuário pressionou o botão do mouse (evento “mousedown“) sobre a caixa de busca (“id=searchGadget“) e posteriormente,

após a digitação do termo a ser buscado, pressionou o botão que executa a busca (“id=searchbt”).

A instrumentação de uma aplicação para registrar a interação de usuários, conforme exemplificado no trecho de log da Tabela 1, pode ser realizada de três formas: instrumentando-se a aplicação no servidor, instrumentando-se a aplicação no computador do usuário ou utilizando-se uma abordagem de *proxy* remoto.

A instrumentação realizada no computador do usuário é implementada nas abordagens WebVip (Scholtz & Laskowski, 1998), TEA (Obendorf et al., 2004), WAUTER (Balbo et al., 2005) e WebQuilt (Hong et al., 2001).

O WebVip foi uma das primeiras ferramentas desenvolvidas para capturar a interação de usuários em aplicações Web. Na utilização do WebVip é necessário que se faça uma cópia de toda a aplicação avaliada, instrumentando-a no computador do usuário de modo a incluir identificadores em cada link das páginas. Desse modo, quando esses links são acessados, o WebVip faz o registro das páginas navegadas pelo usuário em um log (Scholtz & Laskowski, 1998).

O TEA foi desenvolvido para dar suporte à execução de avaliações de usabilidade, funcionando como um *proxy* no computador do usuário, instrumentando cada página visitada (Obendorf et al., 2004). Dessa maneira, é possível capturar a sequência de páginas visitadas pelo usuário. O objetivo do TEA é automatizar a captura de dados da interação de usuários em testes realizados localmente, sem considerar testes remotos, pois a ferramenta de captura precisa estar instalada no computador do usuário. O TEA captura a sequência de páginas acessadas pelos usuários, sem registrar a interação dentro das

páginas. Além disso, tem suporte apenas à aplicações estáticas, desenvolvidas em HTML (Atterer & Schmidt, 2007).

O WAUTER (*Web Automatic Usability Testing EnviRonment*) é composto por um conjunto de ferramentas integradas e voltadas à auxiliar em avaliações automatizadas de usabilidade de aplicações Web (Balbo et al., 2005). Da mesma forma que o TEA, o WAUTER trabalha como um *proxy* instalado no computador do usuário, e também só permite a captura da interação dos usuários em aplicações estáticas, desenvolvidas em HTML. No entanto, sua captura de dados é mais detalhada que no que TEA, possibilitando o registro da interação dos usuários dentro das páginas da aplicação, como o pressionamento de botões, seleções de links e preenchimento de formulários (Balbo et al., 2005).

O WebQuilt, por sua vez, também foi desenvolvido para ser utilizado no suporte a avaliações de usabilidade em aplicações Web. Sua abordagem é bastante similar ao WAUTER e ao TEA, tendo o log da interação feito através de um *proxy* instalado no computador dos usuários. Da mesma forma que os anteriores, o WebQuilt é voltado apenas para aplicações estáticas, desenvolvidas em HTML e sua captura de dados registra apenas a sequência de páginas acessadas pelo usuário na aplicação (Hong et al., 2001).

A captura da interação baseada na instrumentação da aplicação através de software instalado nos computadores dos usuários dificulta a captura remota e automática da interação. Além do tempo e trabalho necessários na instalação das ferramentas nos computadores dos usuários, esta abordagem tem sempre a desvantagem de possuir restrições de segurança nos computadores dos usuários e restrições em relação à compatibilidade de plataformas computacionais (Hong et al., 2001).

Para facilitar a captura da interação dos usuários de forma remota, tem-se também a possibilidade de realizar a instrumentação no servidor da aplicação, como nas abordagens de WET (Etgen & Cantor, 1999), WebRemUsine (Paganelli & Paternò, 2002) e WELFIT (Santana & Baranauskas, 2008).

O WET realiza a captura da interação dos usuários de forma remota e automatizada, incluindo, além das informações de páginas requisitadas, a interação dos usuários na interface da aplicação Web, como links acessados, movimentos e cliques de mouse. No entanto, a utilização do WET requer a instrumentação de cada página no servidor Web, inserindo-se nelas código de programação em linguagem JavaScript (Etgen & Cantor, 1999). Além disso, o WET é limitado a ser utilizado em aplicações estáticas desenvolvidas em HTML.

Da mesma forma que o WET, o WebRemUsine também captura a interação dos usuários através da instrumentação de cada página da aplicação em seu servidor Web. Para realizar a captura é necessário inserir em cada página da aplicação um trecho de código de programação em linguagem JavaScript e também um *applet* em linguagem Java, que fazem a captura da interação dos usuários (Paganelli & Paternò, 2002). A vantagem do WebRemUsine em relação ao WET é que sua captura é mais detalhada, permitindo o registro de todas as ações dos usuários na aplicação, como movimentos e cliques de mouse, seleções, etc.

Do mesmo modo que o WebRemUsine, o WELFIT (*Web Event Logger and Flow Identification Tool*) também necessita que a aplicação seja instrumentada no servidor Web para que a interação dos usuários seja capturada (Santana & Baranauskas, 2008). Apesar de uma captura detalhada da interação, incluindo todos as ações dos usuários em

uma página Web, o WELFIT tem seu foco nos eventos que ocorrem em cada página, não levando em conta a navegação entre as páginas da aplicação.

A maior dificuldade em se capturar a interação dos usuários através da instrumentação da aplicação no servidor, conforme implementado nas abordagens mencionadas, é a necessidade de se alterar cada página no servidor Web da aplicação. Desse modo, além de ser necessário o acesso ao código fonte da aplicação, alterar cada página pode tornar o processo demasiadamente demorado e trabalhoso, dependendo da quantidade de páginas existentes na aplicação.

Como alternativa para esta abordagem há a possibilidade da utilização de uma abordagem de *proxy* remoto, ilustrada na Figura 2, conforme implementada no UsaProxy (Atterer et al., 2006) e em Carta et al., (2011). Esta abordagem funciona estabelecendo-se um servidor *proxy* entre os usuários e a aplicação avaliada. Este servidor, por sua vez, intercepta automaticamente cada requisição de páginas feita pelos usuários durante a interação com a aplicação. Cada página é dinamicamente instrumentada para permitir o registro dos eventos no *browser*. Após instrumentada, a página alterada é devolvida ao usuário, que tem sua interação com a aplicação realizada de maneira normal, sem nenhuma alteração.

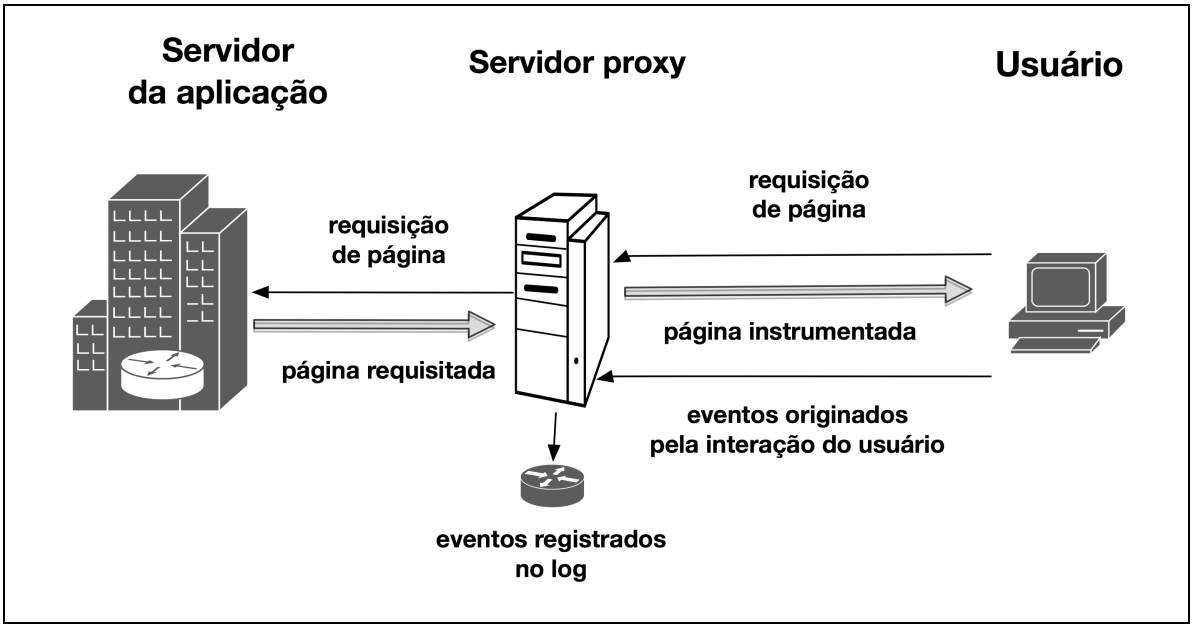

Figura 2. Arquitetura da abordagem de *proxy* remoto.

A instrumentação automática no *proxy* utiliza as características da árvore DOM para registrar os eventos ocorridos em cada objeto dentro de uma página da aplicação. A página instrumentada pelo *proxy* recebe código em linguagem de programação que detecta e armazena os eventos ocorridos durante a interação do usuário e os envia periodicamente ao *proxy*, para que sejam armazenados em log.

O processo de interceptação e instrumentação de cada página demanda tempo e processamento no servidor *proxy*. Para evitar que a interceptação e instrumentação de cada página não gere um atraso perceptível na devolução destas páginas ao usuário, é necessário garantir que a capacidade de processamento e velocidade de conexão no servidor *proxy* sejam suficientes e adequadas à sua demanda, para que o processo de captura não altere a interação do usuário com a aplicação.

Levando em conta as vantagens mencionadas anteriormente, a abordagem de *proxy* remoto constitui-se na alternativa mais adequada, entre as abordagens existentes e estudadas, para ser incorporada ao método WebHint proposto nesta tese.

A abordagem de *proxy* remoto permite realizar a captura da interação dos usuários de forma remota e automática, sem a necessidade de se instalar software nos computadores dos usuários ou nos servidores da aplicação analisada. Desse modo, a preparação da infra-estrutura para captura dos dados da interação é simples e pode ser reutilizada quantas vezes for necessário. Uma vez que o *proxy* esteja sendo executado, basta colocá-lo em frente à aplicação, isto é, configurá-lo para que receba as requisições de páginas dos usuários.

Além disso, o *proxy* utilizado em um estudo pode também ser reutilizado em diferentes estudos de usabilidade, simplesmente configurando-se o *proxy* para interceptar a interação dos usuários com uma nova aplicação em cada novo estudo de usabilidade.

Levando em conta que os usuários acessam a aplicação via *proxy* de maneira transparente, não há nenhuma alteração ou mudança na forma como estes utilizam a aplicação. Assim, o contexto de uso da aplicação é mantido, pois os usuários podem acessar a aplicação em seu ambiente cotidiano, realizando suas tarefas em uma situação real de uso. Além disso, como a captura é realizada de maneira remota e automática, pode-se realizar a captura em grande escala de usuários, monitorando, se necessário, todos os usuários que acessam a aplicação.

Desse modo, as características mencionadas justificam a escolha da abordagem de *proxy* remoto como técnica a ser utilizada para realizar a captura da interação dos usuários no método WebHint, proposto nesta tese.

A utilização da abordagem de *proxy* remoto, por realizar a captura de modo transparente para o usuário, deve sempre levar em conta questões éticas e de privacidade do usuário, ao ser implementada. Sempre que esta abordagem for utilizada, deve-se garantir que dados pessoais de identificação dos usuários, informações de segurança, como senhas ou outras informações privadas, nunca devem ser armazenadas durante a captura da interação.

2.3 Análise da Interação de Usuários

Capturar todas as ações realizadas pelos usuários na interface de uma aplicação Web de maneira automática produz uma grande quantidade de dados a ser analisada. Para tornar a análise desses dados viável, é necessário automatizar as etapas do processo de análise. Desse modo, buscou-se na literatura o estado da arte em técnicas para análise de logs de interação de usuários, visando-se identificar quais as técnicas poderiam ser incorporadas à etapa de análise de dados no método proposto nesta tese.

Para facilitar a organização desta sessão classificou-se as técnicas existentes em dois grupos: técnicas para extração de métricas e técnicas baseadas na análise de sequências de eventos. Esta classificação em dois grupos sumariza e contempla as categorias propostas por Ivory & Hearst (2001) e Hilbert & Redmiles (2000).

2.3.1 Técnicas para Extração de Métricas

Estas técnicas utilizam as informações registradas no log para extrair métricas relacionadas à performance dos usuários realizando tarefas como tempo decorrido na execução de determinada tarefa, número de tarefas completadas, número de tarefas abandonadas e erros cometidos. Além destas métricas, essas técnicas também sumarizam valores relacionados a elementos e funcionalidades da aplicação avaliada como número de acessos a determinadas funcionalidades e funcionalidades mais acessadas (Hilbert & Redmiles 2000).

Diversos trabalhos existentes na literatura utilizam a extração de métricas na análise de logs, como os trabalhos de Olsen & Halversen (1988), Sanderson et al (1994), Chang and Dillon (1997), Macleod & Rengger (1993), Rauterberg (1995), Bacheldor (1999), Paganelli & Paterno (2002), Schwerz et al., (2007).

Conforme Ivory & Hearst (2001), a utilização de técnicas de extração de métricas são adequadas quando se deseja obter valores relacionados a elementos da interface, funcionalidades, aspectos técnicos relacionados à performance das aplicações ou valores relacionados à performance dos usuários executando tarefas. Essas informações podem auxiliar na detecção de problemas de usabilidade. O tempo médio gasto pelos usuários na execução de determinada tarefa pode fornecer, por exemplo, a indicação de possíveis problemas na interface, caso o tempo gasto pelos usuários seja maior do que o esperado.

No entanto, é difícil detectar problemas de usabilidade com base apenas nas métricas obtidas, pois estas não fornecem informações detalhadas a respeito dos dados coletados (Hilbert & Redmiles 2000). Desse modo, não é possível obter informações sobre a

maneira com a qual os usuários executam suas tarefas na interface da aplicação. Por exemplo, é possível apenas saber quantas vezes um usuário completou uma determinada tarefa, observando-se o número de vezes que este executou a ação que inicia a tarefa e quantas vezes executou a ação que completa a tarefa. No entanto, não é possível saber, por exemplo, por que um determinado usuário iniciou a referida tarefa 5 vezes, mas apenas chegou ao final da mesma uma única vez.

Assim, as técnicas para extração de métricas são geralmente utilizadas em conjunto com outras técnicas de modo a complementar as informações obtidas através de outros métodos (Hilbert & Redmiles 2000).

Diferentemente das técnicas de extração de métricas, as técnicas para análise de sequências de eventos em logs levam em conta a forma com a qual os usuários executam suas tarefas na interface da aplicação, isto é, baseiam a análise na sequência de eventos registrados, não apenas em eventos isolados. Dessa maneira, estas técnicas permitem obter informações mais detalhadas a respeito do comportamento dos usuários na interface da aplicação avaliada. O estudo realizado sobre as técnicas para análise de sequências é apresentado a seguir.

2.3.2 Técnicas para Análise de Sequências

As técnicas para análise de sequências de eventos levam em conta a relação entre os eventos registrados no log, não somente dados quantitativos relacionados a eventos isolados, como as técnicas de extração de métricas. Assim, tem-se a possibilidade de analisar o log em uma perspectiva orientada às tarefas executadas pelos usuários. Ou seja, cada tarefa executada pelos usuários na interface da aplicação é registrada no log como

uma sequência de eventos que pode ser analisada posteriormente. Desse modo, é possível obter em detalhes as ações executadas pelos usuários na realização de suas tarefas na interface da aplicação, pois cada ação executada é registrada no log na forma de um evento. Um evento consiste em uma ação do usuário que gera um resposta da aplicação, registrada no log com seu *timestamp* e outras informações que permitem a identificação do evento.

As técnicas utilizadas para analisar logs levando-se em conta as sequências de eventos registradas podem ser classificadas em dois grupos: técnicas para detecção de sequências e técnicas para comparação de sequências.

2.3.2.1 Técnicas para detecção de sequências

O objetivo destas técnicas é a busca por determinadas sequências de eventos dentro do log. Geralmente a detecção de sequências visa encontrar padrões de eventos que se repetem no log (Hilbert & Redmiles 2000).

Por exemplo, as técnicas de Ciclos de Fisher permitem que determinando-se um evento como ponto de início e outro evento como ponto final, encontre-se todas as sequências começando e terminando com estes eventos (Fisher, 1991).

A técnica *Lag Sequential Analysis* (LSA) tem o objetivo de identificar correlações entre eventos em um log. Partindo da seleção de um evento chave e outro evento alvo, a técnica visa encontrar no log a quantidade de vezes que o evento alvo aparece antes ou após o evento chave (Sackett,1978).

A técnica *Maximal Repeating Pattern* (MRP) identifica os padrões que ocorrem repetidamente em uma sequência, listando-os em ordem de tamanho da subsequência encontrada, seguido da quantidade de vezes que o padrão se repete (Siochi & Hix, 1991). A técnica se assemelha à técnica de ciclos de Fisher, no entanto, não se especifica particulares eventos de interesse.

Diversas outras técnicas descritas em Sanderson & Fisher (1994), Gottman & Roy (1990), Santos e Badre (1994) e Hilbert & Redmiles (1998) visam a detecção de padrões de sequências em logs. Conforme Hilbert & Redmiles (2000), essas técnicas têm limitações, pois produzem uma quantidade muito grande de dados descontextualizados, tornando a análise de seus resultados extremamente trabalhosa e demorada.

Outra limitação das técnicas para detecção de sequências é trabalhar com padrões estritos de sequências, que dificilmente ocorrem em logs de eventos de usuários (Hilbert & Redmiles 2000). Por exemplo, na execução de determinada tarefa, os usuários podem executar eventos não previstos, como cliques, seleções, digitações, que não alteram o resultado da tarefa, mas geram diferentes eventos na sequência. Esses eventos dificultam a execução das técnicas de busca por padrões, levando a resultados não satisfatórios em sua utilização.

De modo geral, as técnicas para detecção de sequências têm em seus objetivos a busca por trechos de eventos que se repetem com frequência no log. No entanto, simplesmente detectar eventos repetidos no log não fornece informações suficientes a respeito das tarefas executadas para auxiliar na detecção de problemas de usabilidade. Desse modo, buscou-se técnicas que favorecessem a análise levando em conta as tarefas executadas, representadas no log como sequências de eventos. Assim, é possível se obter informações

mais detalhadas sobre a interação dos usuários, podendo-se detectar problemas na execução das tarefas.

Dessa forma, outro importante grupo de técnicas para a análise de logs, estudadas durante esta pesquisa, são as técnicas para comparação de sequências, conforme apresenta-se a seguir.

2.3.2.2 Técnicas para comparação de sequências

O objetivo destas técnicas é a comparação entre sequências visando medir a diferença ou a similaridade entre elas. Estas técnicas são muito utilizadas nas áreas de bioinformática, especialmente no estudo de sequências de DNA, RNA e proteínas (Setubal & Meidanis, 1997). As principais técnicas utilizadas quando se deseja comparar sequências são Distância de Edição e Alinhamento de Sequências (Gusfield, 1997), conforme descreve-se a seguir.

2.3.2.2.1 Distância de edição

A comparação de sequências através da técnica de distância de edição permite medir a diferença entre duas *strings* com base na quantidade de operações de edição necessárias para se transformar a primeira *string* na segunda (Gusfield, 1997). As três operações possíveis são: a inserção de um caractere na primeira *string*, a exclusão de um caractere da primeira *string* ou a substituição de um caractere na primeira pelo respectivo caractere existente na mesma posição na segunda *string*. A Figura 3 ilustra a comparação da *string* 1 "vintner" com a *string 2* "writers". As operações de inserção são identificadas pela letra "I", as de exclusão pelo letra "E" e as operações de substituição pelo letra "S". Quando há

uma correspondência entre os caracteres nas duas *string*s utiliza-se a letra "C" para representá-la.

Operações:	**S**	**I**	C	**E**	C	**E**	C	C	**I**
String 1:	v		i	n	t	n	e	r	
String 2:	w	r	i		t		e	r	s

Figura 3. Exemplo de uma comparação entre duas *string*s utilizando a técnica de distância de edição.

No exemplo ilustrado na Figura 3, pode-se perceber que o caractere "v" na primeira *string* requer uma operação de substituição por "w", o caractere "r" é inserido em seguida, e o caractere "i" é encontrado também na segunda *string*, não necessitando de nenhuma operação. Assim, sucessivamente as operações são executadas para transformar a primeira *string* na segunda.

A distância de edição é, então, definida como o número de operações necessárias para transformar a *string* 1 na *string* 2. No referido exemplo, cinco operações são executadas, caracterizando a distância de edição entre S1 e S2.

2.3.2.2.2 Alinhamento de sequências

Outra técnica para a comparação de sequências é o alinhamento. O alinhamento de duas *strings* é obtido inserindo-se espaços (-) entre os caracteres de ambas as *strings*, visando preservar as equivalências e de modo que cada caractere em uma *string* esteja alinhado a outro caractere ou espaço na outra, conforme o exemplo da Figura 4 (Gusfield, 1997).

String 1:	**q**	**a**	c	-	d	**b**	d
String 2:	**q**	**a**	w	x	-	**b**	-

Figura 4. Exemplo de um alinhamento de sequências.

No alinhamento ilustrado na Figura 4, os caracteres "*q"*, "*a"* e "*b"* têm correspondência na outra *string*, os caracteres "c" e "w" não têm correspondência e os caracteres "x" e "d" estão alinhados a espaços (-) inseridos.

Do ponto de vista matemático, distância de edição e alinhamento de sequências são métodos equivalentes para representar a relação entre duas *strings*, ou seja, um alinhamento pode ser convertido em uma distância de edição e vice-versa (Gusfield, 1997).

Tanto a técnica de distância de edição, quanto a técnica de alinhamento de sequências podem trabalhar com a atribuição de diferentes pesos às suas operações, ampliando as possibilidades de utilização dessas técnicas. Da atribuição de pesos às operações em um alinhamento de sequências deriva uma importante técnica conhecida como *Longest Common Subsequence (LCS),* que visa trabalhar com sequências que não possuem alta similaridade, se analisados todos os elementos nas sequências, mas que podem possuir maior similaridade se analisadas desconsiderando-se alguns elementos (Gusfield, 1997).

Longest Common Subsequence (LCS)

Conforme Gusfield (1997), a definição de uma subsequência pode ser descrita como: "em uma determinada *string* S, uma subsequência consiste em um subconjunto de caracteres

de S, dispostos na ordem original em que aparecem". A Figura 5 ilustra exemplos de subsequências em uma *string*.

***String* S:**	Hum**an**i**da**de
Subsequência 1:	"idade"
Subsequência 2:	"a n d a"

Figura 5. Exemplos de subsequências da *string* S

Uma subsequência pode ser contínua, como a subsequência 1, "idade", que também é considerada uma *substring*, dada a sua característica contínua. No entanto, para ser uma subsequência, esta não necessita ser contínua, como no exemplo da subsequência 2, "a, n, d, a", que também é uma subsequência de S. Desse modo, conforme Gusfield (1997), a definição de subsequência comum é: "considerando-se duas *string*s S1 e S2, uma subsequência comum é uma subsequência que aparece em ambas S1 e S2. Assim, o problema de encontrar a maior subsequência comum (LCS) consiste em encontrar a maior subsequência que aparece em ambas S1 e S2".

Uma simples forma de encontrar a maior subsequência comum entre duas *string*s é, por exemplo, atribuindo o peso 1 para cada equivalência de caracteres e o peso 0 para cada não correspondência ou espaço em branco em um alinhamento.

2.3.2.3 Comparação de sequências e análise de logs de interação

Apesar de serem mais utilizadas em bioinformática, as técnicas para comparação de sequências podem ser também utilizadas na análise de sequências de eventos originados na interação de usuários com aplicações.

Conforme Hilbert & Redmiles (2000), estas técnicas permitem comparar sequências extraídas de logs de interação com sequências alvo definidas. As sequências alvo podem ser sequências "ideais", que definem o comportamento esperado na execução de certas tarefas pelos usuários. Neste caso, diversas abordagens visam encontrar discrepâncias entre as sequências "ideal" e "real" para auxiliar na detecção de problemas de usabilidade.

As primeiras abordagens desenvolvidas para auxiliar na análise de logs de interação utilizando a comparação de sequências são descritas nos trabalhos de Finlay & Harrison (1990), Sanderson & Fisher (1994), Uehling & Wolf (1995) e Balbo (1996). As ferramentas descritas nestes trabalhos realizavam a comparação de sequências exibindo as diferenças entre duas sequências fornecidas.

Mais recentemente, outros trabalhos desenvolvidos utilizam a comparação de sequências para automatizar a análise dos dados obtidos na captura da interação de usuários com aplicações. Estes, geralmente comparam uma sequência ideal, que define o comportamento esperado para a execução de certa tarefa, com a sequência real executada por um usuário na execução da tarefa, como o USINE (Lacerof & Paternò, 1998), RemUsine (Paternò & Ballardin, 1999) e WebRemUsine (Paganelli & Paternò, 2002). Estas ferramentas, baseiam-se na comparação de um modelo abstrato definido para a execução de uma tarefa, com a sequência de eventos real executada pelo usuário e registrada em um log. Como resultado, apontam as diferenças entre as duas sequências.

Outros trabalhos, como o AWUSA (Buchholz et al., 2007) e o WAUTER (Balbo et al., 2005), utilizam a comparação de sequências para identificar padrões de comportamento que estão relacionados a problemas de usabilidade previamente detectados. Eles

comparam sequências "problema", com sequências de eventos existentes nos logs de interação dos usuários. Desse modo, se uma sequência "problema" aparece no log, esta indica a existência do problema relacionado.

Outros trabalhos desenvolvidos para auxiliar na análise de logs utilizam a comparação de sequências para realizar a sumarização dessas sequências de forma visual. Desse modo, é possível visualizar os caminhos mais comuns executados pelos usuários na interação com a aplicação avaliada.

Os primeiros trabalhos desenvolvidos visando auxiliar na visualização de um log através da sumarização de sequências de eventos registradas são descritos em Helfrich & Landay (1999) e Al-Qaimari & McRostie (1999).

A ferramenta QUIP, proposta por Helfrich & Landay (1999), permite sumarizar os eventos representados por diversas sequências referentes à execução de uma tarefa em forma de um grafo. A representação da tarefa no grafo leva em conta o *timestamp* e a quantidade de vezes que determinado evento aparece para representar com diferentes cores e densidades os vértices e arestas do grafo.

A ferramenta KALDI (Al-Qaimari & McRostie, 1999) também representa a execução de eventos em forma de um grafo e permite associar os eventos existentes no log a *screenshots* obtidos na interface para permitir a reprodução das sequências de forma visual.

Mais recentemente, o WebQuilt proposto por Hong et al. (2001), auxilia na análise de logs de interação de maneira similar ao KALDI e QUIP. O WebQuilt faz um sumário das

páginas visitadas pelos usuários na forma de um grafo indicando os caminhos executados entre as páginas de uma aplicação Web. Da mesma forma que o KALDI, as conexões entre os vértices do grafo têm sua densidade variada dependendo da quantidade de vezes que os usuários executaram o referido caminho.

O WELFIT (Santana & Baranauskas, 2008), também baseia sua análise na sumarização da interação dos usuários em sua representação na forma visual de um grafo. Diferentemente das abordagens anteriores o WELFIT é voltado para analisar a interação dos usuários dentro de uma única página, isto é, o WELFIT sumariza todos os eventos ocorridos dentro de uma página, apresentando como resultado um grafo que demonstra o fluxo de ações dos usuários. No entanto, o WELFIT não sumariza a navegação entre páginas da aplicação analisada, tal qual o WebQuilt ou KALDI.

O ExperiScope (Guimbretière et al., 2007) também tem o objetivo de sumarizar e exibir de forma gráfica a interação dos usuários registrada em logs de eventos. O ExperiScope, no entanto, é voltado para a análise levando em conta apenas os eventos de mouse e teclado ocorridos, utilizando o modelo KLM *(Key-Stroke Level Model)*. Desse modo, da mesma forma que o WELFIT, o ExperiScope, não dá suporte à análise da interação dos usuários navegando por diferentes páginas de uma aplicação Web.

Conforme Carta et al. (2011), o WUP (*Web Usability Probe*) não sumariza a interação dos usuários, mas exibe as sequências de eventos de forma gráfica, representando-os em forma de uma linha de tempo, para facilitar a análise visual das sequências registradas nos logs.

2.3.2.4 Considerações sobre análise de sequências em logs de interação

As técnicas para análise de sequências oferecem recursos para automatizar e facilitar o processo de análise de logs de eventos obtidos na interação de usuários com aplicações. Comparadas às técnicas de extração de métricas, as técnicas de análise de sequências oferecem informações mais detalhadas sobre a execução de tarefas do que as técnicas de extração de métricas, permitindo uma análise mais detalhada da interação dos usuários com a aplicação avaliada. Desse modo, é possível obter uma visão mais precisa de como os usuários executam suas tarefas na aplicação, o que favorece a detecção de problemas de usabilidade (Hilbert & Redmiles 2000).

No entanto, conforme Ivory & Hearst (2001) a maior dificuldade na utilização dessas técnicas, quando se objetiva automatizar o processo de análise, é a natureza dos eventos obtidos com a interação de usuários. Eventos obtidos com a interação de usuários usualmente possuem muitos "ruídos".

Segundo Hilbert & Redmiles (2000), esses ruídos podem ser caracterizados por grupos de eventos que, mesmo não sendo relevantes em determinadas análises, estão presentes no log e podem alterar os resultados da análise caso não sejam adequadamente tratados.

Em determinados casos não é importante, por exemplo, analisar o que os usuários digitam no teclado quando interagem com uma aplicação. Desse modo, a informação de cada caractere digitado pelo usuário pode representar um ruído no log. Quanto mais detalhada é a captura dos eventos, maior o número de informações registradas e consequentemente a quantidade de ruídos.

Desse modo, um passo essencial na análise de sequências de eventos é a "limpeza" ou pré-processamento do log, isto é, selecionar no log apenas as informações relevantes para a análise (Hilbert & Redmiles 2000). Hilbert & Redmiles (2000) descrevem três atividades básicas no pré-processamento de logs: seleção, abstração e recodificação.

Seleção consiste em subtrair do log informações que não são relevantes, permitindo que eventos e sequências de interesse ganhem mais destaque. Por exemplo, em determinados casos pode não ser relevante analisar os movimentos de mouse registrados no log. Suprimindo-se esses eventos, evidencia-se eventos de mais alto nível de abstração, como o pressionamento de botões e seleções de menus, por exemplo.

Abstração consiste em combinar ou agrupar eventos relacionados de modo a gerar eventos de mais alto nível de abstração. Por exemplo, uma sequência de eventos como: atribuição de foco em um campo de um formulário, seguido pela digitação de caracteres, pode ser convertido em um único evento que indique o preenchimento de determinado campo em um formulário.
Recodificação consiste em gerar uma nova sequência de eventos com base nas alterações geradas pela seleção e/ou abstração de eventos. Desse modo, tem-se um log com menos ruídos para ser analisado, gerando melhores resultados.

Este pré-processamento é realizado de maneira automatizada utilizando-se *parsers* ou algoritmos que selecionam grupos de eventos a serem retirados do log, por exemplo, movimentos do mouse e digitação de caracteres, etc (Hilbert & Redmiles, 2000).

Apesar da possibilidade de se automatizar o pré-processamento do log visando diminuir os ruídos que dificultam a análise, logs de interação possuem ainda outras características

que dificultam esta atividade. Alguns eventos não podem simplesmente ser eliminados do log de maneira automatizada, pois podem ser irrelevantes em uma determinada tarefa, mas muito importantes na análise de outras. Por exemplo, alguns usuários costumam selecionar o texto quando fazem uma leitura, ou costumam, clicar com o mouse na janela durante a leitura de um texto. Esses eventos, apesar de irrelevantes para a execução da tarefa de leitura, não podem ser eliminados do log de maneira geral, pois cliques na janela e seleção de texto podem ser eventos importantes na execução de outras tarefas. No entanto, em uma análise automatizada estes eventos podem alterar os resultados. Por exemplo, mesmo tarefas executadas da mesma maneira, se desconsiderados os referidos eventos, podem apresentar diferenças significativas caso sejam submetidas a uma comparação automática de sequências.

Desse modo, levando em contas as limitações das técnicas apresentadas anteriormente, estudou-se *Process Mining* como alternativa para análise dos logs de eventos obtidos na interação dos usuários com aplicações. Apesar de serem utilizadas geralmente na análise de logs de eventos relacionados a processos de negócio (Aalst, 2011), as técnicas de *process mining* podem ser utilizadas também para automatizar e facilitar as etapas de análise de logs de eventos de interação.

2.3.3 *Process Mining*

Conforme Aalst (2011), *process mining* tem como principal objetivo obter informações sobre um processo através da análise de uma série de reais execuções do mesmo, registradas e armazenadas em logs de eventos. Ou seja, a idéia básica por trás de *process mining* é descobrir, monitorar e melhorar processos, através da extração de conhecimento de arquivos de log.

Diferentemente de outras técnicas de análise, *process mining* visa criar automaticamente um modelo consistente com a execução real do processo, observada nos eventos registrados no log. Dessa forma, é possível obter informações precisas sobre um processo, isto é, a representação do que está registrado no log e não apenas um modelo idealizado da realidade (Aalst, 2004). A idéia básica da extração de conhecimento de logs através da utilização de *process mining* é ilustrada na Figura 6.

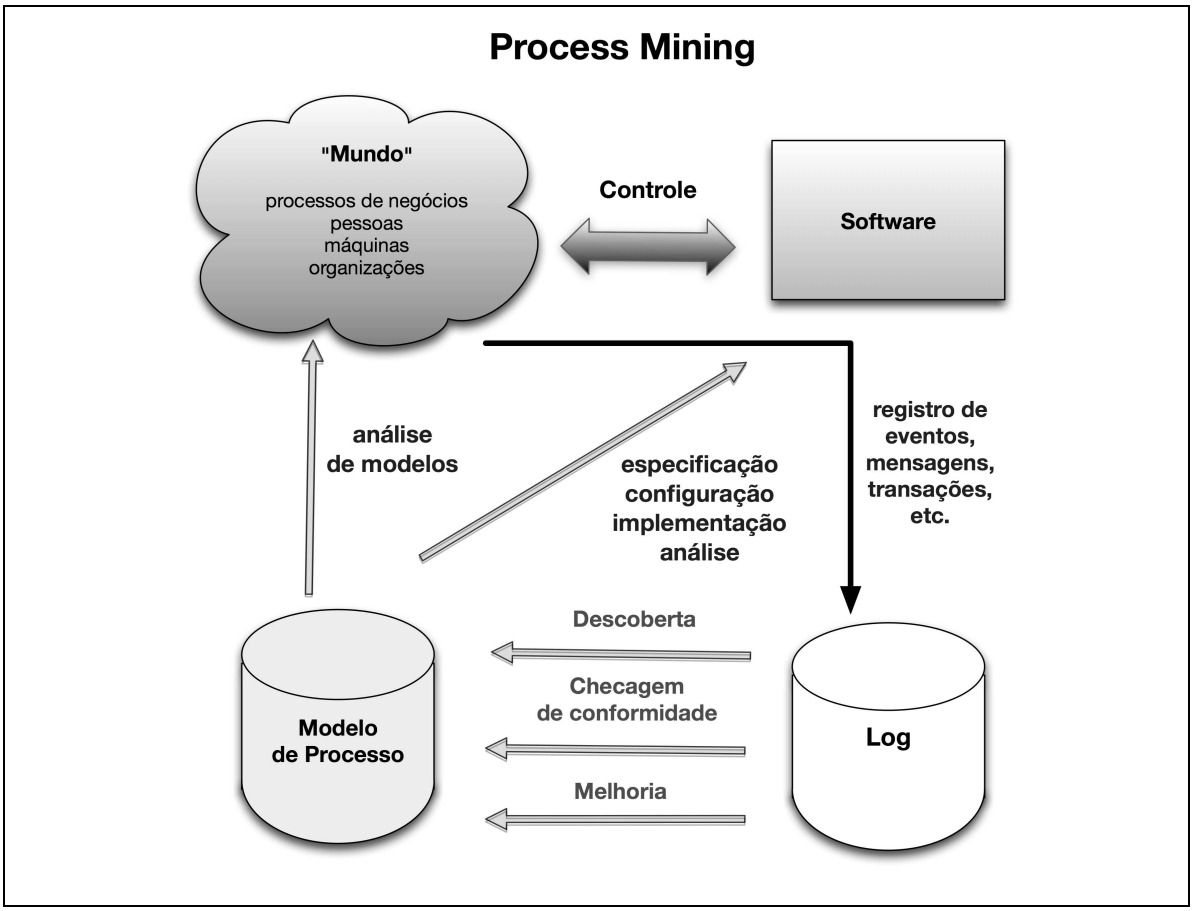

Figura 6. Extração de conhecimento utilizando *process mining* (Aalst, 2011).

A Figura 6 demonstra que eventos gerados por processos reais são registrados em logs que, por sua vez, são analisados automaticamente para se obter informações sobre o processo. O requisito básico para uma análise através de *process mining* é que as informações constantes no log sejam armazenadas de modo que a ordem em que os eventos ocorrem possa ser recuperada. A seguir, apresenta-se a nomenclatura dos elementos básicos em um log de eventos:

- **Processo**: um processo é constituído por um modelo de sequência de atividades.
- **Atividade**: cada evento armazenado no log se refere a uma "atividade", isto é, um passo bem definido no processo.
- **Caso**: um caso é definido como uma instância do processo, ou seja, uma execução do mesmo.

Para contextualizar os elementos apresentados anteriormente, descreve-se a seguir o exemplo da análise de um log utilizando *process mining*. Considere o log representado na Tabela 2, que contém informações sobre 4 instâncias (casos) de um processo. O processo consiste no controle de multas de trânsito.

Tabela 2. Exemplo de um log de eventos.

Caso ID	Atividade	Origem	*Timestamp*
1	Registrar multa	Anne	20-07-2011 14:00:00
2	Registrar multa	Anne	20-07-2011 15:00:00
1	Enviar boleto	Sistema	20-07-2011 15:05:00
2	Enviar boleto	Sistema	20-07-2011 14:07:00
3	Registrar multa	Anne	21-07-2011 10:00:00
3	Enviar boleto	Sistema	21-07-2011 14:00:00
4	Registrar multa	Anne	22-07-2011 11:00:00
4	Enviar boleto	Sistema	22-07-2011 11:10:00
1	Processar pagamento	Sistema	24-07-2011 15:05:00
1	Fechar caso	Sistema	24-07-2011 15:06:00
2	Enviar reaviso	Mary	20-08-2011 10:00:00
3	Enviar reaviso	John	21-08-2011 10:00:00
2	Processar pagamento	Sistema	22-08-2011 09:05:00
2	Fechar caso	Sistema	22-08-2011 09:06:00
4	Enviar reaviso	John	22-08-2011 15:10:00
4	Enviar reaviso	Mary	22-08-2011 10:00:00
4	Processar pagamento	Sistema	29-08-2011 14:01:00
4	Fechar caso	Sistema	29-08-2011 17:30:00
3	Enviar reaviso	John	21-09-2011 10:00:00
3	Enviar reaviso	John	21-10-2011 10:00:00
3	Processar pagamento	Sistema	25-10-2011 14:00:00
3	Fechar caso	Sistema	25-10-2011 14:01:00

Partindo do log ilustrado na Tabela 2, algoritmos de *process mining* podem extrair o modelo ilustrado na Figura 7.

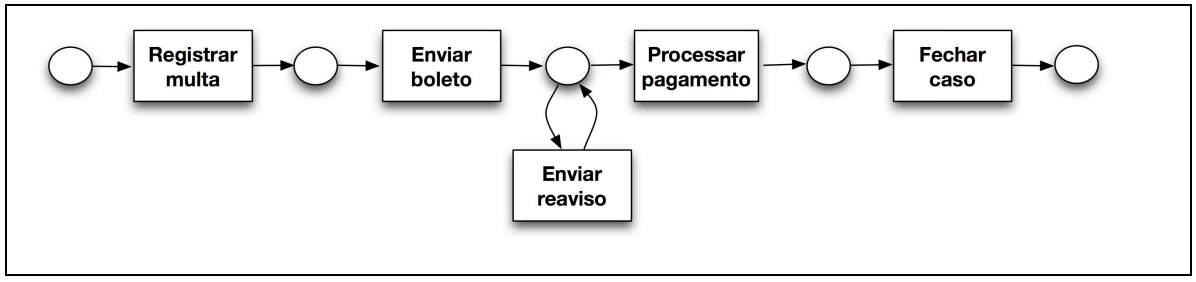

Figura 7. Exemplo de modelo extraído do log de eventos descrito na Tabela 2.

O processo representado na Figura 7 descreve que, após entrar no sistema, a multa é enviada para o proprietário do veículo. Se o proprietário não pagar a multa em até um mês, um reaviso é enviado. Quando a multa é paga, o caso é arquivado.

Dependendo do tipo de informações fornecidas pelo log, diferentes perspectivas podem ser utilizadas na análise. Conforme ilustrado na Tabela 2, o log fornece as atividades executadas e também seu *timestamp*. Desse modo, é possível identificar a ordem de execução dessas atividades e extrair o modelo do processo em uma perspectiva de **controle de fluxo.** De modo geral, a perspectiva de controle de fluxo tem seu foco na ordem das atividades e visa extrair todos os possíveis caminhos de execução do processo.

O mesmo log contém informações sobre quem executou cada atividade (pessoa/sistema). Assim, é possível realizar uma análise em uma **perspectiva organizacional**. Essa perspectiva permite descobrir informações sobre a "rede social" do processo. Por exemplo, no referido log, é possível observar que Anne transfere trabalho para Mary (caso 2) e para John (casos 3 e 4). John, por sua vez, transfere trabalho para Mary. Analisando o log, algoritmos de *process mining* podem demonstrar que Mary nunca envia um reaviso mais de uma vez, enquanto John faz isso com mais frequência. Com esta informação é possível investigar se Mary utiliza uma abordagem diferente de John ao enviar um reaviso, que resulta no pagamento mais rápido da multa pelo cliente.

Se o log contém informações adicionais, como valores relacionados às atividades, pode-se realizar a análise em uma **perspectiva de caso**. De modo geral, a perspectiva de caso foca nas propriedades dos casos e nos valores relacionados aos seus elementos. Assim, uma previsão sobre a execução de novos casos pode ser feita baseando-se em casos registrados anteriormente. Por exemplo, se o log do exemplo anterior possuísse

informações sobre o perfil dos proprietários dos carros (idade, gênero, carro, etc), estas informações poderiam auxiliar a prever se uma determinada multa seria paga dentro do prazo, dependendo do perfil do proprietário.

Além das perspectivas anteriores, existe a **perspectiva de tempo,** com a qual a análise pode ser realizada. Esta perspectiva leva em conta o tempo e a frequência com a qual os eventos acontecem. Desse modo, a perspectiva de tempo pode fornecer informações, por exemplo, sobre gargalos na execução de um processo, ou prever o tempo despendido em futuras execuções do processo, levando em conta os tempos decorridos em execuções anteriores.

Além de diferentes perspectivas de análise, *process mining* possui três diferentes técnicas, que definem a maneira com a qual o log é analisado e quais os resultados obtidos na análise. Estas técnicas são conhecidas como: técnica de descoberta, técnica de checagem de conformidade e técnica de melhoria de processo.

A **técnica de descoberta** produz um modelo de processo utilizando apenas as informações constantes no log de eventos. Partindo do log, algoritmos de *process mining* extraem o modelo do processo.

A **técnica de checagem de conformidade** compara um modelo de processo existente com as informações existentes no log. Essa técnica pode ser utilizada para verificar se um modelo do processo existente se reflete no log e vice-versa.

A **técnica de melhoria de processo** tem como objetivo melhorar um modelo de processo, levando em conta sua real execução. Um dos tipos de melhorias é chamado de reparação,

ou seja, modificar um modelo para melhor refletir a realidade. Por exemplo, se duas atividades são modeladas sequencialmente, mas na realidade podem acontecer em qualquer ordem, o modelo pode ser corrigido para refletir essa realidade.

2.3.3.1 Process mining e logs de eventos de usuários

Apesar de *process mining* ser utilizado geralmente na análise de processos de negócio, suas técnicas podem ser utilizadas também para analisar logs contendo o registro da interação de usuários com interfaces de software ou dispositivos eletrônicos.

Uma tarefa executada por um usuário na interface de um software pode ser representada como um modelo de processo. Os eventos que compõem a execução de uma tarefa podem ser representados como atividades em um processo, pois tal qual atividades em um processo, estas possuem característica sequencial de execução. Da mesma forma, cada execução de uma tarefa por um usuário na interface de um software pode ser representada por um caso na execução de um processo, conforme ilustra a Figura 8. Desse modo, as técnicas de *process mining* podem ser utilizadas na análise de logs contendo a interação de usuários.

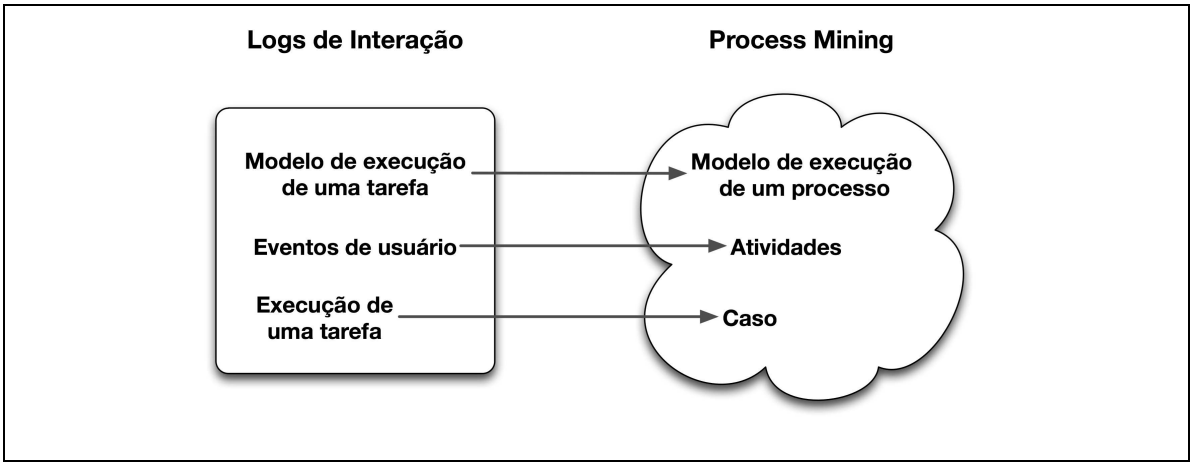

Figura 8. Correlação entre conceitos e terminologias entre logs de interação e *process mining.*

Apesar das possibilidades existentes em se utilizar *process mining* como alternativa para auxiliar na análise de dados da interação de usuários, esta é uma área ainda pouco explorada. Dentre os poucos exemplos de trabalhos realizados nesta área estão os trabalhos de Hofstra (2009) e Funk et al. (2010).

O trabalho de Hofstra (2009) descreve uma pesquisa em que *process mining* foi utilizado para fornecer informações complementares sobre a interação de usuários com uma TV interativa, isto é, um aparelho de TV conectado à internet com funcionalidades interativas. A pesquisa teve como objetivo avaliar qual o impacto que o conhecimento em informática e dispositivos eletrônicos possui na interação com um novo dispositivo. A interação dos usuários em testes de usabilidade foi gravada em vídeo e estes vídeos foram posteriormente assistidos por especialistas que registraram eventos de interesse em um log, com auxílio de um software para registrar o *timestamp* de cada evento selecionado. Estes logs gerados foram, então, analisados utilizando-se *process mining* para se obter

uma visão geral dos padrões de interação que os usuários apresentaram na utilização do novo dispositivo. O modelo obtido através de *process mining,* a partir dos eventos registrados nos logs, permitiu verificar que usuários com mais conhecimento em informática e experiência na utilização de dispositivos eletrônicos realizaram as tarefas com menos dificuldades, tendo um padrão de interação mais simples e eficiente do que usuários com menos experiência (Hofstra, 2009).

Em Funk et al. (2010) é proposto um framework para a captura e análise da interação de usuários com dispositivos eletrônicos de modo a monitorar a utilização desses produtos por usuários. No framework proposto, os dispositivos são instrumentados e recebem um componente que é integrado aos dispositivos para gerar o log de interação dos usuários.

Conforme descrito em Funk et al. (2010), foi desenvolvido um protótipo para realizar a captura da interação dos usuários em uma TV interativa. A TV, além de permitir a visualização de vídeos, possuía uma funcionalidade de recomendação de vídeos, levando em conta o vídeo atual sendo exibido ao usuário. Além dessas funcionalidades a TV também possuía a possibilidade de busca por vídeos em um acervo.

Em um experimento descrito em Funk et al. (2010), os autores tinham o objetivo de analisar como os usuários interagiam com a interface da TV interativa. No experimento, *process mining* foi utilizado para analisar logs de usuários interagindo com a interface do dispositivo, obtendo-se modelo que representava o padrão comum de interação dos usuários. Dentre as descobertas relatadas, os avaliadores verificaram que os usuários apresentaram dificuldades em interagir com a funcionalidade de busca, devido a forma de inserção de caracteres disponível na TV (Funk et al., 2010).

No contexto desta tese, tem-se o objetivo de utilizar as técnicas de *process mining* para analisar logs de interação de usuários em aplicações Web. Para este objetivo tem-se a possibilidade de utilizar a técnica de **descoberta**, em uma **perspectiva de controle de fluxo,** para se extrair o modelo do processo (neste caso de uma tarefa) de um log que contém a execução da referida tarefa por usuários em uma aplicação.

O modelo extraído representa a execução da tarefa, sendo consistente com a real execução da mesma, pois se baseia nas informações existentes no log. Assim, é possível comparar se o modelo extraído do log é similar ao modelo esperado para a execução da tarefa. Diferenças entre a real execução da tarefa e a maneira esperada podem auxiliar na detecção de problemas na interface da aplicação, que levam os usuários a executarem a tarefa de maneira diferente da esperada.

Até o momento em que esta tese foi publicada não foram encontrados na literatura trabalhos que tenham explorado a possibilidade de utilização de *process mining* na análise de logs de interação de usuários com aplicações Web.

2.3.4 Técnicas para Análise de Logs Selecionadas para Compor o Método WebHint.

Levando-se em conta as características das técnicas estudadas e descritas anteriormente para a análise de logs de interação de usuários, selecionou-se algumas destas técnicas para compor o processo de análise automatizada de dados no método WebHint, proposto nesta tese. A técnica de comparação de sequências *Longest Common Subsequence (LCS)* foi selecionada para ser utilizada na extração de tarefas dos logs analisados. Esta técnica é adequada a essa atividade, pois permite que se analise sequências com similaridade parcial. Desse modo, é possível detectar tarefas incompletas, isto é, sequências que

possuem certo grau de similaridade entre si, levando em conta parte dos eventos na sequência.

Além disso, dentre as técnicas existentes e estudadas, LCS é a técnica mais adequada para se trabalhar com sequências de eventos obtidas em logs de interação, que possuem naturalmente muitos "ruídos". A técnica LCS tem a vantagem de lidar com os ruídos existentes, pois permite que estes possuam menos peso na comparação de similaridade entre as sequências. Assim, é possível detectar se determinado trecho do log analisado contém a tarefa procurada, mesmo que este possua eventos não previstos na execução da tarefa. Além da técnica de LCS, selecionou-se técnicas de transformação para o pré-processamento do log antes da aplicação das demais técnicas na análise.

Desse modo, aplica-se técnicas para selecionar os eventos mais relevantes, dependendo do tipo de análise realizada, para se obter um melhor resultado na análise automatizada do log.

Além da técnica LCS e das técnicas para transformação do log, selecionou-se *process mining* para a análise das sequências de eventos que representam as tarefas extraídas do log. *Process mining* foi selecionado, pois permite obter automaticamente o modelo que representa a execução real das tarefas analisadas, a partir das informações constantes no log. Desse modo, conforme já mencionado, esta técnica permite uma análise da interação dos usuários como um todo, através da obtenção automática do modelo que representa os padrões reais de execução das tarefas e sua comparação com a execução esperada das tarefas analisadas.

A seguir, no Capítulo 3, apresenta-se o método WebHint proposto nesta tese para a avaliação de usabilidade de aplicações Web, descrevendo-se em detalhes a utilização das técnicas selecionadas para sua composição.

Capítulo 3

O Método WebHint proposto

A partir do estudo realizado e apresentado anteriormente, desenvolveu-se o método WebHint. Este método tem o objetivo de apresentar-se como uma alternativa para avaliação de usabilidade de aplicações Web, baseando-se na captura e análise automatizada da interação de usuários.

O WebHint propõe uma análise em uma perspectiva diferente dos métodos tradicionais, analisando os padrões obtidos na interação dos usuários com a aplicação. Essa análise é realizada considerando os usuários com um todo, não analisando-os de forma individualizada. Desse modo, o WebHint visa oferecer uma visão abrangente da utilização da aplicação, não obtida na utilização de métodos tradicionais.

Além disso, todo o processo automatizado de captura e análise de dados no WebHint visa manter um baixo custo na execução da avaliação, favorecendo a realização de avaliações periódicas de usabilidade.

Em busca dos objetivos descritos anteriormente, o método WebHint foi desenvolvido e é composto por três principais fases, a definição das tarefas, a captura da interação dos usuários e a análise desses dados, conforme ilustra a Figura 9.

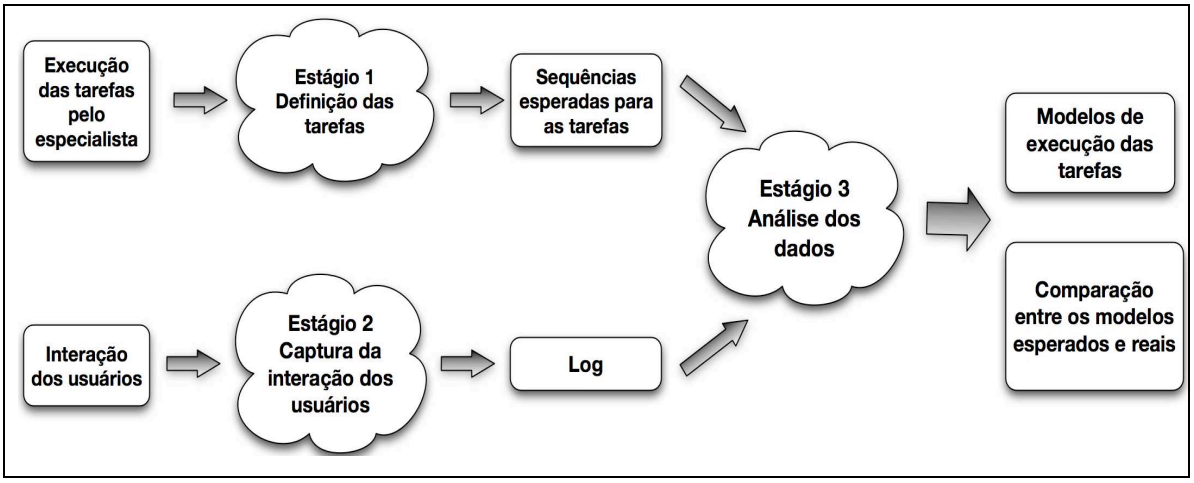

Figura 9. O método WebHint proposto.

3.1 Definição das Tarefas

Na concepção do WebHint optou-se por basear toda a execução do método em uma abordagem orientada a tarefas. Desse modo, a análise de usabilidade de uma aplicação leva em conta a maneira com a qual os usuários realizam suas tarefas na interface da aplicação avaliada. Uma tarefa constitui-se em um conjunto de ações executadas pelos usuários visando alcançar determinado objetivo. Essas ações geram eventos na interface da aplicação que são registrados em arquivos de log e posteriormente analisados para se detectar problemas de usabilidade.

Desse modo, a primeira fase na execução do método é a definição das tarefas a serem analisadas. Na definição das tarefas optou-se por não realizar uma definição abstrata das tarefas, como acontece nas abordagens do WebRemUsine (Paganelli & Paternò, 2002) e WAUTER (Balbo et al., 2005). A definição abstrata das tarefas requer um passo adicional no processo de análise, que consiste em relacionar o modelo definido, em determinada

linguagem ou notação, com os eventos registrados no log. Por exemplo, no WebRemUsine é necessário relacionar a tarefa definida utilizando-se a linguagem *Concurrent Task Trees,* com os eventos no log (Paternò et al, 1997). Esse relacionamento do modelo abstrato com os eventos registrados no log demanda trabalho e tempo, encarecendo a etapa de definição das tarefas.

Assim, no WebHint optou-se por definir as tarefas sem o uso de notações. Para definir uma tarefa no WebHint, é necessário apenas executar a tarefa na interface da aplicação, enquanto os eventos ocorridos são capturados e registrados em um log. Assim, os eventos capturados definem a sequência de eventos esperada para a realização da tarefa. Esta sequência "ideal" definida é utilizada nas etapas posteriores da análise. Desse modo, evita-se o custo de se definir as tarefas de maneira abstrata, tendo-se ainda a necessidade adicional de se realizar o relacionamento entre tarefas definidas e eventos no log.

3.2 Captura da Interação

A segunda fase na execução do método consiste na captura da interação dos usuários com a aplicação avaliada. No WebHint, optou-se por realizar a captura da interação dos usuários utilizando-se uma abordagem de *proxy* remoto.

A abordagem de *proxy* remoto, conforme já mencionado, permite realizar a captura da interação dos usuários de forma remota e automática, sem a necessidade de se instalar software nos computadores dos usuários ou nos servidores da aplicação analisada. Também não se tem a necessidade de alterar manualmente o código fonte da aplicação no servidor. A captura da interação é realizada utilizando-se um servidor *proxy* que fica entre os usuários e a aplicação. Desse modo, os usuários acessam a aplicação via *proxy* de

modo transparente, isto é, não há nenhuma alteração ou mudança na forma como utilizam a aplicação.

Assim, a interação é captura em campo, pois os usuários acessam a aplicação em seu ambiente cotidiano, em uma situação real de uso. Além disso, como a captura é realizada de maneira remota e automática, esta pode ser realizada em grande escala de usuários, ou seja, é possível monitorar a interação de todos os usuários acessando a aplicação.

Esta abordagem, diferentemente da utilização de logs de servidor Web, permite uma captura detalhada das ações dos usuários como cliques, movimentos de mouse e todas as ações executadas pelos usuários em uma aplicação. Além disso, diferentemente das abordagens baseadas na captura no computador do usuário, esta abordagem não necessita da instalação de software nos computadores dos usuários, o que favorece sua utilização em larga escala.
Nos estudos descritos no capítulo 4, nos quais utilizou-se o WebHint para realizar a avaliação de usabilidade de algumas aplicações Web, utilizou-se a ferramenta UsaProxy (Atterer, 2006) para realizar a captura da interação dos usuários. O UsaProxy implementa a abordagem de *proxy* remoto, que possui as vantagens descritas anteriormente.

3.3 Análise dos Dados

Tendo-se a interação dos usuários capturada em logs, a fase seguinte constitui-se na análise desses dados. No WebHint, a análise dos dados é composta por três principais atividades, a extração das tarefas do log, a identificação do modelo que representa a execução real das tarefas pelos usuários e a comparação entre os modelos real e ideal de execução das tarefas.

3.3.1 Extração de tarefas

Para extração de tarefas do log, definiu-se uma combinação de técnicas de transformação e de comparação de sequências a ser utilizada, conforme descreve-se a seguir.

O primeiro passo na extração das tarefas é a limpeza ou pré-processamento do log, isto é, utilizando-se a técnica de seleção, os eventos irrelevantes devem ser retirados do log para facilitar a análise posterior, evidenciando-se os eventos de interesse. Por exemplo, nas execuções do método relatadas nos estudos de usabilidade descritos no Capítulo 4, os movimentos de mouse e a digitação de caracteres foram retirados do log, dando-se ênfase a outros eventos como cliques e seleções de mouse.

Após a limpeza do log, é necessário convertê-lo para o padrão utilizado nos demais passos da análise, utilizando-se as técnicas de abstração e recodificação. Nos estudos descritos no Capítulo 4, houve a necessidade de primeiramente dividir o log, visando lidar com o entrelaçamento de eventos de diversos usuários, registrados sequencialmente no log. Assim, foi possível reconstruir a sequência de eventos que cada usuário executou em sua interação com a aplicação.

Após a limpeza, é realizada a extração das tarefas executadas pelos usuários, isto é, seleciona-se os trechos do log que representam a execução de cada tarefa a ser analisada. Para esta atividade utiliza-se a técnica de comparação de sequências LCS *(Longest Common Subsequence)*, que encontra a maior subsequência comum entre duas *strings*.

Dentre as técnicas existentes para análise de sequências, descritas no Capítulo 2 desta tese, LCS é a técnica mais adequada ao tipo de sequências que se obtém em logs de

interação de usuários. Conforme já mencionado, logs de eventos obtidos na interação dos usuários podem conter eventos que não correspondem a eventos presentes na execução ideal de certa tarefa. Por exemplo, quando o usuário, mesmo executando a tarefa com sucesso, tem desvios da sequência esperada, realizando diversas ações não previstas na execução tarefa. A técnica de LCS é a mais indicada para tratar este tipo de comportamento, pois se concentra nos eventos coincidentes entre as duas sequências.

Assim, tendo-se o objetivo de buscar no log intervalos similares à sequência "ideal" da tarefa, é possível detectar sequências similares, mesmo que a execução real da tarefa contenha eventos não previstos ou esteja incompleta. Desse modo, se o intervalo de log analisado possuir certo grau de similaridade com a sequência de eventos ideal da tarefa, identifica-se o trecho como uma execução da referida tarefa.
Nos estudos descritos no Capítulo 4, foram desenvolvidos algoritmos na linguagem de programação Java para realizar a limpeza e extração das tarefas do log.

3.3.2 Identificação do Modelo de Execução da Tarefa

Após a extração das sequências que representam a execução das tarefas pelos usuários no log, é necessário identificar os padrões de interação existentes na execução dessas tarefas.

No WebHint, selecionou-se *process mining* como a abordagem a ser utilizada na identificação dos padrões de execução das tarefas pelos usuários. Conforme descrito no Capítulo 2 desta tese, uma tarefa executada por um usuário na interface de uma aplicação Web pode ser representada como um modelo de processo em *process mining*. Os eventos que compõem a execução de uma tarefa podem ser representados como atividades em um

processo, pois estas possuem uma característica sequencial de execução, da mesma forma que atividades em um processo.

Desse modo, utilizando-se a técnica de **descoberta**, em um **perspectiva de controle de fluxo,** é possível extrair o modelo da tarefa de um log de eventos que contém a execução da tarefa pelos usuários. Esse modelo representa o padrão predominante de execução da tarefa, com base nas execuções reais da tarefa registradas no log.

Assim, é possível comparar o modelo extraído da execução real da tarefa com o modelo que representa o comportamento esperado para a execução da tarefa. Se o modelo real corresponde ao modelo esperado, pode-se validar o design da tarefa. Caso existam diferenças entre o modelo esperado e o modelo real, pode-se investigar, na interface da aplicação, quais as causas do comportamento não esperado, visando-se descobrir os problemas de usabilidade relacionados.

Na aplicação do método, nos estudos de usabilidade descritos no Capítulo 4, utilizou-se a ferramenta ProM (Dongen et al, 2005), que permite a manipulação e análise de logs através da aplicação de diversos algoritmos de *process mining*. Para ser analisado utilizando-se o ProM, o log foi convertido para se adequar ao formato utilizado pela ferramenta através de algoritmos implementados na linguagem Java.

A ferramenta ProM foi construída utilizando o conceito de *plug-ins,* que podem ser desenvolvidos, incluídos e utilizados na ferramenta. O *plug-in* Fuzzy Miner (Gunter & Aalst, 2007) foi utilizado para a obtenção do modelo de execução das tarefas. Tanto o modelo esperado, quanto o modelo real de execução foram gerados a partir das informações constantes nos logs de eventos.

O Fuzzy Miner implementa a técnica de descoberta em um perspectiva de controle de fluxo, isto é, ele constrói automaticamente o modelo de execução do processo utilizando apenas as informações constantes no log, conforme ilustrado na Figura 10.

Nos estudos realizados, em determinados casos, antes de se utilizar o Fuzzy Miner, foi utilizado o *plugin Sequence Clustering* (Veiga & Ferreira, 2010), para realizar o agrupamento de sequências levando em conta a similaridade entre elas. Esta etapa foi importante em tarefas que possuíam diversos padrões de execução. Assim, agrupando sequências com maior similaridade entre si foi possível obter uma melhor visualização dos diferentes padrões.

3.3.3 Comparação entre o Modelo Ideal e o Modelo Real

Tendo-se extraído o modelo de execução de cada tarefa do log, o passo seguinte é a comparação desse modelo obtido com o modelo esperado de execução da tarefa. Da mesma forma, *process mining* é utilizado na obtenção automática do modelo "ideal" utilizando-se as sequências capturadas na fase de definição das tarefas do WebHint.

Para a comparação entre o modelo real e o modelo ideal de execução de uma tarefa optou-se por não utilizar nenhum tipo de comparação automática. A comparação fica a cargo do avaliador, que, obtendo o modelo ideal e o modelo real de execução de cada tarefa, pode fazer uma comparação visual entre os dois modelos, verificando se há diferenças entre a execução esperada e a execução real da tarefa.

A ferramenta ProM permite a utilização de algoritmos para a aplicação da técnica de *process mining* conhecida como checagem de conformidade, que poderia ser utilizada

para comparar o modelo ideal com as informações constantes no log. No entanto, os algoritmos para a checagem de conformidade não apresentaram um bom desempenho nos logs analisados. Conforme relatado anteriormente, na execução de uma tarefa na interface de uma aplicação, alguns eventos executados pelos usuários e registrados no log não fazem parte da sequência "ideal" de execução da tarefa. Por exemplo, alguns usuários costumam selecionar o texto, quando fazem uma leitura, ou costumam clicar com o mouse na janela. Esses eventos, apesar de não alterarem a execução da tarefa, geram inconsistências quando as sequências são analisadas com os algoritmos de checagem automática de conformidade. Desse modo, mesmo tarefas executadas da mesma maneira, se desconsiderados os referidos eventos, são consideradas distintas pelos algoritmos de checagem de conformidade.

As técnicas de comparação, como o alinhamento de sequências, também foram analisadas visando-se sua utilização nesta fase da análise. No entanto, da mesma forma que os algoritmos de *process mining*, a comparação automática de sequências também não apresenta bons resultados quando se trata de logs de eventos obtidos na interação de usuários, devido à diferença de relevância entre os eventos em tarefas diferentes.

Dessa maneira, mesmo executando a "limpeza" do log, determinados eventos, conforme exemplificado anteriormente, não podem ser retirados do log de forma generalizada e automática, pois podem não ser relevantes em uma determinada tarefa, mas em outras podem ser importantes.

Assim, para a utilização de uma comparação automática que gerasse bons resultados seria necessário um pré-processamento do log feito de maneira específica para cada tarefa, analisando-se cada interação de cada usuário, inviabilizando a utilização da

técnica. Desse modo, optou-se pela comparação visual dos modelos pelo avaliador como a alternativa mais simples e adequada para esta atividade no WebHint. A Figura 10, ilustra como a comparação entre os modelos “ideal“ e “real“ é realizada. O modelo apresentado na Figura 10, à esquerda, demonstra o modelo esperado para a execução da tarefa. O modelo apresentado na Figura 10, à direita, apresenta o modelo obtido a partir dos logs de interação de usuários.

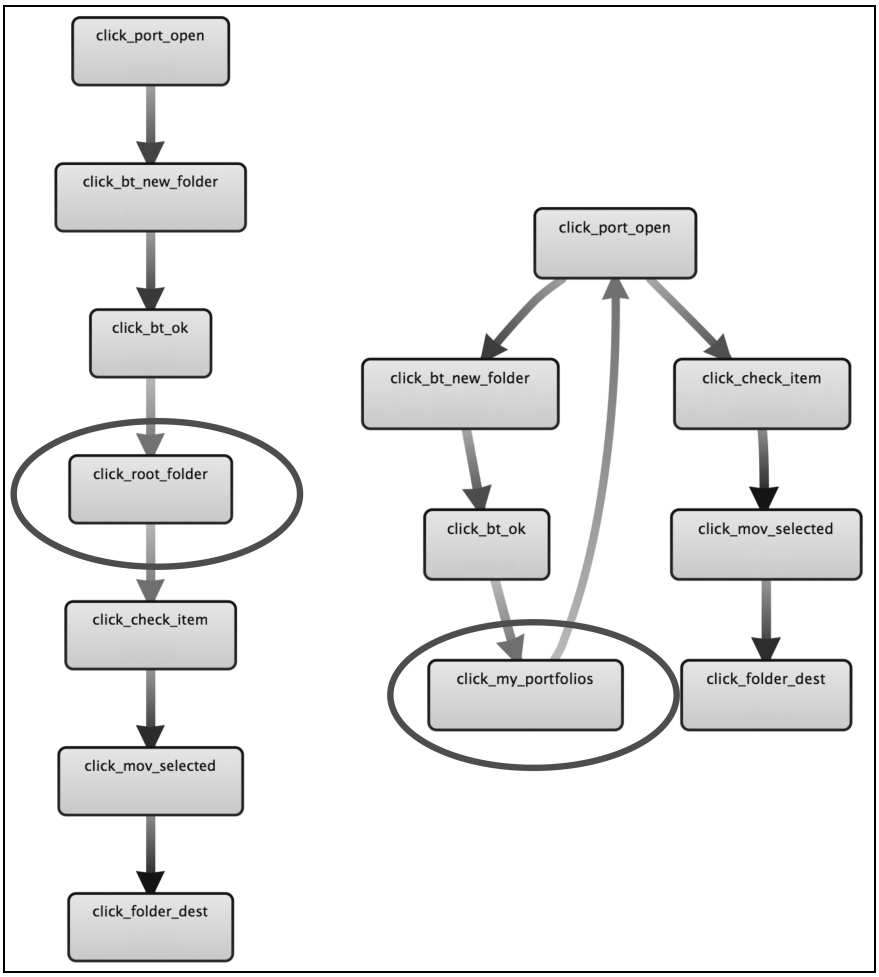

Figura 10. Exemplo de comparação entre o modelo esperado para a execução de uma tarefa e o modelo real obtido a partir do log.

O exemplo ilustrado na Figura 10, apresenta um dos problemas de usabilidade encontrados em um dos estudos realizados utilizando-se o WebHint e descrito no Capítulo 4. Na Figura 10, é possível perceber que o modelo esperado para a execução da tarefa difere do modelo encontrado. Levando em conta essas diferenças, a interface da aplicação foi analisada e o problema de usabilidade encontrado. Descobriu-se que a diferença entre o modelo esperado e o modelo encontrado devia-se a um problema de visibilidade em um dos elementos da interface que induzia os usuários a executarem a tarefa utilizando um caminho mais longo do que o esperado.

Levando em conta as escolhas realizadas na composição do método, a abordagem definida para análise dos dados no WebHint tem algumas vantagens em relação às abordagens existentes e descritas no Capítulo 2 desta tese.

O WebRemUsine (Paganelli & Paternò, 2002), o AWUSA (Tiedtke et. al 2002), o WAUTER (Balbo et al., 2005) e o WUP (Carta et al., 2011) baseiam sua análise na comparação automática entre duas sequências: a sequência esperada para a execução de uma tarefa e a sequência de eventos que um usuário executou. Estas abordagens apenas permitem uma análise individualizada, ou seja, analisam o comportamento de um único usuário, sem levar em conta os padrões de interação apresentados pelos usuários como um todo. O WebHint, por sua vez, visa a análise baseada no modelo de execução real da tarefa, levando em conta a interação de todos os usuários. Além disso, as abordagens WebRemUsine, AWUSA e WAUTER tem os mesmos problemas relatados anteriormente (seção 2.3.2.4) em se realizar uma comparação automática de sequências de eventos de usuário.

O WebQuilt (Hong et al., 2001) encontra os caminhos mais comuns de navegação entre as páginas de uma aplicação. Comparando-se a ele, o WebHint possui a vantagem de, além da navegação entre as páginas, levar em conta as ações executadas também dentro das páginas, o que possibilita uma análise mais detalhada das tarefas. Da mesma forma, o WebHint também tem essa vantagem em relação ao WELFIT (Santana & Baranauskas, 2008) e o Experiscope (Guimbretière et al., 2007), que baseiam sua análise na sumarização dos eventos que ocorrem em uma determinada página, não analisando tarefas que envolvem a navegação entre diferentes páginas da aplicação.

A seguir, o Capítulo 4 é composto por uma coletânea de 3 artigos publicados durante o desenvolvimento desta tese e que descrevem 3 estudos de usabilidade realizados utilizando-se o WebHint para realizar a avaliação de usabilidade de aplicações Web. Os estudos de usabilidade tiveram como objetivo a validação do método como uma alternativa para se avaliar a usabilidade de aplicações Web através da captura e análise automatizada da interação de usuários com as aplicações.

Capítulo 4

Coletânea de Artigos

Este capítulo é composto por três artigos, apresentados integralmente em seu conteúdo e na forma com a qual foram originalmente publicados.

O primeiro artigo que compõe este capítulo, *"Usability Evaluation of Online Applications Based on User Interaction Log Analysis",* apresenta o primeiro estudo de usabilidade realizado com o método WebHint. O estudo foi realizado analisando-se a usabilidade do ambiente de educação à distância TelEduc e teve o objetivo de se constituir em um teste piloto para validação do método. Nas sessões iniciais deste artigo, o método WebHint é também brevemente descrito. Esta descrição do método no artigo deve-se à característica introdutória do trabalho na apresentação do método. Esse artigo foi publicado no ICCEE 2010 - *3rd International Conference on Computer and Electrical Engineering*, em novembro de 2010, na cidade de Chengdu, China.

O segundo artigo que compõe este capítulo, *"Analyzing User Interaction Logs to Detect Usability Problems in Web Applications",* descreve o desenvolvimento e resultados de um estudo de usabilidade realizado com 4 diferentes aplicações Web, utilizando o WebHint. O objetivo desse estudo foi verificar o potencial de utilização do WebHint em diferentes aplicações Web. Este artigo foi publicado no SWS 2011 - *3rd Symposium on Web Society,* em outubro de 2011, na cidade de Porto Elizabete, África do Sul.

O terceiro artigo que compõe este capítulo, "*Discovering and Analyzing Patterns of Usage to Detect Usability Problems in Web Applications",* descreve a realização e os resultados de um estudo de usabilidade desenvolvido em uma versão atualizada da mesma aplicação utilizada no primeiro estudo realizado com o WebHint. Esse estudo teve o objetivo de validar o WebHint como uma alternativa para avaliações sucessivas de usabilidade de aplicações Web. O artigo foi publicado no ISDA 2011 - *11th International Conference on Intelligent Systems Design and Applications,* em novembro de 2011, na cidade de Córdoba, Espanha.

Usability Evaluation of Online Applications Based on User Interaction Log Analysis

Ariel Vargas[1,2], Harold Weffers[2], Heloísa Vieira da Rocha[1]

[1]Institute of Computing (IC)
University of Campinas (UNICAMP)
Campinas, Brazil
e-mail: {vargas, heloisa}@ic.unicamp.br

[2]Laboratory for Quality Software (LaQuSo)
Eindhoven University of Technology (TU/e)
Eindhoven, The Netherlands
e-mail: {a.vargas, h.t.g.weffers}@tue.nl

***Abstract* — In this paper we describe the WebHint method for usability evaluation of online applications. The method is based on remote and automatic capture, and semi-automatic analysis of users interaction. We also describe the development and the results of a usability study carried out with WebHint, in order to verify the potential of the method as an alternative to analyze the usability of web applications.**

Keywords; User interaction analysis; usability evaluation; user activity tracking; remote capture; log file analysis; semi-automatic analysis; user behavior; user experience.

I. INTRODUCTION

Knowing the behavior of users in online applications is an important tool for online marketing. Online stores usually want to obtain information about the behavior of their costumers in order to determine their profiles. With knowledge about the profile of their costumers, the companies can offer personalized services and products for each costumer. However, information about users behavior interacting with the interface of online applications is not only interesting for marketing purposes. Usability analysts can also use this knowledge to evaluate the usability of web applications. Knowing how users perform their tasks in the interface of an online application can help usability analysts to discover usability problems in the application. This information can also be used to validate design decisions, verifying if users perform their tasks in the application as planned by the designers. It is also possible to verify if users experience some failures, abandon a task during the execution, or take a particular path to achieve a certain goal.

In order to evaluate the usability of online applications through the analysis of user behavior we developed the WebHint method [17]. Our method is based on remote and automatic capture and semi-automatic analysis of the interaction of users in online applications [2]. This way, users interact with the application in their usual work environment, while their interaction is automatically captured by software. Different from a traditional user test, there is no usability analysts observing users during the interaction. A traditional user test is an efficient method to find usability problems [10]. However, it is expensive due to the costs of finding users, moving them to the test laboratory, preparing the infrastructure, carrying out the test, collecting and analyzing the results [9,4,8]. Using the WebHint is not necessary to move users to a usability laboratory. The data collection is performed remotely and automatically when users access the evaluated application. This way, the context of usage, which is difficult to simulate in a laboratory, is preserved. The data analysis process is semi-automatic, so the workload and time to analyze the data is proportionally lower than in traditional user tests. Thus, our method allows us to evaluate the interaction of a large amount of users, enabling a quantitative analysis of the usability of the application.

To begin with, this paper is structured as follows: in section II the WebHint method is described; section III describes some related works to the WebHint; section IV describes an study executed to validate the proposed method; in section V, the results of the study are presented; and in section VI, we present some concluding remarks about this work.

II. WEBHINT METHOD

The WebHint is a method for web usability evaluation, consisting of three stages of execution, as is described below and shown in Figure 1.

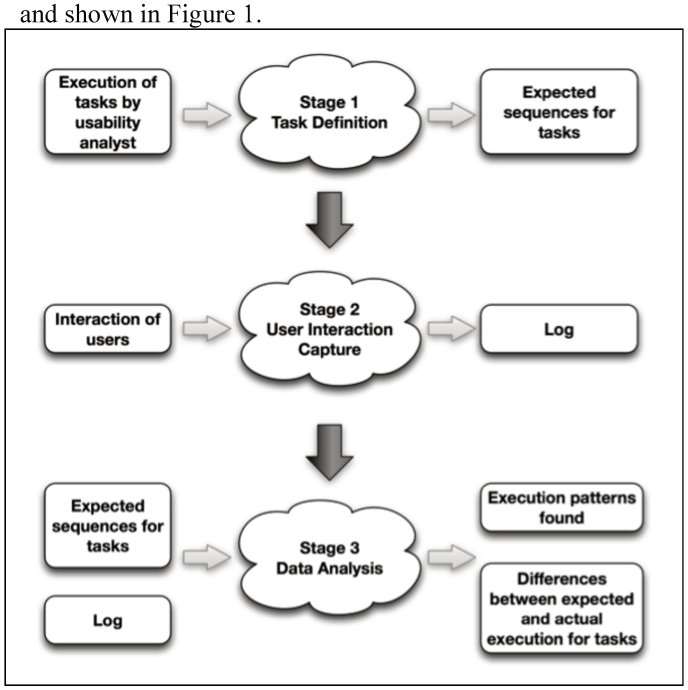

Figure 1. WebHint method

A. Stage 1 - Task Definition

The first stage in WebHint method is the definition of the tasks to be analyzed in the evaluation. A task consists of a sequence of actions performed by users in the application's interface in order to achieve a certain goal. For instance, the sequence of actions performed to send an e-mail message, in a webmail application, is a task.

The definition of tasks in WebHint consists of performing the expected sequence of actions for each task in the interface of the application, as was planned by the application designer. The sequence of actions performed is captured by software and saved in a log file. These captured sequences are subsequently used in the Data Analysis stage of the method.

B. Stage 2 – User Interaction Capture

In the second stage the interaction of users with the application's interface is monitored. All actions performed by all users in the application are captured: mouse movements, keystrokes, links accessed, pages loaded, etc. These data are captured automatically by software and saved in a log file in order to be analyzed in the Data Analysis stage.

The data capture is performed using a remote proxy approach [3]. A proxy server, placed between the application and the users, works intercepting page requests and dynamically embedding programming code in each page to register the browser events triggered by users. Thus, the capture process is transparent for users and allows a detailed data capture of the interaction. Furthermore, using a proxy approach it is not necessary to install software in the server of the application or in the computer of users to execute the data capture. This way, it is possible to preserve the context of usage of the applications and capture the interactions of all the users accessing the application.

C. Stage 3 – Data Analysis

In the Data Analysis stage all information obtained in the previous stages is analyzed. Basically it is composed of two kinds of data: 1 - the sequences of actions that represent the expected behavior for the tasks to be analyzed; 2 - the interaction of all users, captured in the interface of the application evaluated. To analyze these data a sequence of activities is performed:

1) Task sequence extraction

The first activity in the Data Analysis is the extraction of the tasks performed by users from the log, i.e., the identification of segments that represent the execution of a certain tasks in the log file. For this purpose the following data manipulation is executed:

- Log cleaning: the log with all interaction of all users is processed in order to eliminate events not relevant for the analysis. For instance, the mouse movements can be no relevant in one analysis, representing noise in the data. Removing these events from the log allows us to highlight events of interest, as mouse clicks, for instance.
- Tasks extraction: after the cleaning, the log is processed in order to identify the fragments containing sequences of actions that represented the execution of tasks. Using sequence similarity identification LCS (Longest Common Subsequence) [13] the task sequences are extracted from the log.

2) Patterns identification

In the second activity of the Data Analysis stage, the extracted sequences are analyzed in order to identify common patterns of execution for each task. This is done using Process Mining techniques to automatically extract the process models for each analyzed task [1]. This way, it is possible to obtain a visual model that represents the real execution of each task.

3) Sequence Comparison

In the third activity, the process model that represents the real execution of each task (extracted from the log) is compared to the model that represents the expected execution of each task (automatically extracted from the sequences obtained in the Task Definition stage).

Looking for the differences between the "expected" and the "actual" execution of the tasks, it is possible to identify if the tasks are performed as expected or not. If users performed a task different from the expected, the execution pattern found is investigated in the interface of the application, in order to discover the cause of the unexpected behavior and detect the usability problems related.

III. Related Work

The works previously developed in order to support remote and semi-automatic usability evaluation of web applications can be classified in two groups. In the first group are the tools for performing the capture of the interaction of users in web applications as: WebVip [15], WET [7], TEA [11] and UsaProxy [3]. In the second group, are the tools that perform the capture and support the analysis of the data captured as: WebRemUsine [12], AWUSA [16], WAUTER [5], WELFIT [14] and WebQuilt [8]. Bellow, we describe the relation of WebHint to the previous developed works.

A. Task Definition

In stage 1 of WebHint, we simplified the definition of tasks avoiding the use of abstract models to represent the tasks to be analyzed. Different from WebRemUsine and WAUTER using abstract task definition, our method just requires a simple execution of the task in the application's interface to define it. It avoids an additional step in the task definition that consists of making the relation of the elements on the abstract task model to the events present in the log file. This activity can be extremely time-consuming, depending on the complexity of the tasks and the variety of events present in the log.

B. Data Capturing

In stage 2 of WebHint, we capture the interaction of users in the interface of the web applications using a proxy approach. This approach has some advantages over the others, as explained bellow.

AWUSA works with server logs having no detailed information about actions performed by users in the

application's interface such as, mouse clicks, movements, selected objects, etc. The data captured by WebVip and TEA are similar to web server logs, having the same drawbacks as AWUSA. Furthermore, TEA needs to be installed on the users' computer and WebVip needs manual insertion of programming code in each web page to enable the data capture. WET performs more detailed capture than WebVip. However, as WebVip, it also needs manual insertion of programming code in each web page to execute the capture. WebRemUsine and WELFIT capture all actions performed by users in the web application, however they also need manual edition of each web page of the application to execute the capture. WAUTER and WebQuilt capture user interaction in the client side, so they can also capture all actions performed by users. However, they need to install software on the users computer.

The proxy approach, different from the other approaches, performs detailed capture of users actions and has the advantage of working as a proxy between server and client. So, it is not necessary manual insertion of a code on web pages or software installation on client or server computers. UsaProxy is a capturing tool that implements the proxy approach to perform the data capturing.

C. *Data Analysis*

In stage 3 of WebHint, all data captured in stage 2 are analyzed. Our method has some differences related to the analysis executed by other tools, as explained below. WebRemUsine compares the expected sequence of actions for a task to the sequence of actions performed by a user. As a result, the tool shows the differences between both sequences. WELFIT performs an analysis of all users actions on a single page but does not deal with the path followed between pages. AWUSA and WAUTER perform the analysis searching for sequences of actions previously related to usability problems.

WebQuilt analysis is more comprehensive than other approaches. It shows common navigation paths followed by users through the pages of the application.

Similar to WebQuilt, in our data analysis, we find common patterns of actions executed by users. However, our captured data are more accurate, including mouse movements, clicks and all actions performed by users in each page of the application, not only the sequence of pages requested. Therefore, our analysis is deeper and more comprehensive than WebQuilt.

Moreover, using a proxy approach we do not have the workload of manually editing each webpage in order to insert a code to capture user interaction as in WebVip, WET, WebRemUsine and WELFIT. It is also not necessary to install software on the client's computer as in WAUTER or WebQuilt.

Furthermore, in WebHint we can analyze all actions from all users interacting with the application. It allows us to make a quantitative analysis of the usability of the application. Different from WebRemUsine, just performing the comparison between two single sequences of actions, our analysis uses the common patterns found to compare to the expected sequences for each analyzed task. It gives us a comprehensive view of the users behavior.

IV. USABILITY STUDY DEVELOPMENT

In order to validate the WebHint method a pilot usability study was developed. In the study 52 users had their interaction monitored for a period of 2 months in TelEduc (version 4.1.1). TelEduc is an application developed to support on-line courses. In the study we evaluated the most frequently used tools of the application, Mail, Portfolio, Forum and Groups. The Mail works as an internal webmail in the application. The Portfolio works as a file repository where users can save texts, images and any kind of files. Users can share material in their portfolio and also access other portfolios making comments on shared material. In the Forum, users can basically read and post messages. The Groups tool allows users to create groups of participants for activities.

A simulated course was prepared in TelEduc for the study. The Mail, Portfolio, Forum and Groups were prepared with material and information in order to simulate the development of an online course. Some tasks usually performed using these tools were analyzed in the evaluation. The expected sequences of actions to execute each task were performed by a usability analyst in the interface of the application, in order to be captured and saved in a log file. These sequences were used in the analysis stage as the expected sequences for the analyzed tasks.

In order to obtain participants for the study an invitation was sent by email to some academic groups and 52 users agreed to participate. These users received an email with the information they needed to access the application as the URL, login and password. They also received a list of activities they should perform in the course, simulating a piece of homework set by the professor. These activities represented the tasks to be analyzed. When the users accessed the application, their interaction was captured using the UsaProxy tool and saved in a log file for further analysis. The same tool was used to capture the sequences in the task definition.

The analysis of the data was executed following the activities described in the third stage of WebHint method. We implemented algorithms in Java to perform the log cleaning and to extract the tasks from the log. The process-mining tool ProM [1] [6] was used for discovering the patterns of usage of the analyzed tasks.

V. USABILITY STUDY RESULTS

As a result of the developed study we detected some usability problems in the interface of the evaluated application. The detected problems are described below.

A. Problem 1 - Portfolio tool – Comment

This problem was detected in the task that consists of accessing the portfolio of a certain user and posting a comment in one item. The expected sequence for this task is showed in Figure 2 – left. Analyzing the behavior of the users performing this task, the pattern illustrated in Figure 2-right was found. This pattern was found in 28% of the task execution and it differs from the expected user behavior for the task. It shows that the users, after posting the comment, went back to the Comment View page where the posted comments are visible. In order to discover why users executed this pattern, we followed the task in the interface of the Portfolio tool. Thus, it was possible to detect a lack of feedback in the interface. We noticed that users went back to the Comments View page in order to verify if their comment was successfully posted. The Portfolio tool does not inform if the comment was successfully saved or not. The tool just sends users back to the Item View page, without any message or information. It explains why users presented the referred behavior.

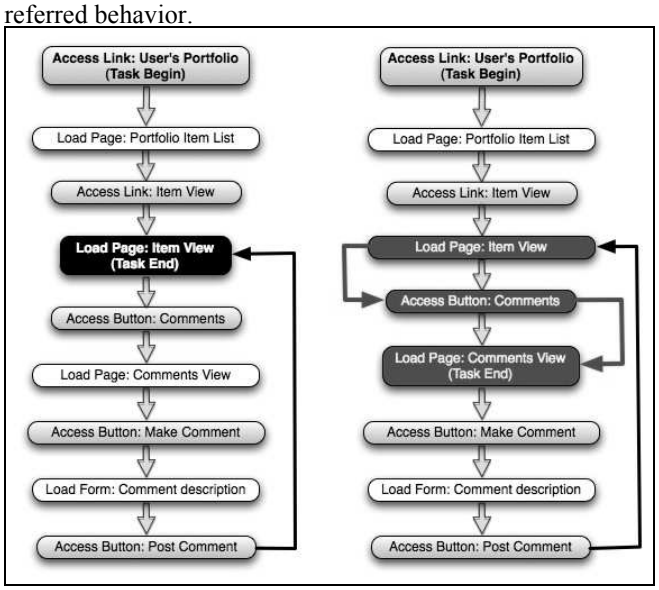

Figure 2. Execution pattern expected (left) and execution pattern found (right).

B. Problem 2 - Mail tool – Message box

In the task consisting of creating and sending a message using the Mail tool, 61% of users performed a different pattern from the expected. The execution pattern found shows that the users clicked frequently on the message box before starting to write a message. Reproducing the sequence of actions to compose a message in the interface of the application, it was possible to indentify that, when users click on the message box, the cursor does not appear in the box, it disappears from the screen. This problem hinders users to realize that the focus of the application is already in the message box. It induces them to click repetitive times on the message box, trying to put the focus there.

C. Problem 3 - Mail tool – Reply message

Analyzing the task where users need to reply to an email message using the tool Mail it was possible to verify a pattern different from the expected. In this pattern, 30% of users had the following unexpected behavior at the end of the task: (1) pressing the button "send"; (2) selecting the receiver of the message; (3) pressing the button "send" again. In the expected sequence for this task, users should just press the "send" button once, to finish the task. Following the sequence of actions executed in the interface of the Mail tool, it was possible to discover that sometimes the interface does not set the receiver of the message by default, when the button "reply" is pressed. So, it is necessary to set the receiver of the message again.

D. Problem 4 – Forum – Messages visibility

Analyzing the task in which users need to answer a message posted in the Forum it was possible to observe that 41% of the users usually read several answers posted by other users, before answering the message posted. This unexpected behavior could have been caused because the users intended to read other answers before composing their own. However, analyzing the interface of the Forum tool it was possible to observe that 21% of the users posted their answers not below the referred question in the hierarchy of the forum. This way, it was possible to identify that some users had difficulty in finding the correct place to answer the message posted, indicating a visibility problem in the hierarchy of the forum.

E. Problem 5 - Portfolio tool – Sharing item

Analyzing the task in which users had to add a new item in their portfolio it was possible to discover that 22% of users forgot to share the added item with the teacher. It means that the teacher was not able to view the posted item to grade it. To detect this problem we looked for a wrong behavior previously known. A usual complaint among the teachers using the Portfolio is that the users frequently forget to share the posted items. Thus, we were interested in verify how many users presented the referred behavior performing the task.

VI. CONCLUDING REMARKS

The pilot study developed with WebHint had the goal of verifying the potential of the method to be used in usability evaluations of online applications. As described above, the method allowed us to find usability problems in the evaluated application, validating the method. Due to be a pilot test, the study was developed in a controlled context, where users executed a group of tasks in the application. This way, our goal was not to make statements about quantitative results obtained in the study. Nevertheless, the presented quantitative results are useful to instance the kind of information the method can provide in a quantitative usability analysis.

As shown previously, the method can be used in three different ways. In the first way, the method can be used to detect usability problems. Looking for differences between

the behaviors presented by users and the expected behavior for the analyzed tasks, we detected the problems 1, 2, 3 and 4, described above.

In the second way, the method can be used to validate the interaction design of the interface. In the study we could observe that users executed the tasks as planned by designers in the Groups tool, validating the interaction design of the interface.

In the third way, the method can be used to determine the severity of known problems in the interface, as described in problem 5. Knowing how many users a certain problem affects and in which frequency it hinders the interaction of the users, it is possible to determine its severity and consequently its priority to be fixed.

The described usability study and its results showed how the WebHint method can be used as an alternative to analyze the interaction of users, allowing a comprehensive view about the usability of the web applications.

Furthermore, as the data collection is performed remotely and automatically when users access the evaluated application, we can preserve the context of usage and obtain data from all users accessing the application. As the data analysis process is semi-automatic, the workload and time to analyze the data does not increase when the number of users involved increase. This way, we can evaluate the interaction of a large amount of users, enabling a quantitative analysis of the usability of the application.

REFERENCES

[1] van der Aalst, W.M.P., Weijters, A.J.M.M. 2004. Process mining: a research agenda. Computers in Industry 53, 231–244.

[2] Andreasen, M. S., Nielsen, H. V., Schrøder, S. O., and Stage, J. 2007. What happened to remote usability testing?: an empirical study of three methods. In Proc. of the SIGCHI - CHI '07. ACM, New York, NY, 1405-1414.

[3] Atterer, R. 2006. Logging Usage of AJAX Applications With the "UsaProxy" HTTP Proxy. In Proc. of the WWW 2006 Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection, Edinburgh, Scotland, May 2006.

[4] Baker, S.; Au, F.; Dobbie, G.; Warren, I. 2008. Automated Usability Testing Using HUI Analyzer. In ASWEC 2008, vol., no., pp.579-588, 26-28 March 2008

[5] Balbo, S, Goschnick, S, Tong, D, Paris, C. 2005. Leading Web Usability Evaluations to WAUTER. In The Eleventh Australasian World Wide Web Conference, Gold Coast, 2005.

[6] van Dongen, B.F., Alves de Medeiros, B.F., Verbeek, B.F., Weijters, A.J.M.M. and van der Aalst, W.M.P. 2005. The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo and P. Darondeau, editors, Application and Theory of Petri Nets 2005, volume 3536 of Lecture Notes in Computer Science, pages 444–454. Springer-Verlag, Berlin, 2005.

[7] Etgen, M. and Cantor, J. 1999. What does getting WET (web event-logging tool) mean for web usability. In Proc. of the Fifth Conference on Human Factors & the Web (Gaithersburg, MD, 1999).

[8] Hong, J, I., Heer, J., Waterson, S., Landay, J,A. 2001. WebQuilt: A proxy-based approach to remote web usability testing. ACM Trans. Inf. Syst. 19, 3 Jul. 2001.

[9] López J. M., Fajardo I., Abascal J.2007 Towards Remote Empirical Evaluation of Web Pages' Usability. In Jacko J. A.(Ed.): Human-Computer Interaction. Interaction Design and Usability. Part I. LNCS 4550

[10] Nielsen, J. Usability Engineering, Academic Press, Boston, MA, 1993.

[11] Obendorf, H., Weinreich, H., and Hass, T. 2004. Automatic support for web user studies with SCONE and TEA. In CHI '04. ACM, New York, NY, 1135-1138.

[12] Paganelli, L , Paternò, F. 2002. Intelligent analysis of user interactions with web applications. In Proc. of the 7th international conference on Intelligent user interfaces, 2002, San Francisco, California, USA

[13] Paterson, M., and Dancik, V. 1994. Longest Common Subsequences. In B. Rovan I. Privara and P. Ruzicka, editors, 19th MFCS'94, LNCS 841, pages 127-142, Kosice, Slovakia, August 1994. Springer Verlag.

[14] Santana, V. F., Baranauskas, M. C. C. 2008. A Prospect of Websites Evaluation Tools Based on Event Logs. In: HCIS 2008 Milan. Springer, 2008. v. 272. p. 99-104.

[15] Scholtz, J. AND Laskowski, S. 1998. Developing usability tools and techniques for designing and testing web sites. In Proc. of the Fourth Conference on Human Factors & the Web, Basking Ridge, NJ, Jun, 1998.

[16] Tiedtke, T., Märtin, C., Gerth, N. 2002. AWUSA – A Tool for Automated Website Analysis. In: Proc. of the 9th Int. Workshop DSV-IS 2002, Rostock, Germany, pp. 251–266, 2002.

[17] Vargas, A., Weffers, H. T. G., Rocha, H. V. A Method for Remote and Semi-Automatic Usability Evaluation of Web-based Applications Through Users Behavior Analysis. In: Proc. of the Measuring Behavior 2010 – 7th International Conference on Methods and Techniques in Behavioral Research, Eindhoven, The Netherlands, 2010.

Analyzing User Interaction Logs to Evaluate the Usability of Web Applications

Ariel Vargas
Institute of Computing (IC)
University of Campinas (UNICAMP)
Campinas, Brazil
e-mail:vargas@ic.unicamp.br

Harold Weffers
Laboratory for Quality Software (LaQuSo)
Eindhoven University of Technology (TU/e)
Eindhoven, The Netherlands
e-mail: h.t.g.weffers@tue.nl

Heloísa Vieira da Rocha
Institute of Computing (IC)
University of Campinas (UNICAMP)
Campinas, Brazil
e-mail:heloisa@ic.unicamp.br

Abstract

In this paper we describe a usability study carried out using the WebHint method to analyze the usability of four Web applications. The method is based on remote and automatic capture, and semi-automatic analysis of users interaction. The experiment and its results show how the method can be used for analyzing the behavior of users interacting with online applications, in order to obtain a comprehensive view about the usability of these applications.

1. Introduction

Observing the behavior of users interacting with a software application, in order to analyze its usability is usually carried out by methods as user test and field observation [8]. In spite of their popularity and efficiency, these methods are expensive and time consuming [7,3,6]. User test is considered an expensive method due to the costs of finding users to perform a test, moving them to the usability laboratory, preparing the infrastructure, carrying out the test, collecting and analyzing the results. Field observation is also an expensive method because it requires one or more usability analysts observing users interacting with the applications, in their work environment, for long periods of time. Observing users in their usual work environment, performing their usual tasks, preserves the context of usage, an important element in a usability analysis. However, the real context of usage is difficult to simulate in a user test carried out in a laboratory. If the test is carried out remotely, using video recording or videoconference, the context of usage can be preserved, avoiding also the costs of moving users to a usability laboratory. Nevertheless, analyzing video recording or observing users by videoconference are extremely time consuming activities. As same as user tests, video recording can be also used in field observation to avoid moving usability analysts to users' workplace. However, as same as user test, analyzing hours of video recording makes the data analysis expensive.

Due to these costs, it is common just to analyze the behavior of a few users in user tests or field observation. Analyzing the behavior of a few users limits the evaluation to a qualitative analysis. Furthermore, certain problems can just be highlighted in a quantitative analysis and also their impact can just be evaluated considering a large number of users [5].

In order to deal with the difficulties of observing users in field to analyze the usability of Web applications, the WebHint method was proposed [10] [11]. The method is based on remote and automatic capture and semi-automatic analysis of users interaction. This way, users interact with the application in their usual work environment, while their interaction is automatically captured by software. Different from a traditional user test, or field observation, there is no usability analyst observing users during their interaction. It is not necessary to move users to a usability laboratory. The data collection is done remotely and automatically when users access the evaluated application. This way, the context of usage is preserved. Process mining tools [1][4] support the data analysis, so the workload and time to analyze the data is proportionally lower than in traditional user tests. Due to this, the method allows the evaluation of a large number of users, enabling a quantitative analysis of the application's usability.

Thus, we developed a usability study carried out using the WebHint method to analyze the usability of four Web applications. The description of the study and its results are presented in this paper, structured as follows: in section II we describe the development of

the usability study; the section III describes the results of the study; and section IV presents some concluding remarks.

2. Usability study development

The developed usability study had the goal of analyzing the user interaction in some web applications using the WebHint method. We selected four applications to analyze: Ryanair (http://www.ryanair.com), Skyscanner (http://www.skyscanner.com), Journey Planner (http://9292.nl) and TU/e (http://www.tue.nl) websites. The Ryanair and the Skyscanner are websites for searching and booking flights. The Journey Planner is a website in which users search for advices for going from a certain place to another one in Netherlands, using the public transport. The last one is the website of the Eindhoven University of Technology (TU/e).

Nineteen users participated of the experiment. They were students from TU/e and from University of Campinas (UNICAMP), invited by email. In the study, we intended to analyze the usability of the applications using only the information provided by the logged interaction of the users. Thus, the participation of users was anonymous and we did not collect more information about the profile of the users.

The users received the instructions to participate of the study by email, including the tasks they should perform and the term of acceptance for participating. They agreed to have their interaction captured during the study, replying the email with their acceptance.

The usability study was carried out following the steps described in the WebHint method [10][11]. The method consists of three main activities: Task Definition, User Interaction Capture and Data Analysis, as described bellow.

2.1 Task Definition

In the Task Definition, the tasks to be analyzed in the evaluation are determined. A task consists of a sequence of actions performed by users in the application's interface in order to achieve a certain goal. The expected behavior for each task performed by users in the experiment was defined in this stage. Each task was performed on the interface of each application and the sequences of actions were captured and saved in log files. The sequences of actions for each task were used in the data analysis to be compared to the patterns of behavior found in the users interaction.

2.2 User Interaction Capture

In the User Interaction Capture, the interaction of users with the interface of the application was monitored. All actions performed by users were captured, as mouse movements, keystrokes, links accessed, pages loaded, etc. To capture the interaction of the users, a proxy server was placed between the users and the applications. This way, users accessed the applications through the proxy server that captured their interaction. The capturing tool UsaProxy [2] was used in the study due to its ability of capturing the Web browser events without need to install anything in the computer of the users or in the servers of the applications. This way, the data capturing was automatic and does not interfere on the interaction of users with the applications.

2.3 Data Analysis

In the Data Analysis all data obtained in the previous stages were analyzed. Basically they were composed of two kinds of data: 1 - the sequences of actions that represent the expected behavior for the tasks to be analyzed; 2 - the captured interaction of users.

The analysis was performed following the activities described in the WebHint method as detailed below.

2.3.1 Task sequence extraction: The first activity was the extraction of the tasks performed by users, i.e., the identification of segments that represented the execution of certain tasks in the log file. The following data manipulation was automatically executed using implemented algorithms:

- Log division: the log with the interaction of all users was divided in several log files containing the interaction of one single user per file. This was done in order to deal with interlaced events of different users, recorded sequentially in the log file.
- Executed tasks extraction: each log file was analyzed in order to determine the fragments containing sequences of actions that represented the execution of tasks. Using sequence similarity identification LCS (Longest Common Subsequence) [9] the task sequences were extracted from the log.

2.3.2 Patterns identification: In the second activity of the Data Analysis, the extracted sequences were analyzed in order to identify common patterns of execution for each task performed by users. The process mining framework ProM [1] [4] was used to find the patterns of execution for each analyzed task.

2.3.3 Sequence Comparison: In the third activity, the expected sequences for each analyzed task were compared to the patterns found. Looking for the differences between the "expected" and the "actual" execution of the tasks it was possible to identify if the

tasks were performed as expected or not. If users performed a task different from the expected, the execution pattern was followed in the interface of the application in order to discover the cause of the unexpected behavior.

3. Usability study results

Analyzing the behavior of users recorded in the logs we obtained the results presented below, organized by application.

3.1 Ryanair Website

In the Ryanair website, users had the following tasks to perform:

1. Searching for a flight from Eindhoven to Rome, going on the next Friday and returning on next Monday, considering the day of access;
2. Changing the dates of the previous search to 1 week later;

The analysis of the logs of users performing the tasks described above, allowed us to obtain the findings described bellow.

Finding 1 - Users forgot to accept the terms of use: Analyzing the task in which users should search for a flight it was observed a pattern of behavior different from the expected. The Figure 1, left, depicts the expected behavior for the task and the Figure 1, right, illustrates the pattern found and performed by 36% of users (7 in 19).

The pattern found shows that users forgot to check the box, illustrated in Figure 2, accepting the terms of use of the website. In the diagram (Figure 1 - right) it is possible to observe that users pressed the button Send before accepting the terms of use. Analyzing the interface of the application, we observed that the application only executes the search if the box is checked. The loop in the diagram (Figure 1 - right) shows that users needed to press the button Send again, after accepting the terms of use, to finally performing the search.

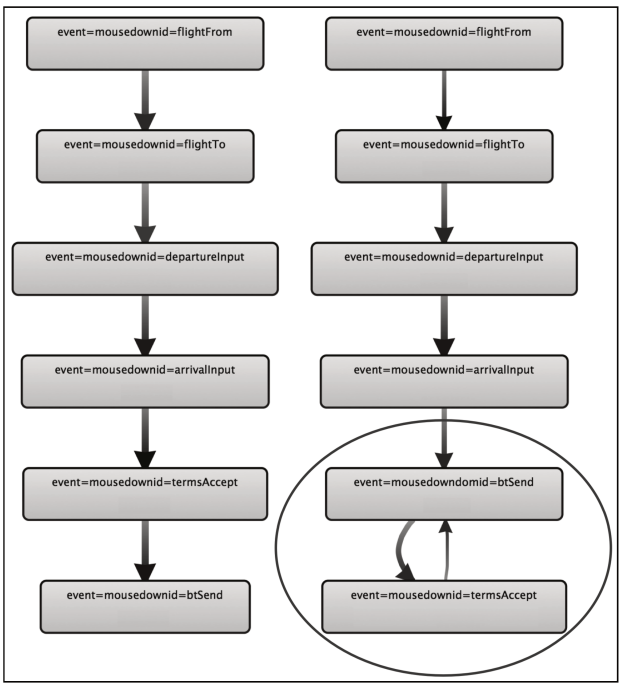

Figure 1. Expected behavior (left) and pattern of behavior found (right) for searching a flight.

Considering the behavior presented by users, it was possible to indentify that the interface has a usability problem related to the visibility of the checkbox. As it is mandatory to accept the terms of use to perform a search, the interface should make it clear, avoiding the users' mistake.

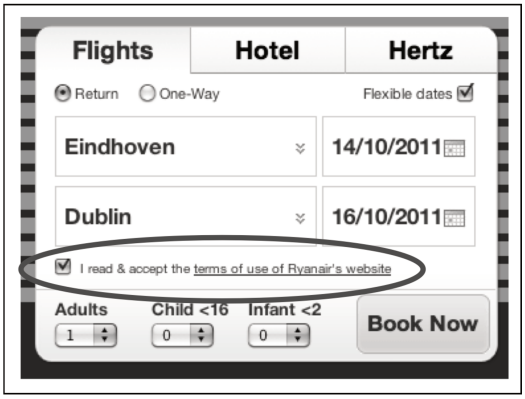

Figure 2. Terms of use of the website.

Finding 2 - Technical problem with the checkbox: In the analysis of the same task referred above, another behavior different from the expected was found and depicted in Figure 3. The figure shows that users checked the box accepting the terms of use of the website and after clicked on the button Send to perform the search, as expected. However, sequentially after pressing the button Send, they came back to the home

page, checking the box and pressing the button Send again, to perform the search.

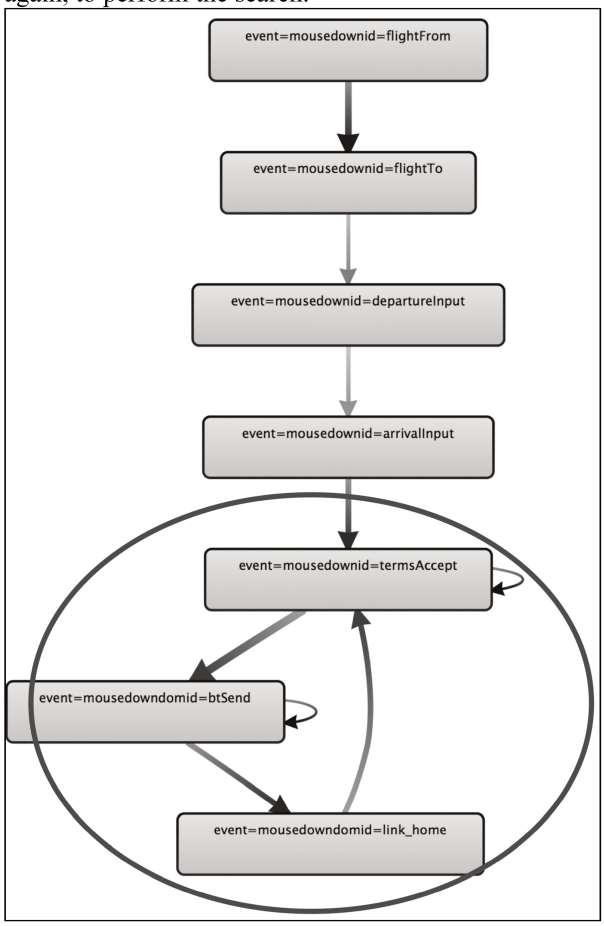

Figure 3. Pattern of behavior found.

Following the task on the interface of the application it was possible to notice that sometimes the application presents a technical problem. Even checking the terms of usage, sometimes, the application does not recognize it, sending the user to an error page, as illustrated in Figure 4.

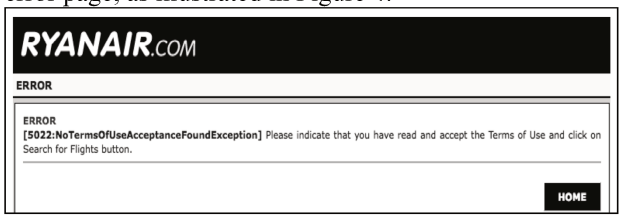

Figure 4. The cause of an unexpected behavior of users.

The referred problem explains the behavior of the users, indication a usability problem caused by a technical problem in the application. The problem affected 31% of the users (6 in 19) executing the task.

Finding 3 – Users did not use the navigation links to change the dates of a search: The task consisting of changing the dates of a search, previously executed, could be performed in two different ways: using the links (Figure 5) or making a new search.

Figure 5. Changing the dates using the links.

Analyzing the behavior of users, we discovered that none of them used the navigation links to perform the task, i.e., all users performed a new search. This behavior is consistent with the expected because the navigation links increase only one day each time. This way, making a new search is more efficient than clicking 14 times the links.

For performing a new search, users also had two different options: returning to the home page, option chosen by 40% of users (4 in 10); or using the button "New Search", option chosen by 60% of users (6 in 10). We did not find any pattern of interaction different from the expected for this task, validating the design of the task.

3.2 Skyscanner Website

In the Skyscanner website, users had the following tasks to perform:

1. Searching for a flight from Amsterdam to Paris, going on the first day of the next month and returning on the fifteenth day of the next month, considering the current day of access;
2. Changing the dates to return 2 days later;
3. Making new searches with other dates, origins and destinations;

The following results were obtained analyzing the log of interaction of users performing the tasks above.

Finding 4 – Most of the users executed a new search for changing the dates: Analyzing the task in which users searched for a flight we did not find any behavior different from the expected. All users performed the task as expected.

In the task consisting of changing the dates of a search, increasing 2 days on the date of returning, users had two options: using the arrow links (Figure 6 - a), or executing a new search.

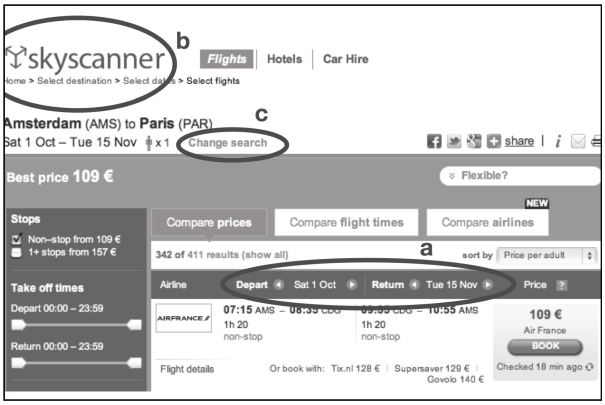

Figure 6. Arrows to increase/decrease dates (a), link to home page (b) and "Change Search" link (c).

The expected behavior for this task was users performing the task using the arrows (figure 6 - a). As the change is just increasing two days in the date of return, using the arrows is more efficient than making a new search. However, only 18% of users (3 in 16) used the arrows. The other 82% (13 users) performed a new search. Observing the interface of the application illustrated in Figure 6 (a), it is seen that the arrows have a small size, possible reason why most of the users did not notice the presence of the feature.

Finding 5 – Most of users chose the "Change Search" feature for new searches: After a first search, users had to perform other two searches changing origin, destination and dates. For performing this task, users had two different options: coming back to the home page (Figure 6 - b) or using the option "Change Search", depicted in Figure 6 - c.

Among the users performing the task, 40% (6 in 15) came back to the home page to execute the new search. The other 60% (9 users) performed the new search using the option "Change Search" that allows changing the search at the same page, without reload it. The behavior of users performing the task is consistent with the expected, validating the interaction design of the task.

3.3 Journey Planner Website

In the Journey Planner website users had the task of planning a trip by train from Eindhoven to Amsterdam, returning at the same day. Eleven users performed the task and 3 different patterns of behavior were found:

- 27% performed the task as expected;
- 27% presented a problem with the return trip;
- 45% experienced a technical problem with the application during their interaction;

Finding 6 - Users experienced a problem planning the return trip: The pattern found and illustrated in Figure 7 shows that users, after pressing the button for planning the return trip "lecTravelAdvice_cmdReturnTrip", did not change the parameters of the search, they simply pressed the button "lecRequest_cmdSubmit" to execute the search for the trip. They did not notice that the application does not invert the parameters of the search, making the previous origin the new destination and vice versa, as expected by them. In fact, the application just sends the users back to the search page with the original parameters. This way, when they executed the new search they just repeated the previous one. Thus, users needed to return again to the search page to invert the parameters and re-plan the trip back.

The usability problem is caused because the referred feature works as same as the "change search", confusing the users.

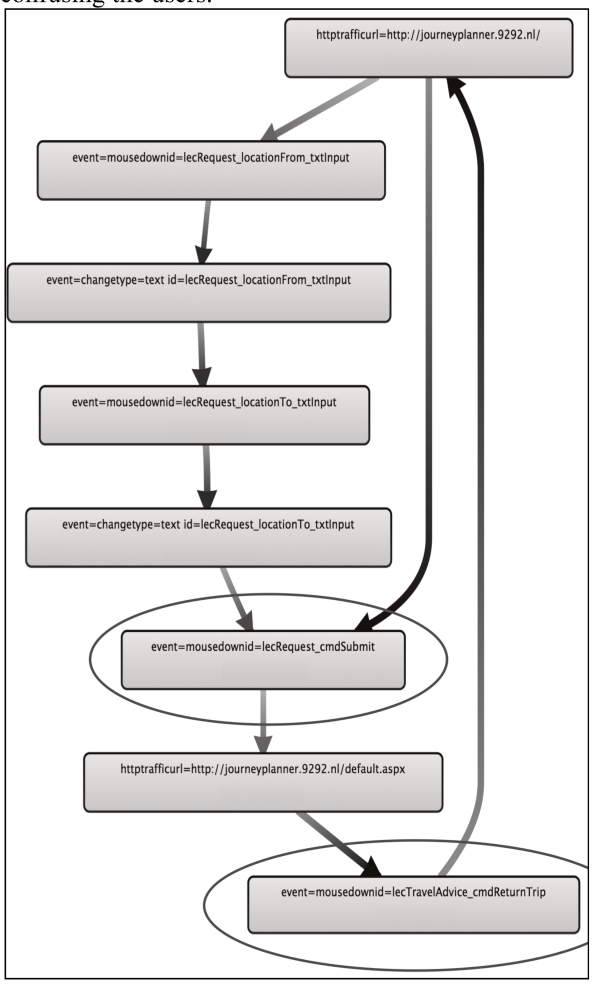

Figure 7. Users did not change the parameters when planned the trip back.

Finding 7 - Users experienced a technical problem: During the interaction with the application, 45% of the users experienced a technical problem. The application presented an "error message", sending users to an error page as illustrated in Figure 8. The diagram shows that

the users were sent to the error page "http://journeyplanner.9292.nl/error.aspx" when they were in the search page "http://journeyplanner.9292.nl/" or when they were in the results page "http://journeyplanner.9292.nl/default.aspx". This information is useful to guide the developers to investigate what caused this problem in order to fix it.

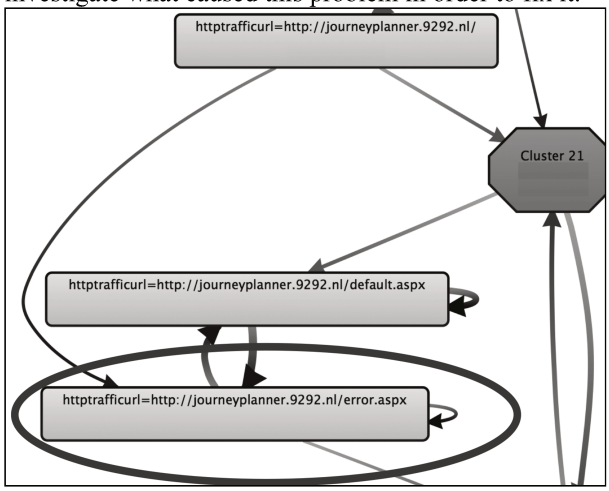

Figure 8. Users experienced a technical problem during their interaction.

3.4 TU/e website

In the TU/e website the task users should perform consisted of finding the page with information about the "menu of the day" in the restaurant of the auditorium of the university. This is a common task performed by students to select the restaurant of their preference.

Finding 8 - The search feature was the first attempt to look for the information: The results of the log analysis showed that none of the users completed the task, i.e., no one found the page with the information. It indicates that the structure of the information in the website is not clear, being difficult for users to find the information. Furthermore, the pattern of behavior found and illustrated in Figure 9, shows that the first option for 73% of users (11 in 15) was trying the search engine of the site.

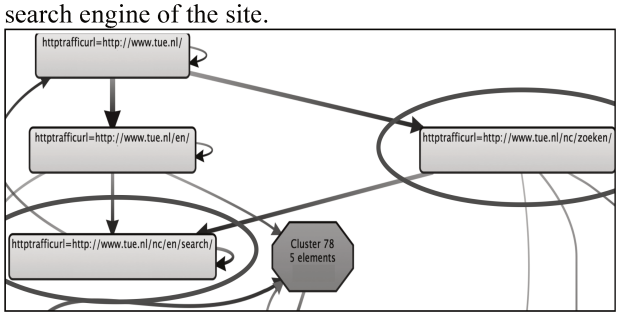

Figure 9. Pattern found showing the use of the search engine as the first attempt to find one information.

It demonstrates the importance of this feature and the need of enhancing it to improve the usability of the application.

4. Concluding remarks

The described study and its results show how the WebHint method can be used to analyze the interaction of users in Web applications. In the study, the interaction of users was captured remotely and automatically using a proxy server approach. Thus, there was no need of changing or installing software in the servers of the applications, or in the computers of the users. Users accessed the application in their usual work environment, maintaining the context of usage. It also avoided moving them to a usability laboratory. Furthermore, the analysis of the logs was performed using process mining, allowing us to deal with a large quantity of data without increasing the costs of the analysis.

Using the WebHint method, different from other methods as user tests of field observation, increasing the number of users involved in the analysis does not increase the time and workload of performing it. Due to the data capturing is automatic and the data analysis is supported by process mining tools, performing the analysis with 10 or 1000 users has the same cost.

Furthermore, the usability study also showed that comparing the expected behavior for the tasks with the patterns found in the users interaction logs, allows us to detect usability problems as described in findings 1 and 6. Besides helping to detect problems, the analysis showed also the number of users affected by the problems found. It helps to determine the severity of the problems, usually a difficult and important task in a usability evaluation.

Despite of usability problems, the analysis of the logs also revealed technical problems present in the applications, as exemplified in findings 2 and 7. The finding 7 shows a technical problem occurred in the server of the application. This kind of problem can be difficult to detect due to its casual characteristic. However, using the WebHint method, the interaction of users with the application can be monitored for days allowing the detection of this kind of problem.

Analyzing the behavior of users interacting with an application cannot only reveal problems, but it can also be useful to validate the interaction design of the interface and tasks. The findings 3, 4 and 5 are examples of that, where the behavior presented by users was consistent to the expected.

The usability analysis performed using the WebHint method can also be useful to analyze the trends of usage of features in an application, as exemplified in findings 3, 4, 5 and 8. The finding 8, for

instance, revealed that the search feature in the TU/e website was the first attempt when users look for an information, indicating its importance in the application.

In doing so, the described usability study and its results showed how WebHint method can be used to analyze the interaction of users, allowing a comprehensive view about the usability of the Web applications.

5. References

[1] van der Aalst, W.M.P., Weijters, A.J.M.M. 2004. Process mining: a research agenda. Computers in Industry 53, 231–244.

[2] Atterer, R. 2006. Logging Usage of AJAX Applications With the "UsaProxy" HTTP Proxy. In Proc. of the WWW 2006 Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection, Edinburgh, Scotland, May 2006.

[3] Baker, S.; Au, F.; Dobbie, G.; Warren, I. 2008. Automated Usability Testing Using HUI Analyzer. In ASWEC 2008, vol., no., pp.579-588, 26-28 March 2008

[4] van Dongen, B.F., Alves de Medeiros, B.F., Verbeek, B.F., Weijters, A.J.M.M. and van der Aalst, W.M.P. 2005. The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo and P. Darondeau, editors, Application and Theory of Petri Nets 2005, volume 3536 of Lecture Notes in Computer Science, pages 444–454. Springer-Verlag, Berlin, 2005.

[5] Etgen, M. and Cantor, J. 1999. What does getting WET (web event-logging tool) mean for web usability. In Proc. of the Fifth Conference on Human Factors & the Web (Gaithersburg, MD, 1999).

[6] Hong, J, I., Heer, J., Waterson, S., Landay, J,A. 2001. WebQuilt: A proxy-based approach to remote web usability testing. ACM Trans. Inf. Syst. 19, 3 Jul. 2001.

[7] López J. M., Fajardo I., Abascal J.2007 Towards Remote Empirical Evaluation of Web Pages' Usability. In Jacko J. A.(Ed.): Human-Computer Interaction. Interaction Design and Usability. Part I. LNCS 4550

[8] Nielsen, J. Usability Engineering, Academic Press, Boston, MA, 1993.

[9] Paterson, M., and Dancik, V. 1994. Longest Common Subsequences. In B. Rovan I. Privara and P. Ruzicka, editors, 19th MFCS'94, LNCS 841, pages 127-142, Kosice, Slovakia, August 1994. Springer Verlag.

[10] Vargas, A., Weffers, H. T. G., Rocha, H. V. A Method for Remote and Semi-Automatic Usability Evaluation of Web-based Applications Through Users Behavior Analysis. In: Proc. of the Measuring Behavior 2010 – 7th International Conference on Methods and Techniques in Behavioral Research, Eindhoven, The Netherlands, 2010.

[11] Vargas, A., Weffers, H. Rocha, H. V. Usability Analysis of User Interaction in Online Applications. In: Proceedings of ICCEE 2010 - 3rd International Conference on Computer and Electrical Engineering, Chengdu, China, 2010.

Discovering and Analyzing Patterns of Usage to Detect Usability Problems in Web Applications

Ariel Vargas[1,2], Harold Weffers[2], Heloísa Vieira da Rocha[1]

[1]Institute of Computing (IC)
University of Campinas (UNICAMP)
Campinas, Brazil
e-mail: {vargas, heloisa}@ic.unicamp.br

[2]Laboratory for Quality Software (LaQuSo)
Eindhoven University of Technology (TU/e)
Eindhoven, The Netherlands
e-mail: {a.vargas, h.t.g.weffers}@tue.nl

***Abstract*— In this paper we describe two usability studies in which the interaction of users with an online application was remotely and automatically captured and analyzed. The usability studies were performed using the WebHint method for usability analysis of web applications. We evaluate two different versions of the application in order to observe the applicability of the method for successive usability analysis. The results show how the WebHint method can be used as an alternative approach to carry out successive evaluations of the usability of an application in order to analyze the evolution of different versions of its interface.**

Keywords - usability evaluation, remote usability evaluation, log file analysis, usage mining, pattern mining, sequence mining, user behavior analysis.

I. INTRODUCTION

Methods for observing users interacting with a computer application as user test and field observation are largely used for detecting problems in the interface of computer applications [9]. In spite of their popularity and efficiency, these methods are expensive and time consuming [8][4][7]. Due to the costs, usually these methods are applied to analyze the behavior of a few users in usability evaluations. Analyzing the behavior of a few users hinders a quantitative analysis of the usability. In addition, to determine the actual impact of the problems found in an evaluation is important to know the number of users affected and how often these users are affected by the problems [6]. Another important element when the usability of an application is evaluated is its context of usage. The context is difficult to simulate in user tests carried out in laboratory. If the tests are carried out remotely, with users in their usual work environment, the context can be preserved. However, the number of users involved in the tests is still limited by the costs of executing and analyzing the data from the test. Field observation, in its turn, preserves the context of usage but due to its costs, also usually involve only a few users.

To deal with the difficulties mentioned above, the WebHint method was proposed in [12] for Web usability analysis. The method is based on remote and automatic capture of user events [2] and on the analysis of these events in order to discover patterns of usage. This way, users interact with the application in their usual work environment, while their interaction is automatically captured by software. The context of usage is preserved because the data collection is done remotely and automatically, when users access the evaluated application from their usual environment. As the data analysis is supported by process mining tools [1][5], the workload and time spent to analyze the data is proportionally lower than in user tests or field observation. In doing so, the method allows the analysis of the users' interaction in field and the analysis of large quantity of users.

This way, we develop two usability studies applying the WebHint method to analyze the interaction of users with TelEduc (http://www.teleduc.org.br), an e-learning Web application. The studies were carried out to evaluate two different versions of the application. Our goal was to observe the applicability of the method for successive usability analysis.

The description of the studies and results are presented in this paper, structured as follows: in section II we describe the development of the usability studies; the section III describes the results of the studies; and section IV presents some concluding remarks.

II. USABILITY STUDIES

Two usability studies were carried out to evaluate the usability of TelEduc, an e-learning application to support online courses. In the first usability study, the version 4.1.1 of TelEduc was evaluated and in the second study, the version 4.2.2 was evaluated.

In the studies we evaluated the most frequently used tools of TelEduc, the mail, the portfolio and the forum. The Mail works as an internal email tool in the application, in which users can send and receive messages. The portfolio works as a file repository, in which users can save texts, images and any kind of file. Users can share material in their portfolio and also access portfolios of other students, making comments on shared material. In the forum, users can read and post messages.

The participants of the studies were university students and professors, invited by email. These users received an email with the information they needed for accessing the application, as the URL, login and password. They also received a list of activities they should perform in the application.

In the first usability study we had 52 users participating as volunteers, having their interaction monitored when they accessed the application and executed the tasks. In the

second study, we had 17 users participating of the study as volunteers. The usability studies were carried out following the steps described in the WebHint method [12][13]. The method consists of three main activities: Task Definition, User Interaction Capture and Data Analysis, as described bellow.

1) Task definition

In the task definition, the tasks to be analyzed in the evaluation are determined. A task consists of a sequence of actions performed by users in the application's interface in order to achieve a certain goal. The expected behavior for each task performed by users was defined in this stage. Each task was performed in the interface of the application and the sequences of actions were captured and saved in log files. The sequences of actions for each task were used in the data analysis to be compared to the patterns of behavior found in users' interaction. The same set of tasks was used in both experiments and are briefly described bellow:

1. Including a new item in the portfolio, sharing it with the professor and associating it to an evaluation.
2. Making a comment in one item in the portfolio of a certain student.
3. Creating a new folder in the portfolio and moving a certain item into the folder.
4. Replying a certain message in the mail tool.
5. Replying a certain message to all senders in the mail tool.
6. Participating of the forum commenting a certain topic.

B. User interaction capture

In the user interaction capture, the interaction of users with the interface of the application was monitored. All actions performed by users were captured, as mouse movements, keystrokes, links accessed, pages loaded, etc. To capture the interaction of users, a proxy server was placed between users and the application. This way, users accessed the application through the proxy server that captured their interaction. The capturing tool UsaProxy [3] was used in the studies due to its ability of capturing the Web browser events without need to install anything in the computers of users or in the server of the application. This way, the data capturing was automatic and did not interfere on the interaction of users with the application.

C. Data analysis

The analysis of the data was executed following the activities described in the WebHint method [12][13].

1) Task sequence extraction

The first activity was the extraction of the tasks performed by users, i.e., the identification of segments that represented the execution of tasks in the log file. Implemented algorithms automatically executed the following data manipulation.

Log division: the log with the interaction of all users was divided in several log files, containing the streaming of events of one single user per file, in order to deal with interlaced events of different users, recorded sequentially in the log.

Executed tasks extraction: each log file was analyzed in order to determine the intervals representing the execution of tasks. Using sequence similarity identification LCS (Longest Common Subsequence) [11], the executed tasks were extracted from the log.

2) Patterns identification

In the second activity of the data analysis, the extracted task intervals were analyzed to identify common patterns. The process mining framework ProM [1] [5] was used to find the patterns of execution for each analyzed task. ProM allows the manipulation of event logs using several algorithms for process mining. In our analysis we used some algorithms for discovering the execution patterns for the tasks executed by users.

3) Sequence comparison

In the third activity, the expected sequences of events for each analyzed task were compared to the patterns found. Looking for the differences between the "expected" and the "actual" execution of the tasks it was possible to identify if the tasks were performed as expected or not. If users performed a task different from the expected, the execution pattern was followed in the interface of the application in order to discover the cause of the unexpected behavior.

III. USABILITY STUDIES RESULTS

As a result of the usability studies we discovered some usability problems in the interface of the application as described bellow.

A. Portfolio: comment posting

This problem was detected in the task that consists of making a comment in one item of the portfolio of a certain user. The expected sequence of actions for this task is showed in Figure 1 - left.

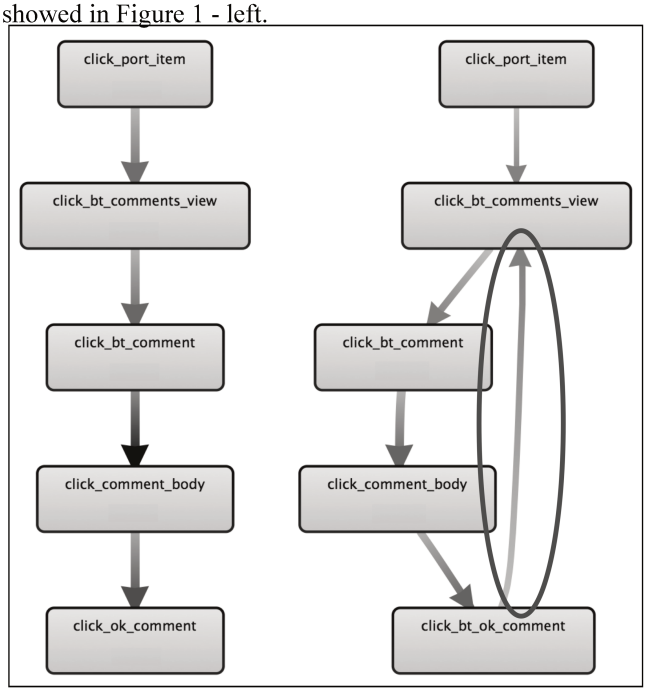

Figure 1. Expected behavior for the task (left) and pattern found (right) - users returning to the Comments View page.

Analyzing the behavior of users performing this task, the pattern illustrated in Figure 1- right was found and differs from the expected behavior for the task. It shows that users, after posting a comment, returned to the Comments View page, where the posted comments are visible. In order to discover why users executed this pattern, we followed the task in the interface of the application. Thus, it was possible to detect a lack of feedback in the interface. We noticed that users returned to the Comments View page in order to assure that their comments were posted. It happened because the portfolio does not inform if the comment is successfully posted or not. The tool just sends users back to the Item View page, without any message or information.

This problem was found in both of usability studies. In the first study, 28% of users (14 in 52) presented the referred behavior and in the second study, 50% of users (7 in 14) presented it. This way, it is possible to observe that the usability problem persists in the new version of the application.

B. Portfolio: sharing item

Analyzing the task in which users had to add a new item in their portfolio, it was possible to observe that part of users forgot to share the added item with the professor. A usual complaint among the professors using the portfolio is that students frequently forget to share the posted items. If the students do not share a posted item, the professor is not able to view and evaluate it.

The problem was verified in the first usability study and also in the second one. In the first study, 22% of users (12 in 52) presented the referred behavior, forgetting to share the item. In the second study, 54% (6 in 11) of users presented the referred behavior, confirming the persistence of the problem in the new version of the tool.

C. Portfolio: adding folder

The expected behavior for the task in which users had to create a folder in their portfolio and move a certain item into the folder is illustrated in Figure 2 – left. The diagram shows that after creating the folder users should access the link to return to the root folder of the portfolio (click_root_folder). Sequentially, they should select the item (click_check_item), press the button to move the selected item (click_move_selected) and finally select the folder where the item should be moved (click_folder_dest). However, 77% of users executing this task (10 in 13) presented a behavior different from the expected, as depicted in Figure 2 – right. The diagram, on the right, shows that instead of accessing the root folder link (click_root_folder), users accessed the link "my portfolios" (click_my_portfolios). The referred action sent users back to the list of portfolios, where they needed to enter again in the portfolio to continue the task (click_port_open).

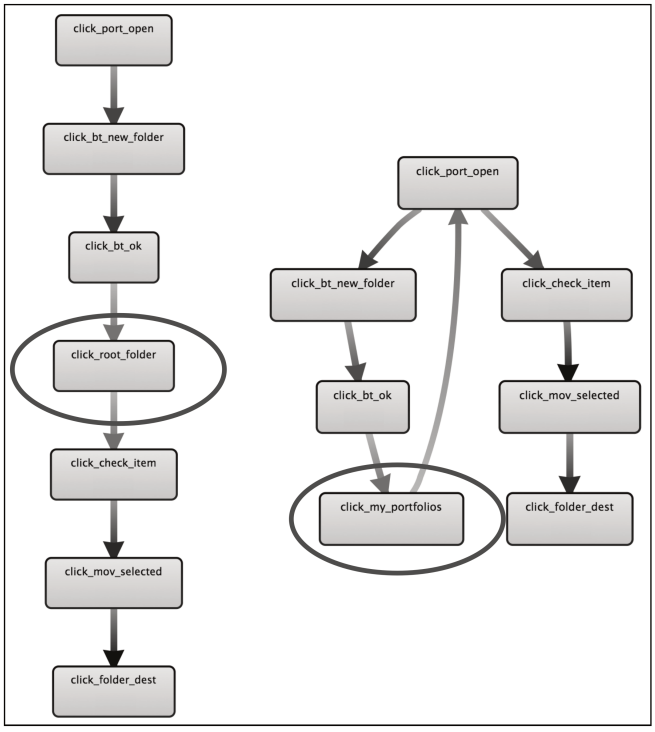

Figure 2. Expected behavior for the task (left) – users returning to the root folder. Pattern found (right) - users leaving the portfolio and entering again to finish the task.

The behavior of users shows that most of them did not notice the link to return to the root folder, indicating a visibility problem with the link. It led users to take a longer path to execute the task. This problem was only detected in the second study, with the new version of the tool.

D. Portfolio: item association

Analyzing the task consisting of posting one item in the portfolio and associating it to an evaluation, it was possible to observe that 50% of users (7 in 14) presented a behavior different from the expected. The pattern found indicates that users pressed several times the buttons that associate the posted item to an evaluation.

In order to discover the cause of this unexpected behavior, we followed the task in the interface of the application. We discovered that the interface opens a small panel when users select the option to start the association, as illustrated in Figure 3 - a.

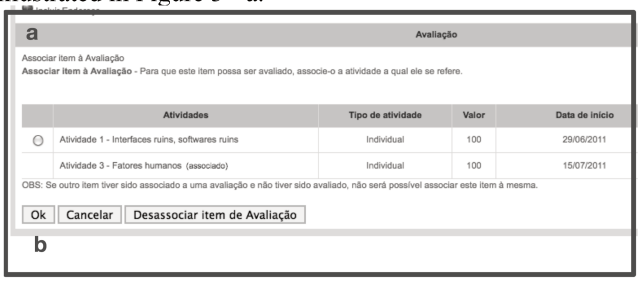

Figure 3. The panel (a) does not close when users press the buton OK (b) to finish the task.

However, the panel does not close when users press the button OK (Figure 3 - b) to finish the task. For this reason, users pressed several times the button OK, trying to finish the task. This usability problem is caused by a lack of feedback. The interface should inform users if the task was successfully executed or not. Furthermore, the panel remaining open gives to users the wrong impression that the task is still not completed. The referred problem was found only in the second study, with the new version of the tool.

E. Mail: message box

In the task consisting of creating and sending a message using the mail tool, 61% of users (32 in 52) performed a different pattern from the expected. The pattern found shows that users clicked frequently on the message box before starting to write a message, as illustrated in Figure 4.

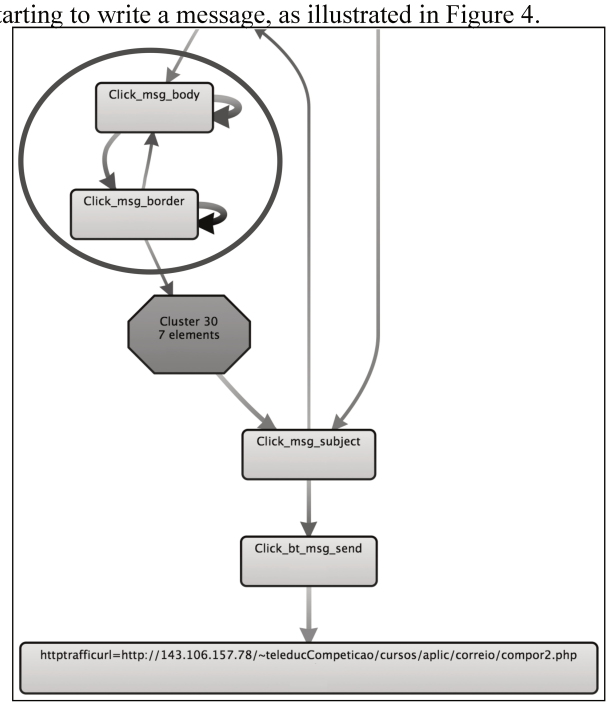

Figure 4. Users clicking several times in the message box, trying to make the cursor appears.

Reproducing the sequence of actions to compose a message in the interface of the application, it was possible to indentify that, when users click on the message box, the cursor does not appear in the box, it disappears from the screen. For this reason, it was difficult for users to realize that the focus of the application was already in the message box. It led them to click repetitive times on the message box, trying to put the focus there. This problem appeared in the first study but it was corrected in the new version of the application. This way, users did not presented the referred behavior in the second study, validating the correction of the problem.

F. Mail: message window size

Analyzing the task in which users should reply a certain message in the mail tool, it was possible to identify a behavior different from the expected for the task. In the pattern found 25% of users (3 in 12) opened and closed the message some times before replying it. Following the task in the interface of the tool, we discovered that the window opened for replying the message has an inadequate size for the content, hiding the button Send. This way, as observed in the pattern found, some users only saw the button when they scrolled the page, explaining why they opened and closed the message some times before replying it. This problem appeared only in the second usability study, with the new version of the application.

G. Mail: replying all

Analyzing the task in which users should reply one email message to all senders, we found a pattern of interaction different from the expected. After pressing the button to reply the message to all, users selected again the receivers of the message before sending it. Following the task in the interface of the application, we observed that the unexpected behavior is caused by the problem illustrated in Figure 5.

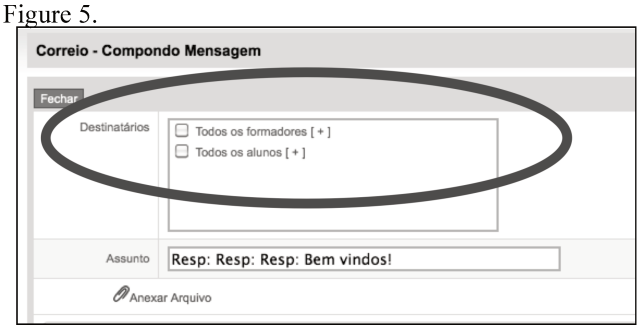

Figure 5. The interface hides the receivers of the message.

The Figure 5 shows that the interface hides the receivers when users are replying the message. For this reason, 50% of users (6 in 12) tried to select again the receivers of the message. This usability problem led users to perform this unnecessary step in the task to assure that the message was going to be correctly sent. This problem only appeared in the second usability study, with new version of the application.

H. Forum: messages visibility

In the first usability study we discovered that in the task in which users needed to answer a message posted in the Forum, 41% of users (21 in 52) read several answers posted by others, before answering the message. This unexpected behavior, illustrated in Figure 6, could be explained as users analyzing others answers before composing their own.

However, observing the interface of the forum, it was possible to notice that 21% of users posted their answers not below the referred question in the hierarchy of the forum, as depicted in Figure 7. This way, it is possible to identify that some users had difficulty in finding the correct place to answer the message posted, indicating a visibility problem in the hierarchy of the forum.

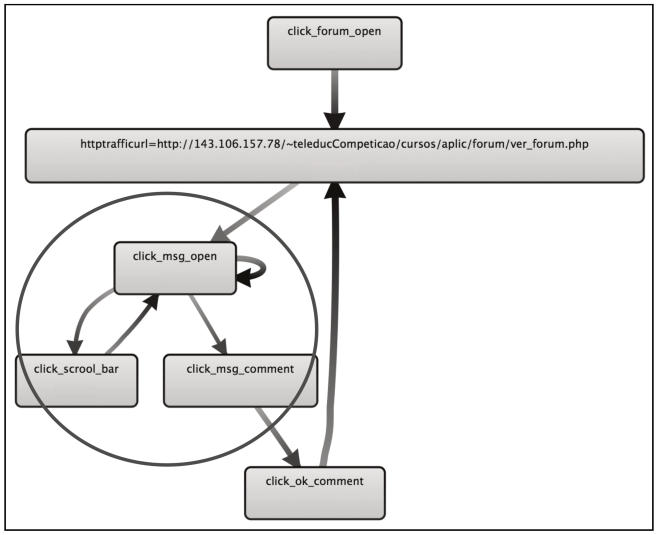

Figure 6. Pattern found: users reading several messages before posting their message.

In the second usability study, the same behavior appeared in 33% of users performing the task (3 in 9), showing that the referred usability problem is still present in the new version of the application.

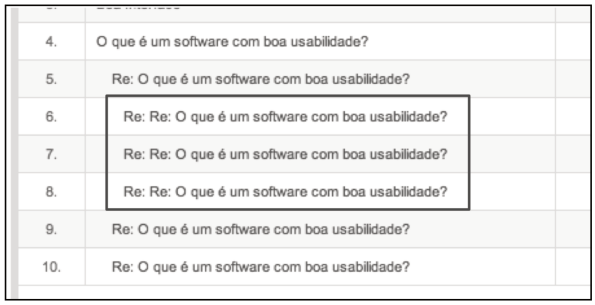

Figure 7. Users answered the message in the wrong place.

I. Forum: message box

In the task consisting of answering a message in the forum we found a pattern of behavior not expected for the task, as illustrated in Figure 8. The pattern found shows that users, after selecting the option for answering the message, clicked several times on the message box and its borders.

Analyzing the interface of the tool, we discovered that sometimes the message box do not became enabled for writing when users try to answer a message. Due to this problem, users tried to click several times on the message box, trying to enable it for writing. In the study 33% of users (3 in 9) experienced this problem. This problem was only found in the new version of the application (second study). However, in the previous version (first study) this problem appeared in the mail tool, as described in section E.

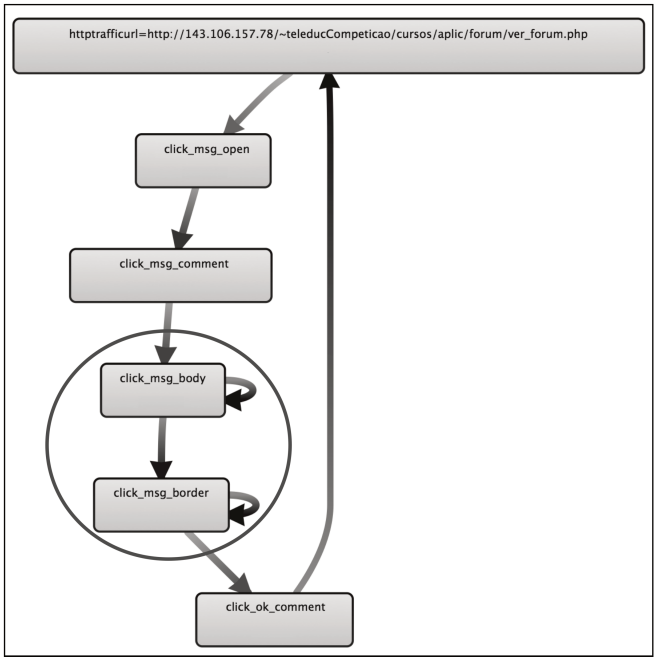

Figure 8. Users clicking several times on the message box of the forum.

IV. CONCLUDING REMARKS

In the described usability studies we analyze the interaction of users in two versions of the Web application TelEduc. The studies allowed us to verify the applicability of the method for successive usability evaluations, analyzing the interaction of users in different versions of the application.

In the first study, we had the costs of setting up the environment to capture data from users' interaction. However, as we used a proxy approach to capture the data, there was no need to change or install software on the application server or on users' computers. We only needed to place the proxy between the application server and the users. This way, they accessed the application thought the proxy, and their interaction with the application was captured. In the second study, we reused the same environment already set, saving time and avoiding repeating this activity.

In the first study, we also defined the tasks performed by users and analyzed in the data analysis phase of WebHint. In the second study, we reused the same set of previously defined tasks, saving time and avoiding repeating also this activity.

Thus, it is possible to observe that using the WebHint method we can repeat the usability studies any time and as often as necessary, having the costs to define the tasks and to set the environment for the data capture only once.

As in the studies the interaction of users was captured remotely and automatically, they accessed the application in their usual work environment, maintaining the context of usage. Furthermore, the analysis of the logs was performed using process mining tools, allowing us to deal with a large quantity of data from several users.

This way, different from other methods as user tests or field observation, we had no limitation on the number of users participating in the study. Using the WebHint method, increasing the number of users involved in the study does not increase the costs of capturing the data or the time and workload to perform the data analysis.

Regarding the usability of the evaluated application, the results of the studies showed that the problem related in the item E of section III was fixed in the new version, not appearing in the behavior of users in the second study, validating its correction.

However, the results of the studies also showed that some problems found in the first experiment are still present in the new version of the application, affecting the behavior of users as described in items A,B and H of section III.

Furthermore, most of the problems were found only in the second experiment with the new version of the application, as exemplified in items C, D, F, G and I, of section III. They indicate that the new version has usability problems not present in the previous version, showing that the usability of the application decreased in the new version.

Therefore, the described usability studies and their results show how the WebHint method can be used as an alternative approach to carry out successive evaluations of the usability of an application, in order to analyze the evolution of different versions of its interface.

The remote and automatic data capturing and the semi-automatic data analysis, supported by process mining, allows us to preserve the context of usage of the tasks and gives us also the possibility of involving how many users we want in the evaluation, without increasing the costs or time of the evaluation. This way, the WebHint method can be used as an alternative approach to carry out usability evaluations of Web applications.

REFERENCES

[1] van der Aalst, W.M.P., Weijters, A.J.M.M. 2004. Process mining: a research agenda. Computers in Industry 53, 231–244.

[2] Andreasen, M. S., Nielsen, H. V., Schrøder, S. O., and Stage, J. 2007. What happened to remote usability testing?: an empirical study of three methods. In Proc. of the SIGCHI - CHI '07. ACM, New York, NY, 1405-1414.

[3] Atterer, R. 2006. Logging Usage of AJAX Applications With the "UsaProxy" HTTP Proxy. In Proc. of the WWW 2006 Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection, Edinburgh, Scotland, May 2006.

[4] Baker, S.; Au, F.; Dobbie, G.; Warren, I. 2008. Automated Usability Testing Using HUI Analyzer. In ASWEC 2008, vol., no., pp.579-588, 26-28 March 2008

[5] van Dongen, B.F., Alves de Medeiros, B.F., Verbeek, B.F., Weijters, A.J.M.M. and van der Aalst, W.M.P. 2005. The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo and P. Darondeau, editors, Application and Theory of Petri Nets 2005, volume 3536 of Lecture Notes in Computer Science, pages 444–454. Springer-Verlag, Berlin, 2005.

[6] Etgen, M. and Cantor, J. 1999. What does getting WET (web event-logging tool) mean for web usability. In Proc. of the Fifth Conference on Human Factors & the Web (Gaithersburg, MD, 1999).

[7] Hong, J, I., Heer, J., Waterson, S., Landay, J,A. 2001. WebQuilt: A proxy-based approach to remote web usability testing. ACM Trans. Inf. Syst. 19, 3 Jul. 2001.

[8] López J. M., Fajardo I., Abascal J.2007 Towards Remote Empirical Evaluation of Web Pages' Usability. In Jacko J. A.(Ed.): Human-Computer Interaction. Interaction Design and Usability. Part I. LNCS 4550

[9] Nielsen, J. Usability Engineering, Academic Press, Boston, MA, 1993.

[10] Obendorf, H., Weinreich, H., and Hass, T. 2004. Automatic support for web user studies with SCONE and TEA. In CHI '04. ACM, New York, NY, 1135-1138.

[11] Paterson, M., and Dancik, V. 1994. Longest Common Subsequences. In B. Rovan I. Privara and P. Ruzicka, editors, 19th MFCS'94, LNCS 841, pages 127-142, Kosice, Slovakia, August 1994. Springer Verlag.

[12] Vargas, A., Weffers, H. T. G., Rocha, H. V. A Method for Remote and Semi-Automatic Usability Evaluation of Web-based Applications Through Users Behavior Analysis. In: Proc. of the Measuring Behavior 2010 – 7th International Conference on Methods and Techniques in Behavioral Research, Eindhoven, The Netherlands, 2010 (in press).

[13] Vargas, A., Weffers, H. Rocha, H. V. Usability Analysis of User Interaction in Online Applications. In: Proceedings of ICCEE 2010 - 3rd International Conference on Computer and Electrical Engineering, Chengdu, China, 2010.

Capítulo 5

Conclusões

Esta tese apresentou o desenvolvimento de uma pesquisa realizada com o objetivo de desenvolver um método para a avaliação de usabilidade de aplicações Web baseado na captura e na análise automatizada da interação de usuários.

A pesquisa desenvolvida nesta tese teve como motivação as dificuldades existentes e limitações que os métodos tradicionais de avaliação de usabilidade possuem quando executados para avaliar aplicações Web, especialmente em sua fase de manutenção.

Os métodos tradicionais, como teste com usuários e observação em campo, são considerados métodos bastante caros devido aos custos de envolver usuários, de preparar a infraestrutura e especialmente da própria execução dos métodos e análise de seus resultados.

A qualidade dos resultados da execução desses métodos depende da escolha dos usuários participantes da avaliação, que precisam ser representativos de todos os perfis de usuários da aplicação (Nielsen, 1993). Na Web, devido à diversidade e característica distribuída dos usuários, a obtenção de usuários representativos dos diversos perfis torna-se bastante difícil, quando não, impraticável.

Além disso, os métodos tradicionais de avaliação de usabilidade baseiam sua execução na observação e análise de usuários interagindo com as aplicações. Esta análise depende do conhecimento e experiência dos especialistas analisando a interação dos usuários para encontrar problemas de usabilidade.

Desse modo, os métodos tradicionais, se utilizados em grande escala de usuários, tornam-se demasiadamente caros devido a seus custos e tempo de execução. Além disso, por basear sua execução no esforço e conhecimento de especialistas, esses métodos não são passíveis de automatização.

Outra característica importante de aplicações Web é sua constante evolução, o que necessita de avaliações de usabilidade periódicas e constantes para garantir que alterações nas aplicações não prejudiquem sua usabilidade.

Da mesma forma, os métodos tradicionais de avaliação de usabilidade tornam-se demasiadamente caros, em termos de tempo e custo, para serem utilizados em avaliações constantes de usabilidade de aplicações Web.

Assim, a principal hipótese assumida nesta tese foi a de que*, utilizando-se um método de avaliação de usabilidade fundamentado na captura e análise da interação de usuários de maneira remota e automatizada, é possível diminuir as dificuldades existentes nos métodos tradicionais na fase de manutenção de uma aplicação Web, possibilitando ainda: a análise da interação de usuários em campo e em grande escala de usuários; a identificação de padrões de interação dos usuários com a aplicação; uma avaliação com baixo custo de execução, favorecendo a realização de avaliações periódicas de usabilidade de uma aplicação Web.*

O método WebHint, desenvolvido como resultado desta tese, permitiu validar a hipótese assumida e responder às questões de pesquisa descritas no capitulo introdutório desta tese e reapresentadas a seguir.

A primeira questão de pesquisa levantada foi: *Como realizar a avaliação de usabilidade de uma aplicação Web, capturando-se a interação de seus usuários em campo, de maneira remota e automatizada, sem interferir na interação dos usuários, possibilitando a obtenção de dados de grandes quantidades de usuários, ou até mesmo de todos os usuários acessando a aplicação avaliada?*

No WebHint, conforme detalhado nos capítulos anteriores, a captura dos dados da interação dos usuários é realizada utilizando-se uma abordagem de *proxy* remoto, permitindo o registro detalhado de todas as ações dos usuários na interface da aplicação avaliada.

Assim, a captura da interação dos usuários é realizada de maneira remota e automática, sem a necessidade de se alterar a aplicação avaliada ou instalar software nos servidores da aplicação ou nos computadores dos usuários, permitindo que a captura da interação seja realizada em larga escala.

Além disso, a captura é também realizada de maneira transparente para os usuários, isto é, sua interação com a aplicação não é alterada, pois os usuários acessam a aplicação em seu ambiente real de utilização, executando suas tarefas reais.

Nos estudos de usabilidade utilizando-se o WebHint, relatados nos artigos que compõem o Capítulo 4 desta tese, a captura da interação dos usuários foi feita de maneira remota e

automática, utilizando-se a ferramenta UsaProxy, que implementa a abordagem de *proxy* remoto, validando a abordagem selecionada.

O processo de análise de dados definido no WebHint responde às perguntas de pesquisa 2 e 3, levantadas no Capítulo 1 desta tese, que consistem em:

Como analisar, de maneira automatizada, os dados obtidos na captura automática da interação dos usuários, de modo a lidar com grandes volumes de informação, sem multiplicar o custo e tempo da análise, para cada usuário envolvido na avaliação?

Como analisar a interação dos usuários de modo a obter informações sobre o comportamento predominante e tendências de uso da aplicação, levando em conta os usuários como um todo?

Na etapa de análise de dados do método WebHint, definiu-se a utilização de um conjunto de técnicas para a manipulação dos logs de interação, que automatizam as etapas do processo de análise. Todo o processo de manipulação dos logs e extração das tarefas analisadas é automatizado, dessa maneira, ao aumentar o número de usuários envolvidos na análise, não se multiplica o tempo ou o trabalho despendido na análise.

A análise das tarefas extraídas dos logs, utilizando-se *process mining,* permite que se obtenha automaticamente, a partir das informações constantes no log, um modelo real de execução das tarefas, que representa os padrões predominantes de execução dessas tarefas pelos usuários. Desse modo, é possível analisar o comportamento dos usuários como um todo, identificando as tendências de uso e a maneira com a qual as tarefas são executadas pelos usuários.

Os estudos de usabilidade desenvolvidos e relatados nos artigos que compõem o Capítulo 4 desta tese, permitiram validar o WebHint como alternativa para se avaliar a usabilidade de aplicações Web através da captura e análise automatizada da interação de usuários, validando também as respostas às questões de pesquisa mencionadas anteriormente.

No terceiro artigo que compõe o Capítulo 4, demonstrou-se como o WebHint pode ser utilizado como alternativa para a realização de sucessivas análises de usabilidade de uma aplicação Web, respondendo-se à quarta questão de pesquisa levantada no capítulo introdutório desta tese:

Como desenvolver a avaliação de modo que esta possa ser repetida quantas vezes for necessário sem possuir um alto custo em sua execução, favorecendo análises periódicas da usabilidade da aplicação?

A possibilidade de se executar sucessivas avaliações de usabilidade pode ser utilizada tanto durante a fase de desenvolvimento da aplicação, quanto após seu lançamento, para avaliar a usabilidade de diferentes versões da aplicação.

Conforme relatado no terceiro artigo do Capítulo 4, foram realizados dois estudos de usabilidade utilizando o WebHint para analisar diferentes versões da aplicação TelEduc.

No primeiro estudo, teve-se o custo de preparação do ambiente para a captura dos dados, isto é, instalar o servidor *proxy* e também os custo de definição das tarefas. Entretanto, utilizando-se a abordagem de *proxy* remoto evitou-se a necessidade de alterar a aplicação ou instalar software no servidor ou computadores dos usuários. Assim, foi necessário

somente configurar o *proxy* para ser executado entre o servidor da aplicação e os usuários.

No segundo estudo, pode-se reutilizar toda a infraestrutura de *proxy* para realizar a captura da interação, evitando-se os custos e a necessidade de se repetir esse trabalho. As tarefas definidas também foram reutilizadas, pois o segundo estudo utilizou uma nova versão da mesma aplicação, evitando também os custos de se repetir todo esse trabalho.

Assim, com a análise automatizada dos dados capturados utilizando-se o método WebHint, o estudo poderia ser repetido quantas vezes fosse necessário sem custo adicional para a captura e com um custo constante para a análise dos dados, tornando viável a realização de avaliações sucessivas da usabilidade da aplicação.

Analisando-se diferentes versões da aplicação, foi possível verificar se problemas encontrados na primeira análise foram solucionados, ou não, na nova versão da aplicação. Conforme descrito no artigo, alguns problemas encontrados no primeiro estudo foram solucionados na nova versão, mas outros problemas encontrados no primeiro estudo permaneceram na nova versão da aplicação. Além disso, o segundo estudo demonstrou que a nova versão da aplicação possui problemas de usabilidade não encontrados na versão anterior, o que indica que alterações na interface prejudicaram sua usabilidade.

Desse modo, pode-se demonstrar a viabilidade de se utilizar o WebHint para avaliar a evolução de uma aplicação e também observar a importância de se avaliar a usabilidade de aplicações Web ao longo de sua evolução.

5.1 O WebHint e seus objetivos

O método proposto nesta tese não tem o objetivo de substituir os métodos tradicionais de avaliação de usabilidade. Os métodos tradicionais de avaliação de usabilidade, como testes com usuários e observação em campo, têm a possibilidade de realizar uma análise mais detalhada da interação dos usuários, pois além das ações que o usuário realiza durante sua interação com a aplicação, estes métodos são capazes de obter outras informações importantes referentes aos usuários. Observando-se um usuário é possível identificar comentários, intenções, características emocionais (satisfação, frustração, etc.) e reações que o usuário apresenta a cada interação com a aplicação. Essas informações não são possíveis de serem capturadas e analisadas automaticamente.

Dessa forma, o WebHint propõe uma análise em uma perspectiva diferente dos métodos tradicionais, conforme já mencionado, analisando a interação sob a ótica de padrões de interação dos usuários como um todo. Assim, o WebHint pode ser utilizado de diversas maneiras e com diversos objetivos como:

Primeira visão geral da interação

O WebHint pode ser utilizado como uma primeira análise da interação dos usuários para identificar tarefas, funcionalidades ou partes de uma aplicação que apresentam problemas e necessitam de uma avaliação mais detalhada. Assim, métodos mais caros, como testes, podem ser utilizados para analisar os pontos específicos da aplicação que apresentam problemas. Desse modo, diminui-se os custos de se realizar, por exemplo, testes de usabilidade em toda a aplicação, deixando-os para serem realizados em funcionalidades, tarefas e partes mais criticas da aplicação.

Análise da interação dos usuários como um todo

Padrões de interação na realização de tarefas na aplicação não são observáveis nos métodos tradicionais, pois estes são executados com pequenos grupos de usuários. Desse modo, o WebHint pode ser utilizado para agregar esta perspectiva à análise, oferecendo uma visão mais abrangente da interação dos usuários como um todo. Dessa forma, é possível obter a informação de, por exemplo, quantos usuários são afetados por um certo problema de usabilidade encontrado. Informações desse tipo auxiliam a definir a severidade de cada problema encontrado em uma avaliação e sua prioridade em ser corrigido. Além disso, apresentar informações tais como o número de usuários afetados por um problema é importante para evidenciar a necessidade de correção do problema, tarefa muitas vezes difícil em avaliações de usabilidade.

Análises sucessivas

Conforme já mencionado, o método proposto pode ser utilizado para realizar avaliações sucessivas da usabilidade de uma aplicação. Desse modo, é possível monitorar e analisar periodicamente a interação dos usuários, visando verificar o surgimento de problemas de usabilidade durante a evolução da aplicação. Além disso, é possível também verificar como os usuários interagem com a aplicação após a correção de certos problemas, verificando se as alterações obtém o sucesso esperado.

5.2 O WebHint e seus principais avanços e contribuições

O WebHint constitui-se como um método original para a realização de avaliações de usabilidade de aplicações Web, no qual o processo de captura e análise dos dados da

interação de usuários, apresenta avanços em relação às abordagens existentes, descritas na literatura estudada e apresentada no Capítulo 2 desta tese.

Na captura dos dados, o WebHint utiliza a abordagem de *proxy* remoto para a captura da interação dos usuários em logs, que possui as vantagens já mencionadas anteriormente em relação a outras abordagens existentes (logs de servidores Web e captura através de software instalado no computador dos usuários).

Apesar de ser uma abordagem já conhecida, no WebHint, a abordagem de *proxy* remoto foi utilizada pela primeira vez como parte integrante de um método completo, que contempla todas as etapas de uma avaliação automatizada de usabilidade.

Na análise dos dados capturados na interação dos usuários, o WebHint apresenta um processo original de análise de logs. Diversas técnicas são utilizadas de maneira integrada e complementar para a realização do pré-processamento e extração de tarefas dos logs. As técnicas de seleção e recodificação são utilizadas no pré-processamento dos logs e a técnica de comparação de sequências *LCS (Longest Common Subsequece)* é utilizada na extração de tarefas dos logs.

Além disso, no WebHint utilizou-se, pela primeira vez, técnicas de *process mining* para a análise da interação de usuários com aplicações Web. Conforme já mencionado, na literatura, poucos são os registros da utilização de *process mining* para analisar logs de eventos de usuários, sendo os trabalhos de Hofstra (2009) e Funk et al. (2010) os únicos encontrados nesse tema até o momento.

No entanto, nos trabalhos de Hofstra (2009) e Funk et al. (2010), os logs de eventos analisados foram obtidos a partir da interação de usuários com aparelhos de TV, que possuem interfaces bem mais simples do que a maioria das aplicações Web. Assim, as tarefas executadas por usuários em aplicações Web são mais complexas, com maior quantidade e variedade de eventos. Além disso, conforme já mencionado, a quantidade de ruídos nesses logs é bastante alta, tendo-se ainda as dificuldades existentes em se lidar com as diferenças de relevância entre os eventos nesses logs.

Essas características tornaram desafiante e inovador o uso de *process mining* para a análise de logs de eventos de usuários em aplicações Web. Conforme resultados dos estudos de usabilidade relatados no Capítulo 4, a aplicação de *process mining* utilizada no WebHint para analisar eventos de usuários em aplicações Web, constitui-se em um avanço importante nessa área de estudo e pesquisa.

Além disso, o WebHint possui ainda avanços em relação ao processo de análise de dados aplicado em outras abordagens existentes e mencionadas no Capitulo 2. O WebHint baseia sua análise no modelo de execução real das tarefas executadas pelos usuários e registradas no log. Este modelo é obtido através de técnicas de *process mining,* levando em conta a interação de todos os usuários interagindo com a aplicação. Assim, a análise do WebHint permite um visão da interação dos usuários como um todo, isto é, favorece um visão geral da interação com base nos padrões de execução das tarefas.

Outras abordagens como o WebRemUsine (Paganelli & Paternò, 2002), AWUSA (Tiedtke et. al 2002) e WAUTER (Balbo et al., 2005) baseiam sua análise na comparação automática entre duas sequências, a sequência esperada para a execução de uma tarefa e a sequência de eventos que um usuário executou. Dessa forma, tal qual o WUP (Carta et

al., 2011), apenas permitem uma análise individualizada, ou seja, analisam o comportamento de um único usuário, sem levar em conta os padrões de interação apresentados pelos usuários como um todo. Além disso, as abordagens WebRemUsine, AWUSA e WAUTER possuem as limitações e problemas relatados anteriormente (seção 2.3.2.4), relacionados à comparação automática de sequências de eventos de usuário.

O WebHint possui ainda vantagens em relação às abordagens utilizadas no WebQuilt (Hong et al., 2001), WELFIT (Santana & Baranauskas, 2008) e Experiscope (Guimbretière et al., 2007), que baseiam sua análise na sumarização de eventos ocorridos na interação dos usuários. O WebQuilt encontra os caminhos mais comuns de navegação entre as páginas de uma aplicação. Comparando-se se a ele, o WebHint possui a vantagem de, além da navegação entre as páginas, levar em conta as ações executadas também dentro das páginas, o que possibilita uma análise mais detalhada das tarefas. Da mesma forma, o WebHint possui vantagens em relação ao WELFIT e ao Experiscope, pois estes baseiam sua análise em eventos que ocorrem apenas dentro de uma determinada página, não analisando tarefas que envolvem a navegação entre diferentes páginas da aplicação.

5.3 Limitações, Escopo e Trabalhos Futuros

O WebHint pode ser usado para avaliar qualquer aplicação Web desde que esta implemente uma arquitetura de cliente e servidor, devido à abordagem selecionada para a captura dos dados, a abordagem de *proxy* remoto.

A forma com a qual os dados são capturados também define os tipos de aplicações Web que podem ser avaliadas. Por exemplo, nos estudos de usabilidade, relatados no Capítulo

4, foi utilizada a ferramenta UsaProxy, que implementa a abordagem de *proxy* remoto para a captura dos dados. Utilizando-se esta ferramenta a captura fica limitada a ser realizada em aplicações Web desenvolvidas em HTML (W3C, 2011) e AJAX (Garret, 2005), pois o UsaProxy não permite realizar a captura dentro de elementos desenvolvidos em Flash e *applets* Java.

Para capturar a interação dos usuários em aplicações desenvolvidas com diferentes tecnologias, o *proxy* utilizado deve ser capaz de instrumentar automaticamente essas aplicações.

A utilização da abordagem de *proxy* remoto, apesar de possuir as vantagens já mencionadas (seção 2.2), requer que a infraestrutura utilizada seja adequada ao fluxo de usuários que acessam a aplicação avaliada, para garantir que a instrumentação dinâmica não altere o tempo de resposta normal da aplicação e prejudique a interação dos usuários. Desse modo, alguns fatores precisam ser levados em conta quando a abordagem é utilizada, como:

- Servidor *proxy*: o computador que funciona como *proxy* entre os usuários e a aplicação deve possuir capacidade de processamento suficiente para dar suporte à quantidade de usuários simultâneos que acessam usualmente a aplicação.
- Largura de banda: a conexão do servidor *proxy* com o servidor da aplicação precisa ter velocidade suficiente para garantir que a comunicação entre o *proxy* e o servidor não altere o tempo de resposta normal da aplicação.

Nos estudos de usabilidade realizados e descritos Capítulo 4, não foram necessários requisitos especiais para a arquitetura de *proxy* implementada, pois o número de usuários acessando simultaneamente a aplicação era relativamente pequeno, variando entre 10 e

50. Desse modo, utilizou-se uma máquina virtual no *cluster* do Instituto de Computação da Unicamp para servir como *proxy* durante os estudos. Testes realizados antes dos estudos demonstraram que o *proxy* não alterava o tempo de resposta da aplicação de modo perceptível para o usuário.

A definição das tarefas no WebHint é um estágio bastante importante no processo de execução da avaliação. No WebHint, optou-se por não utilizar uma definição abstrata de tarefas visando-se simplificar e diminuir o tempo necessário na definição das tarefas. Mesmo assim, dependendo da aplicação e complexidade das tarefas analisadas, esta etapa pode demandar bastante trabalho e tempo em sua execução. Assim, a quantidade de tarefas e a complexidade das tarefas é um fator determinante no custo da análise utilizando-se o WebHint. Desse modo, uma das questões de pesquisa que podem ser exploradas em trabalhos futuros realizados a partir desta tese é:

Como tornar a etapa de definição das tarefas menos trabalhosa e eficiente em termos de tempo e custo?

Apesar de ser uma questão que necessita de uma pesquisa mais extensiva para ser respondida, alguns métodos e abordagens estudados no Capítulo 2 desta tese podem servir como um direcionamento para responder esta pergunta.

Os métodos de avaliação de usabilidade baseados em modelos e simulação, por realizar a geração de dados sintéticos, simulando a interação de usuários em uma aplicação, com base em modelos pré-definidos, poderiam ser utilizados para gerar as sequências "ideais" de eventos que definem as tarefas a serem analisadas. Dependendo da complexidade da aplicação, esta abordagem poderia se fazer viável na tentativa de diminuir os custos da

etapa de definição das tarefas, evitando que um especialista precisasse definir as tarefas executando a sequência de eventos ideal de cada uma.

No entanto, é necessário garantir que os custos da definição dos modelos e de toda a infraestrutura necessária para realizar as simulações não aumentem ainda mais o custo da definição das tarefas, como acontece em abordagens tais quais o WebRemUsine (Paganelli & Paternò, 2002).

No primeiro artigo publicado em Vargas et al. (2010), tinha-se a intenção de incluir na etapa de análise de dados do WebHint, um mecanismo para automaticamente detectar problemas de usabilidade já conhecidos, isto é, baseando-se em um comportamento previamente detectado. Conforme mencionado nos capítulos 2 e 3, realizar a comparação automática de sequências de logs de eventos de interação depende do pré-processamento do log, isto é, evidenciar os eventos relevantes e subtrair os eventos de pouca relevância na execução das tarefas.

Mesmo executando a "limpeza" do log, determinados eventos, conforme exemplificado anteriormente (seção 2.3.2.4), não podem ser retirados do log de forma generalizada e automática, pois podem não ser relevantes em uma determinada tarefa, mas em outras podem ter alta relevância. Assim, para a realizar uma detecção automática que gerasse bons resultados seria necessária uma limpeza do log feita de maneira específica para cada tarefa, analisando-se a interação de cada usuário. Desse modo, o trabalho prévio de limpeza do log seria muito trabalhoso e demorado, não justificando os benefícios da comparação automática para detectar sequências associadas a problemas já conhecidos.

A mesma dificuldade em se comparar automaticamente sequências de interação resultou na opção de deixar a etapa final de comparação entre o modelo de tarefa esperado e o modelo obtido da interação dos usuários a cargo do avaliador. Desse modo, outra questão de pesquisa que poderia ser explorada em trabalhos futuros partindo desta tese é:

Como executar a etapa de pré-processamento do log de maneira automatizada e mais eficiente, de modo a possibilitar a aplicação de técnicas de comparação automática de sequências que gerem bons resultados em logs de interação?

Esta é uma questão que necessita de pesquisas mais aprofundadas para ser respondida. No entanto, algumas direções podem partir da área de *machine learning* (Alpaydin, 2004), na busca por técnicas que permitam identificar a relevância de eventos em um log, levando em conta sua presença em sequências que representam tarefas previamente analisadas.

Tornando a etapa de pré-processamento de logs de interação mais eficiente, técnicas automatizadas de comparação de sequências e técnicas de checagem de conformidade em *process mining* poderiam ser utilizadas para automatizar a análise dos resultados da avaliação, diminuindo a o esforço do especialista na etapa final da avaliação.

Melhorando-se a etapa de análise dos dados de modo a automatizar a etapa final da avaliação, abre-se a possibilidade de expandir o método em outras direções. Por exemplo, na possibilidade de criação de um sistema mais automatizado, que monitorasse a interação dos usuários em uma aplicação e realizasse comparações constantes e automáticas entre a interação dos usuários e o modelo esperado para cada tarefa. Dependendo das diferenças encontradas, o sistema poderia indicar a existência de problemas de usabilidade e sugerir a verificação por um especialista. Assim, a avaliação

de usabilidade seria automática, constante e contínua, auxiliando o especialista, de forma mais eficiente, na detecção de problemas de usabilidade.

Referências Bibliográficas

van der Aalst, W. M. P. 2011. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, Berlin.

van der Aalst, W.M.P., Weijters, A.J.M.M. 2004. Process mining: a research agenda. *Computers in Industry* 53, 231–244.

Alpaydin, Ethen. 2004. *Introduction to Machine Learning*. MIT Press.

Al-Qaimari, G., Mcrostie, D. 1999. KALDI: A computer-aided usability engineering tool for supporting testing and analysis of human-computer interaction. In J. Vanerdonckt and A. Puerta, Eds., *Proceedings of the Third International Conference on Computer-Aided Design of User Interfaces* (Louvain-la-Neuve, Belgium, October). Dordrecht, The Netherlands: Kluwer Academic Publishers.

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*, Edinburgh, Scotland, May 2006.

Atterer, R., Schmidt, A. 2007. Tracking the interaction of users with AJAX applications for usability testing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA, April 28 - May 03, 2007). CHI '07. ACM, New York, NY, 1347-1350.

Atterer, R., Wnuk, M., Schmidt, A. 2006. Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. In *Proceedings of the 15th international Conference on World Wide Web* (Edinburgh, Scotland, May 23 - 26, 2006). WWW '06. ACM, New York, NY, 203-212.

Bacheldor, B. 1999. Push for performance. *Information Week September* 20, 18–20.

Balbo, S. 1996. EMA: Automatic Analysis Mechanism for the Ergonomic Evaluation of User Interfaces. *CSIRO Tech. rep.*

Balbo, S, Goschnick, S, Tong, D, Paris, C. 2005. Leading Web Usability Evaluations to WAUTER. In *The Eleventh Australasian World Wide Web Conference*, Gold Coast, July, 2005.

Balbo, S; Steinkrug, J; Lee, L; Scheidt, S; Hullin, C and Gogler, Janetter. 2008. The Importance of Including Users in Clinical Software Evaluation: What Usability Can Offer in Home Monitoring [online]. In: Grain, Heather (Editor). *HIC 2008 Conference*: Australias Health Informatics Conference; The Person in the Centre, August 31 - September 2, 2008 Melbourne Convention Centre. Brunswick East, Vic.: Health Informatics Society of Australia, 2008

Baker, S.; Au, F.; Dobbie, G.; Warren, I. 2008. Automated Usability Testing Using HUI Analyzer. In *ASWEC 2008 - 19th Australian Conference on Software Engineering,* vol., no., pp.579-588, 26-28, 2008

Buchholz, G., Engel,J., Märtin,C., Propp, S. 2007. Model-Based Usability Evaluation – Evaluation of Tool Support. *Lecture Notes in Computer Science (Springer)*, Berlim, vol. 4550, pg. 1043-1052, 2007.

Buchner, A. G., Mulvenna, M. D. 1998. Discovering Internet marketing intelligence through online analytical web usage mining. *SIGMOD Rec.* 27, 4 (Dec. 1998), 54-61.

Calisir, F., Bayraktaroglu, A.E., Gumussoy, C.A., Topcu, Y.I., Mutlu, T. 2010. The relative importance of usability and functionality factors for online auction and shopping web sites. *Online Information Review*, Vol. 34 Iss: 3, pp.420 – 439, Emerald Group Publishing Limited, 2010.

Carta, T., Paternò, F., Santana, V. F. 2011. Web Usability Probe: A Tool for Supporting Remote Usability Evaluation of Web Sites. *Proceedings of the 13th IFIP TC 13 International Conference on Human-computer Interaction*, September 05-09, 2011, Lisbon, Portugal

Casaló, L.V., Flavián, C., Guinalíu, M. The role of security, privacy, usability and reputation in the development of online banking. *Online Information Review*, Vol. 31 Iss: 5, pp.583 – 603, Emerald Group Publishing Limited, 2007.

Castillo, Jose C. and Hartson, H. Rex. Remote Usability Testing Methods a la Carte. *Technical Report TR-07-05*, Computer Science, Virginia Tech, 2007.

Cato, J. 2001. *User-Centred Web Design*, Addison Wesley.

Chang, E. Dillon, T. S. 1997. Automated usability testing. In *Proceedings of INTERACT '97.*

Chi, E. H. 2002. Improving Web Usability Through Visualization, *IEEE Internet Computing*, v.6 n.2, p.64-71, March 2002

Chi E. H., Rosien A. and Heer J**.** 2002**.** LumberJack: Intelligent Discovery and Analysis of Web User Traffic Composition. In *Proceedings of ACMSIGKDD Workshop on Web Mining for Usage Patterns and User Profiles*, Canada, ACM Press, 2002.

Chilana, P.K., Ko, A.J., Wobbrock, J.O., Grossman, T. Fitzmaurice, G. 2011. Post-deployment usability: a survey of current practices. In *Proceedings of the 2011 annual conference on Human factors in computing systems* (CHI '11). ACM, New York, NY, USA, 2243-2246. DOI=10.1145/1978942.1979270 http://doi.acm.org/10.1145/1978942.1979270

Dai H, Mobasher B. 2005. Integrating Semantic Knowledge with Web Usage Mining for Personalization. In *Web Mining: Applications and Techniques*, Anthony Scime (ed.), IRM Press, Idea Group Publishing.

Dix, A, Finlay, J., Abowd, G. D., and Beale, R. 2004. *Human - Computer Interaction*. Prentice Hall, Harlow, England.

Domenech, J. M., Lorenzo, J. A Tool for Web Usage Mining. 2007. In: *Intelligent Data Engineering and Automated Learning - IDEAL 2007*. Lecture Notes In Computer Science. Volume 4881/2007. Springer Berlin / Heidelberg. 2007

van Dongen, B. F., de Medeiros, A. K. A., Verbeek, H. M. W., Weijters, A., and van der Aalst, W. M. P.. 2005. The ProM framework: a new era in process mining tool support. In: Ciardo, G., and Darondeau, P. (Eds.), Applications and theory of Petri nets 2005,

proceedings, vol 3536. *Lecture Notes in Computer Science*. Springer-Verlag Berlin, Berlin, pp. 444–454, 2005.

Etgen,M. and Cantor, J. 1999. What does getting WET (web event-logging tool) mean for web usability. In *Proceedings of the Fifth Conference on Human Factors & the Web* (Gaithersburg, MD, June).

Fleming, J. 1998. *Web navigation: designing the user experience*, O'Reilly & Associates, Inc., Sebastopol, CA

Finlay, J., Harrison, M. 1990. Pattern recognition and interaction models. In *Proceedings of INTERACT '90.*

Fisher, C. 1991. Protocol Analyst's Workbench: Design and Evaluation of Computer-Aided Protocol Analysis. *PhD. Thesis*, Carnegie Mellon University, Dept. of Psychology, Pitts-burgh, PA.

Funk, M., Rozinat, A., Karapanos, E., de Medeiros, A.A., Koca, A., 2010. In situ evaluation of recommender systems: framework and instrumentation. *International Journal of Human-Computer Studies* 68(8), 525-547.

Garrett JJ. 2005. Ajax: A new approach to web applications. *Adaptive Path*. Feb 18, 2005. http://www.adaptivepath.com/publications/essays/archives/000385.php.

Gottman, J. M, Roy, A. K. 1990. *Sequential analysis: A guide for behavioral researchers*. Cambridge University Press, Cambridge, England.

Guimbretiére, F., Dixon, M., and Hinckley, K. 2007. ExperiScope: an analysis tool for interaction data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA, April 28 - May 03, 2007). CHI '07. ACM, New York, NY, 1333-1342.

Gunther, C.W., van der Aalst, W.M.P. 2007. Fuzzy Mining - Adaptive Process Simplification Based on Multi-perspective Metrics. In *Proceedings of BPM. Lecture Notes in Computer Science*, Springer-Verlag (2007)

Gusfield, D. 1997. *Algorithms on strings, trees and sequences*. Cambridge University Press, New York.

Helfrich, B., Landay, J. A. 1999. QUIP: quantitative user interface profiling. Available at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.2367&rep=rep1&type=pdf

Hilbert, D. M., Redmiles, D. F. 1998. An approach to large-scale collection of application usage data over the Internet. *In Proceedings of ICSE '98.*

Hilbert, D.M., Redmiles, D. F. 2000. Extracting usability information from user interface events. *ACM Comput. Surv.* 32, 4 (December 2000), 384-421. DOI=10.1145/371578.371593 http://doi.acm.org/10.1145/371578.371593

Hofstra, P.P.H.J., 2009. Analysing the effect of consumer knowledge on product usability using process mining techniques. *Master's Thesis*, Eindhoven University of Technology, Department of Industrial Design, Eindhoven, The Netherlands.

Hong, J, I., Heer, J., Waterson, S., Landay, J,A. 2001. WebQuilt: A proxy-based approach to remote web usability testing. *ACM Trans. Inf. Syst*. 19, 3 Jul. 2001.

Ivory, M. Y. and Hearst, M. A. 2001. The state of the art in automating usability evaluation of user interfaces. *ACM Comput. Surv*. 33, 4 (Dec. 2001), 470-516.

Lecerof, A., Paternò, F. 1998. Automatic support for usability evaluation. *IEEE Transactions on Software Engineering*. V. 24, issue 10, 1998.

López J. M., Fajardo I., Abascal J. 2007. Towards Remote Empirical Evaluation of Web Pages' Usability. In Jacko J. A.(Ed.): *Human-Computer Interaction. Interaction Design and Usability*. Part I. LNCS 4550 Springer. Pp. 594-603

Lynch, K., Gilmore, S. Usability: The Key to Product Success. *The Pragmatic Marketer*. Vol 4, Issue 5. Pragmatic Marketing, Inc. Scottsdale, 2006.

Macleod, M., Rengger, R. 1993. The Development of DRUM: A Software Tool for Video-assisted Usability Evaluation. In *Proceedings of HCI '93.*

Mobasher, B. 2007. Data Mining for Web Personalization. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.). *The Adaptive Web: Methods and Strategies of Web Personalization, Lecture Notes in Computer Science*, Vol. 4321. Springer-Verlag, Berlin Heidelberg New York.

Mobasher, B., Cooley, R., and Srivastava, J. 2000. Automatic personalization based on Web usage mining. *Communications of ACM* 43, 8 (Aug. 2000), 142-151.

Nichols, D.M., McKay, D. & Twidale, M.B. (2003). Participatory Usability: supporting proactive users. In E. Kemp, C. Phillips & B. L. W. Wong (Eds), *Proceedings of the 4th Annual Conference of the ACM Special Interest Group on Computer Human Interaction* - New Zealand Chapter (CHINZ'03), 3-4 July 2003, Dunedin, New Zealand (pp. 63-68). Palmerston North, New Zealand: New Zealand Chapter of ACM SIGCHI, c2003.

Nielsen, J. 2000. *Designing Web Usability: The Practice of Simplicity*, New Riders Publishing, Thousand Oaks, CA

Nielsen, J. Heuristic Evaluation. 1994. In Nielsen, J., and Mack, R. L. (Eds.), *Usability Inspection Methods*, John Wiley & Sons, New York, 25-64.

Nielsen, J. 1993. *Usability Engineering*, Academic Press, Boston, MA

Obendorf, H., Weinreich, H., and Hass, T. 2004. Automatic support for web user studies with SCONE and TEA. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems* (Vienna, Austria, April 24 - 29, 2004). CHI '04. ACM, New York, NY, 1135-1138.

Olsen, D. R., Halversen, B. W. 1988. Interface usage measurements in a user interface man- agement system. In *Proceedings of UIST '88.*

Oreilly, T. 2007. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. *Communications & Strategies*, No. 1, p. 17.

Palmer, J. 2002. Designing for web site usability. *Computer*, 35(7):102-103, IEEE.

Paganelli, L , Paternò, F. 2002. Intelligent analysis of user interactions with web applications. In *Proceedings of the 7th international conference on Intelligent user interfaces*, January 13-16, 2002, San Francisco, California, USA

Pascual, V, Dursteler, J.C. 2007. WET: a prototype of an Exploratory Search System for Web Mining to assess Usability, iv,pp.211-215, In *11th International Conference Information Visualization (IV '07)*, 2007

Paterno, F. 2000. Model-based design of interactive applications. *Intelligence* 11, 4 (Dec. 2000), 26-38.

Paterno, F., Ballardin, G. 1999. Model-aided remote usability evaluation. In A. Sasse and C. Johnson, Eds., *Proceedings of the IFIP TC13 Seventh International Conference on Human- Computer Interaction* (Edinburgh, Scotland, August), pp. 434–442. Amsterdam, The Netherlands: IOS Press.

Paternò, F., Mancini, C., and Meniconi, S. 1997. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In *Proceedings of the IFIP Tc13 interantional Conference on Human-Computer interaction* (July 14 - 18, 1997). S. Howard, J. Hammond, and G. Lindgaard, Eds. IFIP Conference Proceedings, vol. 96. Chapman & Hall Ltd., London, UK, 362-369.

Rauterberg, M. 1995. From novice to expert decision behaviour: A qualitative modeling approach with Petri nets. In Y. Anzai, K. Ogawa, and H. Mori, Eds., Symbiosis of Human and Artifact: Human and Social Aspects of Human-Computer Interaction, Volume 20B of *Advances in Human Factors/Ergonomics* (1995), pp. 449–454. Amsterdam, The Netherlands: Elsevier Science Publishers.

Roy, M.C., Dewit,O., Aubert, B.A. 2001. The impact of interface usability on trust in Web retailers. *Internet Research*, Vol. 11 Iss: 5, pp.388 - 398, Emerald Group Publishing Limited.

Sackett, G. P. 1978. *Observing Behavior*, Vol. 2. University Park Press, Baltimore, MD.

Sanderson, P. M. Fisher, C. 1994. Exploratory sequential data analysis: foundations. Human-Computer Interaction. *Special Issue on ESDA, 9.*

Sanderson, P. M., Scott, J. J. P, Johnston, T., Mainzer, J., Watanabe, L. M., James, J. M. 1994. Mac-SHAPA and the enterprise of Exploratory Sequential Data Analysis (ESDA). *International J. of Hum.–Comput. Studies, 41.*

Santana, V. F., Baranauskas, M. C. C. 2008. A Prospect of Websites Evaluation Tools Based on Event Logs. In*: IFIP 20th World Computer Congress, 1st Human-Computer Interaction Symposium (HCIS 2008)*, 2008, Milão. Human-Computer Interaction Symposium. Boston : Springer, 2008. v. 272. p. 99-104.

Santos, P. J. Badre, A. N. 1994. Automatic chunk detection in human-computer interaction. In *Proceedings of Workshop on Advanced Visual Interfaces AVI '94*.

Scholtz, J., Laskowski, S. 1998. Developing usability tools and techniques for designing and testing web sites. In *Proceedings of the Fourth Conference on Human Factors & the Web*, Basking Ridge, NJ, Jun, 1998.

Schwerz, A.L. Morandini M.,Silva, S. R. 2007. A Task Model Proposal for Web Sites Usability Evaluation for the ErgoMonitor Environment. Human-computer Interaction. *Interaction design and usability. Lecture Notes in Computer Science*, 2007, Volume 4550/2007, 1188-1197, DOI: 10.1007/978-3-540-73105-4_129

Setubal, J., Meidanis, J. 1997. *Introduction to Computational Molecular Biology*. PWS Publishing Company.

Shahabi C., F. Banaei-Kashani, J. Faruque. 2003. Efficient and Anonymous Web Usage Mining for Web Personalization. In *INFORMS Journal on Computing*, Vol 15, Nº 2, Spring 2003, PP. 123-147

Sherman P. 2006. *Usability Success Stories - How Organizations Improve By Making Easier-To-Use Software and Web Sites*. Gower Publishing Limited, England.

Siochi, A. C. Hix, D. 1991. A study of computer-supported user interface evaluation using maximal repeating pattern analysis. In *Proceedings of CHI '91*.

Spiliopoulou, M. 2000. Web usage mining for Web site evaluation. *Commun. ACM* 43, 8 (Aug. 2000), 127-134.

Spiliopoulou, M, Faulstich, L. 1998. WUM: A tool for Web utilization analysis. *In Proc. 6th Int'l Conf. on Extending Database Technology (EDBT'98),* Valencia, Spain, March 1998.

Tiedtke, T., Märtin, C., Gerth, N. 2002. AWUSA – A Tool for Automated Website Analysis. In: *PreProceedings of the 9th Int. Workshop DSV-IS 2002*, Rostock, Germany, pp. 251–266, 2002

Thomas, J.C.; Kellogg, W.A. 1989. Minimizing ecological gaps in interface *design*, *Software, IEEE* , vol.6, no.1, pp.78-86, Jan 1989

Thompson, K. E., Rozanski, E. P., and Haake, A. R. 2004. Here, there, anywhere: remote usability testing that works. In *Proceedings of the 5th Conference on information Technology Education* (Salt Lake City, UT, USA, October 28 - 30, 2004). CITC5 '04. ACM, New York, NY, 132-137.

Uehling, D. L., Wolf, K. 1995. User Action Graphing Effort (UsAGE). *In Proceedings of CHI '95.*

Vargas, A., Weffers, H. T. G., Rocha, H. V. A Method for Remote and Semi-Automatic Usability Evaluation of Web- based Applications Through Users Behavior Analysis. In: *Proc. of the Measuring Behavior 2010 – 7th International Conference on Methods and Techniques in Behavioral Research*, Eindhoven, The Netherlands, 2010.

Veiga, G.M., Ferreira, D.R. 2009. Understanding spaghetti models with sequence clustering for ProM'. *Proc. Workshop on Business Process Intelligence*, Germany, 2009, pp. 92–103

W3C. 2011. World Wide Web Consortium. Disponível em: <http://www.w3.org>. Acesso em novembro de 2011.

West, R., Lehman, K. 2006. Automated summative usability studies: an empirical evaluation. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*(CHI '06), Rebecca Grinter, Thomas Rodden, Paul Aoki, Ed Cutrell, Robin Jeffries, and Gary Olson (Eds.). ACM, New York, NY, USA, 631-639. DOI=10.1145/1124772.1124867 http://doi.acm.org/10.1145/1124772.1124867

Wu, K., Yu, P.S. and Ballman, A., "Speedtracer: A web usage mining and analysis tool", *IBM Systems Journal*, 37 (1), 1998, 89-105.