Identificação de Padrões de Utilização da Web Mediada por Tecnologias Assistivas

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Vagner Figuerêdo de Santana e aprovada pela Banca Examinadora.

Campinas, 08 de abril de 2009.

Profª. Drª. Maria Cecília Calani Baranauskas
Instituto de Computação - UNICAMP
(Orientadora)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Maria Fabiana Bezerra Müller – CRB8 / 6162

Santana, Vagner Figuerêdo de
Sa59i Identificação de padrões de utilização da Web mediada por tecnologias assistivas/Vagner Figuerêdo de Santana. -- Campinas, [S.P. : s.n.], 2009.

Orientador : Maria Cecília Calani Baranauskas.
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1.Interação humano-computador. 2.Desenho universal. 3.Acessibilidade. 4.Sites da web - Desenvolvimento. I. Baranauskas, Maria Cecília Calani. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

(mf/bm/imecc)

Título em inglês: Patterns indentification of Web usage mediated by assistive technologies

Palavras-chave em inglês (Keywords): 1. Human-computer interaction 2.Universal design. 3.Accessibility. 4.Web site design.

Área de concentração: Avaliação de interfaces de usuário

Titulação: Mestre em Ciência da Computação

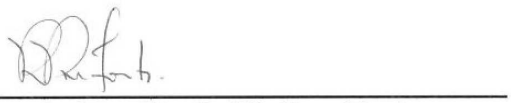
Banca examinadora: Profa. Dra. Maria Cecília Calani Baranauskas
Profa. Dra. Renata Pontin de Mattos Fortes
Prof. Dr. Hans Kurt Edmund Liesenberg

Data da defesa: 08 de abril de 2009

Programa de Pós-Graduação: Mestrado em Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 08 de abril de 2009, pela Banca examinadora composta pelos Professores Doutores:



Profª. Drª. Renata Pontin de Mattos Fortes
ICMC / USP.

Prof. Dr. Hans Kurt Edmund Liesenberg
IC / UNICAMP.

Profª. Drª. Maria Cecilia Calani Baranauskas
IC / UNICAMP.

Instituto de Computação
Universidade Estadual de Campinas

Identificação de Padrões de Utilização da Web Mediada por Tecnologias Assistivas

Vagner Figuerêdo de Santana[1]

Abril de 2009

Banca Examinadora:

- Prof[a]. Dr[a]. Maria Cecília Calani Baranauskas
  Instituto de Computação - UNICAMP (Orientadora)
- Prof[a]. Dr[a]. Renata Pontin de Mattos Fortes
  Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo
- Prof. Dr. Hans Kurt Edmund Liesenberg
  Instituto de Computação - UNICAMP
- Prof. Dr. Ismar Frango Silveira (Suplente)
  Faculdade de Computação e Informática - Universidade Presbiteriana Mackenzie
- Prof. Dr. Rogério Drummond Burnier Pessoa de Mello Filho (Suplente)
  Instituto de Computação - UNICAMP

[1]Suporte financeiro do projeto PROESP/CAPES

Resumo

A Web conta com dezenas de milhões de *websites*, mas poucos deles estão em conformidade com requisitos simples de acessibilidade, como utilizar tamanhos relativos ou descrever elementos gráficos, o que pode indicar problemas de design de Interface de Usuário (IU). Para se identificar este tipo de problema utiliza-se avaliação do código de *websites*. No entanto, outros problemas surgem apenas durante a interação do usuário com a IU. Sem considerar dados resultantes do uso, problemas de usabilidade e/ou barreiras de acessibilidade podem permanecer desconhecidos. Portanto, identificar como usuários interagem com IUs é uma forma de detectar problemas de design e descobrir maneiras de utilização diferentes das previstas durante o projeto de IU.

Entre as maneiras de capturar a interação do usuário com IUs estão a utilização de vídeos, captura dos movimentos dos olhos do usuário, etc. Na Web, alguns métodos são menos indicados que outros devido à necessidade de utilização de componentes específicos de hardware e/ou software, o que pode reduzir a participação do público alvo e reduzir a representatividade de uma avaliação. Para se conhecer como usuários utilizam a IU na Web, é possível utilizar *logs* capturados no lado do servidor e no lado do cliente. Este último é o que possibilita a obtenção de mais detalhes e, conseqüentemente, reconstrução das ações do usuário em maior granularidade do que apenas identificar quais páginas o usuário visitou. Assim, pequenas diferenças durante a interação podem ser identificadas e, conseqüentemente, auxiliar especialistas na identificação de padrões de utilização e possíveis problemas de design.

Dado o volume de dados resultante da captura de eventos no lado do cliente, a utilização de ferramentas automáticas se faz necessária, seja para captura ou para extração de informações dos *logs*. Este trabalho apresenta requisitos para ferramentas de avaliação de *websites* baseadas em *logs* de eventos. Ainda, com base nestes requisitos, define um modelo de captura assíncrona de eventos para auxiliar desenvolvedores de ferramentas de avaliação de *websites*. Assim, requisitos e modelo de captura constituem um ponto de partida para desenvolvedores que desejam capturar *logs* de eventos disparados no dispositivo utilizado pelo cliente, apoiando a captura de dados de utilização real.

Outra contribuição apresentada é a especificação de um modelo para identificar padrões

de utilização da Web mediada por tecnologias assistivas. A implementação deste modelo demonstra a viabilidade de se capturar dados durante a utilização real, envolvendo usuários de tecnologias assistivas, em seus ambientes cotidianos de utilização da Web. Um grafo de utilização resulta desse modelo. Ele é composto por fluxos de uso e caminhos mais comuns de uma página avaliada, assim como representações visuais de diferenças significativas nesses fluxos. O modelo também propõe uma técnica de como indicar possíveis problemas de design nas páginas Web avaliadas, auxiliando na tarefa de avaliar *websites*, tendo como foco principal a identificação de padrões de utilização de usuários de tecnologias assistivas.

Abstract

The Web has millions of websites, but few of them are in accordance with simple accessibility requirements such as the use of relative sizes or description of graphic elements, that can represent User Interface (UI) design problems. To identify this kind of problem the data source usually used is the Web page's code. However, other problems may appear only when the user interacts with the UI. Moreover, without considering usage data, usability problems and/or accessibility barriers can stay unknown. Thus, identifying how users interact with UIs is a way of detecting design problems and discovering uses different from the ones defined during the UI Project phase.

There are different approaches to capturing user interaction with UIs as video capture, users' eyes movements, etc. In the Web, some methods are less appropriated than others due to the need for using specific components of hardware and/or software, which can reduce the participation of the target audience and reduce how meaningful an evaluation is. To know how users use the UI in the Web, it is possible to use logs captured at server-side and at client-side. The data that can be captured at client-side allows obtaining more details and, consequently, rebuild user's actions in a greater granularity than the identification of the pages that a user had visited. Then, small differences during the interaction can be identified and may help specialists in the task of identifying usage patterns and possible design problems.

Due to the amount of data resulted from the event capture at client-side, the use of automatic tools became necessary for the capture or the extraction of information from logs. This work presents requirements for website evaluation tools based on event logs. In addition, according to these requirements, defines an asynchronous event capture model to help developers of website evaluation tools. Therefore, requirements and capture model represent a starting point for projects involving the development of tools to capture event logs triggered at client's device, supporting the data capture during real use.

Other contribution is the specification of a model to identify patterns of Web usage mediated by assistive technologies. The development of this model shows the viability of capturing data during real use, involving assistive technology users, in their common use environment. A usage graph results from the model. It is constituted by usage flows and

common walks of an evaluated Web page, as well as the visual representation of significant differences in these flows. The model also proposes a technique for identifying possible design problems in evaluated Web pages, helping in the task of evaluating Web pages, aiming at the identification of usage patterns of assistive technology users.

Agradecimentos

Gostaria de agradecer à Prof^a. Dr^a. Maria Cecília Calani Baranauskas pela orientação sempre correta, dando autonomia, sem deixar de ser atenciosa.

À minha família pelo apoio incondicional.

Aos componentes dos grupos Todos Nós, InterHAD e e-Cidadania, e aos colegas de estudo da sala 80 do Instituto de Computação pelo compartilhamento fervoroso de conhecimento enquanto estudávamos para as disciplinas de Teoria dos Grafos e Computação Distribuída.

A todos os usuários que participaram anonimamente da avaliação.

Finalmente, agradecer ao PROESP/CAPES pelo apoio financeiro ao projeto.

Sumário

Capítulo 1

Introdução

A Web vem, cada vez mais, apoiando serviços de diversas áreas e está se tornando parte essencial da vida das pessoas na sociedade contemporânea. Alguns exemplos que impulsionam esse crescimento são o suporte à comunicação, possibilidades de entretenimento e a agilidade agregada a serviços de governo. Assim, buscar que o máximo de pessoas tenha autonomia para utilizar e contribuir para esses serviços é fundamental.

A Internet conta com cerca de 120 milhões de *websites* [13]. Destes, mais de 90% falham ao serem confrontados com padrões mínimos de acessibilidade como fornecer descrições adequadas para elementos visuais [61], o que revela que há um longo caminho a ser percorrido para tornar a Web um ambiente sem barreiras para os usuários.

A adequação de *websites* a requisitos de acessibilidade e usabilidade é uma maneira promissora de alcançar esse objetivo e remete à avaliação da interface de usuário (IU), muitas vezes apoiada por ferramentas de avaliação. Existem dois grandes grupos de tais ferramentas: as que usam o código das páginas Web como fonte de dados e as que usam dados resultantes da utilização (i.e., *logs*).

Os *logs* utilizados por ferramentas de avaliação podem ser capturados no lado do servidor ou no lado do cliente. A captura de *logs* no lado do servidor é tecnicamente mais simples, mas os dados revelam apenas informações relacionadas às páginas que os usuários visitaram. Em contrapartida, a captura de *logs* no lado do cliente é mais complexa e envolve mais tarefas, mas, possibilita a obtenção de informações mais detalhadas de como usuários interagem com cada elemento da página Web.

As ferramentas que usam o código de páginas Web ou que usam *logs* capturados no lado do servidor são fundamentais para avaliações de IU. No entanto, devido à heterogeneidade de hardware e software utilizados para acessar *websites*, a diversidade de perfis de usuário e os diversos contextos de uso possíveis, a utilização de ferramentas que usam dados de utilização capturados no lado do cliente complementa outros tipos de avaliações e pode revelar como os usuários interagem com a IU mais detalhadamente. Outro ponto positivo

em relação a essa abordagem é a possibilidade de avaliar como usuários de tecnologias assistivas interagem enquanto estão em ambientes já conhecidos, com configurações de software e hardware usadas comumente.

1.1 Contexto e objetivo

Este trabalho surgiu da identificação da crescente demanda por ferramentas e ambientes que apóiem a tarefa de tornar *websites* mais acessíveis e mais usáveis. Essa demanda é impulsionada por diversos fatores, alguns deles são: o retorno financeiro obtido por *websites* que oferecem serviços com boa usabilidade [30] e a legislação sobre acessibilidade. No Brasil, por exemplo, o Decreto 5.296/04 aponta que estabelecimentos de ensino de qualquer nível deverão proporcionar condições de acesso e utilização de todos os seus ambientes para pessoas com deficiência ou com mobilidade reduzida [25].

Uma das frentes de trabalho é apoiar a adequação de *websites* e correção de problemas de codificação, integrando diferentes tecnologias da Web para alcançar código válido, usável e acessível. Com este propósito foi desenvolvido o WARAU[1] (*Websites* Atendendo a Requisitos de Acessibilidade e Usabilidade). Ele é um ambiente de apoio para mantenedores de *websites* que desejam adequar seus *websites* a requisitos de acessibilidade e usabilidade. Ele também fez parte do projeto PROESP/CAPES, é uma das ações do grupo Todos Nós e está detalhado, juntamente com uma contextualização mais abrangente sobre tópicos e iniciativas voltadas à promoção da acessibilidade na Web, no apêndice C. Outra frente de trabalho é avaliar a utilização real de *websites*, foco principal deste trabalho.

Com base no levantamento bibliográfico foi identificado um número ainda pequeno de propostas de ferramentas que apóiam a captura de dados no lado do cliente durante a utilização real.

Neste contexto, o objetivo deste trabalho é contribuir para a avaliação de interfaces Web para que estas sejam mais acessíveis e usáveis por todos, indiscriminadamente. Para isso foi proposta e desenvolvida uma ferramenta que usa *logs* de eventos do lado do cliente para análise de padrões de utilização, tendo como foco principal a identificação de fluxos de eventos que podem indicar barreiras para usuários de tecnologias assistivas e que seriam difíceis de obter em testes presenciais, seja pela dificuldade de encontrar usuários totalmente representativos (i.e., usuários que utilizam o serviço avaliado) quanto pela utilização de ambientes e tarefas não naturais.

[1] `http://warau.nied.unicamp.br`

1.2 Método

O levantamento bibliográfico inicial reuniu diversas características e estratégias ainda não utilizadas nas ferramentas de avaliação existentes. Então, para guiar o desenvolvimento do modelo proposto e sua implementação, utilizou-se como referencial teórico a Semiótica Organizacional.

Semiótica Organizacional é uma disciplina que lida com informação e sistemas de informação de uma maneira que leva em consideração tanto questões técnicas quanto aspectos sociais e humanos [75].

A Semiótica Organizacional conta com um conjunto de métodos chamado MEASUR (*Methods for Eliciting, Analysing and Specifying Users' Requirements*), que pode ser utilizado por pesquisadores para o entendimento, desenvolvimento, gerenciamento e uso de sistemas de informação [58].

O MEASUR conta com 5 métodos principais, são eles: *Problem Articulation Method* (PAM), que auxilia na clarificação do problema tratado; *Semantic Analysis Method* (SAM), que apóia o levantamento e representação de requisitos; *Norm Analysis Method* (NAM), que fornece meios para especificar padrões gerais de comportamento dos agentes dentro do sistema; *Comunication and Control Analysis*, que auxilia na análise de toda comunicação ocorrida entre todos os agentes e sistemas identificados no PAM; *Meta-Systems Analysis*, que apóia a solução de meta-problemas no planejamento e gerenciamento do projeto [58].

Neste trabalho, o artefato do MEASUR utilizado foi a Escada Semiótica. Ela apóia a análise de sistemas da informação em seis diferences degraus, contribuindo para a clarificação do que é necessário produzir para que um sistema resolva não somente os problemas relacionados à plataforma de Tecnologia da Informação (TI), mas também considere aspectos sociais da utilização deste sistema [74]. Ela foi usada para organizar as características e definir os requisitos para ferramentas de avaliação baseadas em *logs* de eventos.

Uma vez levantados os requisitos, o foco passou a ser a modelagem do sistema. Para tanto foram definidos os casos de uso (Apêndice A) e o diagrama de casos de uso (Apêndice B), modelagem do sistema guiado pelo paradigma de Programação Orientada a Objetos (POO), utilizando técnicas apresentadas em [22] para simular propriedades básicas de POO em Javascript. Adicionalmente, a modelagem se baseou em padrões de projeto tratados em [38] e a comunicação entre os objetos do sistema utilizou o padrão Model-View-Controler, conforme proposto em [10] para integrar tecnologias Java para Web, como Java Server Pages e Servlets.

1.3 Contribuições e organização

As principais contribuições deste trabalho são:

– Definição dos requisitos para ferramentas de avaliação de *websites* baseadas em *logs* de eventos, apresentados em uma instância da Escada Semiótica;

– Especificação e implementação de um modelo para identificar padrões de utilização da Web mediada por tecnologias assistivas;

– Demonstração da viabilidade de se capturar dados durante a utilização real, envolvendo usuários de tecnologias assistivas, em seus ambientes cotidianos de utilização.

Os capítulos seguintes apresentam as diferentes etapas do desenvolvimento da ferramenta WELFIT[2] (*Web Event Logger and Flow Identification Tool*), produto principal deste trabalho. Cada capítulo aborda uma etapa do trabalho e contém texto integral de relatórios técnicos ou artigos, publicados ou submetidos para conferências internacionais. O encadeamento dos trabalhos se dá da seguinte maneira: o capítulo 2 apresenta o levantamento de requisitos para ferramentas de avaliação baseada em *logs* de eventos; o capítulo 3 conta com a definição e detalhes de implementação do modelo de captura usado para registrar eventos no lado do cliente, seguindo requisitos elicitados no capítulo 2; o capítulo 4 traz uma forma de representar o volume de eventos capturados pelo *data-logger* especificado no capítulo 3, de forma a sumarizar os dados capturados durante várias sessões de utilização; o capítulo 5, por sua vez, apresenta os componentes definidos nos capítulos 2, 3 e 4, define o modelo de identificação de padrões de utilização e apresenta sua implementação na ferramenta chamada WELFIT. A seguir serão apresentados os capítulos e seus respectivos locais de publicação, no caso de trabalhos já publicados:

- Capítulo 2: "*A Prospect of Websites Evaluation Tools Based on Event Logs*"[3] (2008), Vagner Figuerêdo de Santana e Maria Cecília Calani Baranauskas. *Human-Computer Interaction Symposium - 20th IFIP World Computer Congress*, Milão, Itália, Springer Boston, pp. 99-104.
  `http://dx.doi.org/10.1007/978-0-387-09678-0_9`

  A partir da variedade de ferramentas de avaliação de *websites* baseadas em *logs* existente, são apontadas algumas de suas limitações (e.g., dependência de modelo de tarefas, dependência de plug-ins, utilização de tarefas simuladas, separação de

[2] `http://argos.nied.unicamp.br:8888/welfit/`

[3] Human-Computer Interaction Symposium, Volume 272/2008, 2008, pages 99-104, A Prospect of Websites Evaluation Tools Based on Event Logs, Vagner Figuerêdo de Santana e Maria Cecília Calani Baranauskas.

acessibilidade de usabilidade, etc). Algumas destas características resultam em sistemas fortemente acoplados com outros componentes de software e tornam a configuração e uso mais custosos. Assim, o capítulo levantou pontos fortes e fracos das ferramentas estudadas. Depois, essas características foram reunidas de acordo com seus assuntos, resultando em 8 aspectos mais concisos e utilizados para agrupar as características de soluções, limitações e lacunas das ferramentas estudadas. Com este levantamento foi possível definir requisitos para ferramentas similares em uma instância da Escada Semiótica, auxiliando desenvolvedores a reutilizar idéias consolidadas e evitar pontos fracos já identificados.

- Capítulo 3: "*An Asynchronous Client-Side Event Logger Model*" (2008), Vagner Figuerêdo de Santana e Maria Cecília Calani Baranauskas. Relatório técnico IC-08-28, publicado no Instituto de Computação, UNICAMP. Derivado deste relatório técnico há o trabalho intitulado "*KEEPING TRACK OF HOW USERS USE CLIENT DEVICES: An Asynchronous Client-Side Event Logger Model*", que foi aceito para apresentação no formato de pôster e publicação no formato de quatro páginas no *11th International Conference on Enterprise Information Systems* (ICEIS 2009).

  Estudos sobre utilização de *websites* normalmente consideram *logs* de servidor como fonte de dados para identificar padrões de uso. Esta solução apresenta limitações quando o objetivo é representar como usuários interagem com elementos de IU específicos, uma vez que esta abordagem pode não conter informações detalhadas sobre o que os usuários fazem quando usam uma página Web. Este capítulo apresenta um modelo para capturar *logs* de eventos no lado do cliente e uma implementação como o módulo cliente da ferramenta chamada WELFIT. O *data-logger* especificado conta com componentes responsáveis por endereçar os requisitos definidos no capítulo 2. Estes componentes endereçam tarefas levantadas nos requisitos que a ferramenta segue. Exemplos dessas tarefas são: identificar sessões, compactar os *logs* de eventos usando uma técnica eficiente, conectar com o servidor onde os dados serão persistidos, etc. Através do modelo apresentado, sistemas de mineração podem capturar dados de uso da Web mais detalhados e diferentes implementações do modelo podem auxiliar especialistas de Interação Humano-Computador (IHC) a registrar eventos de diversos dispositivos como celulares, controles remotos, *set-top boxes*, etc. Por fim, a implementação do modelo definido foi eficaz e eficiente em relação aos requisitos que guiaram seu desenvolvimento.

- Capítulo 4: "*Revealing Usage Patterns of Web Pages*" (2009), Vagner Figuerêdo de Santana e Maria Cecília Calani Baranauskas. Artigo submetido para conferência internacional.

  A partir do grande volume de dados obtidos ao capturar eventos no lado do cliente,

é possível obter padrões de uso de páginas Web. Assim, o desafio de ferramentas de avaliação que utilizam esse tipo de fonte de dados passa a ser processar e lidar com esse grande volume de dados, uma vez que tarefas simples de poucos minutos podem resultar em uma seqüência de centenas de eventos. Assim, este capítulo apresenta uma técnica para processar e construir o grafo de utilização de páginas Web e como apresentá-lo visualmente, sumarizando informações estatísticas sobre *logs* de eventos de diversas sessões de utilização. Assim, a estrutura proporcionada pelo grafo de utilização apóia tanto a análise dos dados de utilização tanto computacionalmente quanto graficamente. Portanto, com representações e gráficos de padrões de utilização, especialistas de IHC podem identificar problemas de interface e especialistas em mineração podem reusá-los para aplicar diferentes técnicas estatísticas para extrair novas informações.

- Capítulo 5: "*WELFIT: A Web Event Logger and Flow Identification Tool of Client-Side Logs*" (2009), Vagner Figuerêdo de Santana e Maria Cecília Calani Baranauskas. Artigo submetido para conferência internacional.

  A partir da variedade de fontes de dados utilizadas por ferramentas de avaliação de *websites* (e.g., páginas Web, *logs* de servidor, movimentos do mouse) é possível identificar que poucas permitem avaliação remota utilizando dados representando a utilização real em uma granularidade maior que o nível de *page-view* (Tabela 1.1). Complementarmente, não considerar dados que representam a interação real do usuário com a IU pode deixar de revelar problemas de usabilidade e/ou barreiras de acessibilidade.

  Este capítulo apresenta o WELFIT, um sistema que:

  - Segue os requisitos definidos no capítulo 2;
  - Captura eventos conforme *data-logger* especificado no capítulo 3;
  - Utiliza a técnica de sumarização e representação dos *logs* de utilização apresentada no capítulo 4;
  - Especifica e apresenta uma implementação do modelo proposto com o objetivo de identificar padrões de utilização baseado em *logs* de eventos.

  O sistema depende apenas do suporte à tecnologia utilizada para desenvolver o módulo cliente, neste caso, Javascript, e da aceitação do usuário em participar da avaliação. Com o modelo apresentado é possível capturar dados de utilização de *websites* durante a utilização real, identificando fluxos de utilização e caminhos comumente percorridos, e representar visualmente diferenças significativas nestes

Tabela 1.1: Tipos de avaliação e fonte de dados usadas em trabalhos correlatos.

	Ferramenta	**Avaliação remota**	**Fonte de dados**
Captura e avaliação de *logs*	WebVIP (1998)	Não	Cliente
	WET (1999)	Não	Cliente
	WebQuilt (2001)	Sim	Servidor
	WebRemUSINE (2002)	Sim	Cliente
	MouseTrack (2006)	Sim	Cliente
Avaliação de código de páginas Web	EvalIris (2004)	Sim	Páginas Web
	NAUTICUS (2006)	Sim	Páginas Web
	MAGENTA (2006)	Sim	Páginas Web
Mineração de *logs* de servidor	Web Utilization Miner (1999)	Sim	Servidor
	WebSIFT (2000)	Sim	Servidor
	LumberJack (2002)	Sim	Servidor
	DCW (2007)	Sim	Servidor

caminhos. Com base nessas representações, a ferramenta auxilia especialistas na identificação de problemas de design nas páginas avaliadas.

Cada capítulo apresenta uma parte da pesquisa; trabalhos que serviram de base são referenciados nos capítulos específicos. A quantidade de dados capturados e mencionados ao longo do trabalho aumenta de acordo com a etapa em que cada artigo ou relatório técnico foi escrito. Finalmente, o modelo e todas as técnicas e métodos utilizados amadureceram durante a pesquisa, portanto, quando são definidos em mais de um artigo, ganharam especificações mais maduras e informações mais recentes sobre a pesquisa, no que diz respeito ao grau de detalhamento.

Capítulo 2

A Prospect of Websites Evaluation Tools Based on Event Logs

2.1 Introduction

In the last years many researchers have been working on automatic and remote usability evaluation of user interfaces focusing on tests with more users without resulting in costly evaluations [49]. The total or partial use of automatic usability evaluation methods can reduce time and costs involved in the development of a Web application, as it liberates specialists from repetitive tasks as manual log analysis [63]. While researchers have focused on usability issues, less has been done specifically to increase the accessibility level of the applications. In addition, the integration of Accessibility and Usability (A&U) concepts is not usual in user interface (UI) evaluation tools.

One of the aspects that make interesting the approach of automatic evaluation is the remote evaluation during real use, and involving users in their regular environments. In addition, remote analysis of A&U makes unnecessary the presence of participants in the test places, which could be a barrier. Moreover, tests in controlled environments are artificial and may influence the results [65]. Thus, the UI evaluation in real situations points to the use of event logs, since it makes possible to capture data while users work on their regular tasks, in their usual environments [39].

Interface events have a duration ranging from 10 milliseconds to one second [43], showing that the number of events that occur during few minutes simple tasks can be huge. Thus, due to the typical amount of interface events, the use of automatic evaluation tools is usually necessary so that information extracted from the events reach a level of abstraction that is useful to specialists [43]. This paper identifies solutions, limitations and gaps of website evaluation tools based on event logs; it is organized as follows: section 2.2 presents ideas well succeeded and the gaps identified; section 2.3 discusses the results

of this work; section 2.4 presents conclusions and points to new directions.

2.2 Event logs and websites evaluation: identified solutions, limitations, and gaps

UI events are natural results of using windows based interfaces and their components (e.g., key strokes, mouse clicks) [43]. Since it's possible to record these events and they indicate the user behavior during the interface usage, they represent an important source of information regarding usability. Event logs produce results as frequency of use of certain functions, places where users spend more time and the sequence they complete tasks [81].

According to Hong *et al.* [44], the goal of evaluation software based on event logs is to use a capture method that is easy to apply to any website, and that is compatible with various operating systems and Web browsers. This section summarizes the main issues present in the following tools: WET [37], WebRemUSINE [63], and WebQuilt [44]. Also, we combine some ideas with other works related to the evaluation of websites as, for example, NAUTICUS [21], a tool that integrates A&U and points to a promising direction for new tools. Our analysis is organized according to eight aspects: the configuration needed to use the tools, the capture mechanisms, the ways logs are stored and transmitted, the dependence on users' actions, the changes needed at the UI, scalability, levels of abstraction, and integration of A&U. We present the solutions, limitations we found, and gaps to be investigated.

1. Regarding the configuration of the environment, i.e. the set of tasks needed for a tool to be used to evaluate a website, we can point out:

    - **Solutions** – Events capture on the client-side at WET depends only on the native script languages of the newer browsers (e.g., Javascript), avoiding the need of plug-ins installation. Other solutions that make possible to capture across different websites without requiring any change on evaluated Web pages are the use of a proxy logger to capture server-side data [44], or the use of specific browser to capture client-side data [17].
    - **Limitations** – Require maintainers of the evaluated website to change their Web pages so that the data capture starts, as in [63, 37]; or that files must be hosted on the server of the evaluated websites [37]. Other important points are the dependency of task models [63] and dependency of specific software installed, as in [63, 17].

- **Gap** – Events capture at client-side across different websites without requiring any change in the evaluated web pages or using specific browsers.

2. Regarding server-side vs. client-side capture, server-side capture concerns the use of logs generated only on the server, while client-side capture refers to data obtained through the recording of information available only on the client's device.

   - **Solutions** – Use of events triggered at client-side interface, because there are more detailed data sources of how the user uses a Web site [63, 37]. Also, the use of a proxy logger to record all communication between the client's browser and the visited Web pages, without requiring any change in the evaluated pages or any plug-in installed in the client's device [44].
   - **Limitations** – The limitations related to the data capture deals with the storage of data too, because the amount of data obtained through events on the client-side is representative and the space available to store them in client's device is restricted, as in [37]. On the other hand, when there is space available on the client-side, the problem arises from the time spent on the log transmission at the end of a test session, as can occur in [63]. An alternative would be to use only server-side data, but it can lead to less accurate results [17].
   - **Gap** – The combination of server-side data and the client-side data, since server data may indicate events not present on the client-side (e.g., response times of accessing a link). On the other hand, client-side data have more detailed information about the user route in a Web page.

3. Storing and transmitting logs refers to strategies used to record event logs, including the location, data storage capacity, and the transfer methods to a server.

   - **Solutions** – Use of a proxy logger that captures the data about client requests and server responses [44]. Thus, the data storage capacity depends on the tool server capacity and it's not necessary to send data to another element. To avoid the limit of available space in cookies, some components (e.g., Java applets) can be used to store and transmit client-side events [63].
   - **Limitations** – When recording data on a participant machine, the tool becomes dependent on the available space in that device, as in [37]. Moreover, the transmission of such data at the end of a test session, as in [63, 37], may require too much bandwidth connection, and thus interfere with the use of the interface. Another identified limitation deals with the use of software components that are plug-ins dependent (e.g., Java applets), as in [63]. This can be

an obstacle, since the user may not have knowledge to install the plug-in or doesn't have a broadband connection to download it quickly.

- **Gap** – The use of cyclic transmission of event logs from client-side to the server, avoiding the limit of available space in cookies and the time required to transfer captured data in batch at the end of a test session.

4. By dependence on user actions we mean what is necessary from the participants for the tool to start the data capture (e.g., acceptance of the use).

   - **Solution** – The user accesses the first URL from the main page of the tool and then no further action is required until the end of the test session [44].
   - **Limitation** – Interfere the common use of the interface, requiring that the user select a task from a list or select options to start and stop the capture every test session. This may call the user's attention or even not allow that the data between these actions be captured, as in [63, 37].
   - **Gap** – An alternative to requiring a user action at the beginning of each test session is to make the acceptance of a test to be valid for several sessions, until the user chooses to stop the data capture.

5. Regarding the impacts that the evaluated UI suffers when a tool is in use, the key metric used is how the presentation of the tool interfere with the use of the website under evaluation.

   - **Solutions** – Display options to indicate the beginning and the end of capture [37] or using a Web page, at the beginning of each session [44].
   - **Limitations** – The interface changes concern the reduction of the useful space for the UI to display any component of the evaluation tool, as occurs in the presentation of the list of tasks in [63]. Another issue is the change of URLs used in the pages processed by the tool, as in [44].
   - **Gap** – Ensure that the tools' controls are always visible, don't compete with other UI elements, and don't reduce the useful area of the Web page.

6. Scalability refers to how the evaluation tools deal with many evaluation sessions or long test sessions.

   - **Solutions** – The use of specific software (e.g., Web browser, applet Java) that can access participant's file system to record logs and then avoid data storage capacity of cookies, as in [63]. Or use of a proxy logger to keep track of user requested data [44], depending only on the tool's server.

- **Limitation** – The biggest barrier found was the space available for storage of logs in cookies, as it is not possible to record all usage, it forces the events captured to be a reduced sub set of existing events, as in [37].

- **Gaps** – Avoid limits of cookies without making the tool plug-ins dependent and avoid to transfer logged data only at the end the tests sessions.

7. As concerns the strategies used to achieve higher levels of abstraction from logs of events (e.g., implicit interest, use of grammars to relate events).

   - **Solutions** – Using task models and grammars that enable events of Web pages to be represented as higher levels actions [63] and conversion of low level events into sequences of actions [44].

   - **Limitation** – Dependency of specific task models and/or grammars to reach higher levels of abstraction, as in [63].

   - **Gap** – The use of methods that do not depend on task models to obtain higher level information. A possible alternative to these models is the sequence characterization based on the Markov Chains [43].

8. Finally, the integration of A&U is necessary, because if developers evaluate them separately, then some problems may appear regarding different priorities for guidelines interfering in the way the target audience uses an interface.

   - **Solutions** – Development and use of criteria covering aspects of A&U and evaluation of structure and content of pages [21].

   - **Limitation** – The identified limitation found in NAUTICUS is the dependence of a dictionary of terms commonly used in bad structured pages.

   - **Gap** – The integration of A&U was not found in websites evaluation tools based on logs, thus becoming the main gap of the studied tools.

2.3 Discussion

In this study we could identify characteristics that can strengthen evaluation tools, making them more robust, less costly, and easier to use and reuse. The integration of A&U has been identified as the main gap in the analyzed tools. We propose requirements that a logs based websites evaluation tool should have (Table 2.1). The requirements are organized in the Semiotic Ladder (SL), an artifact of Organizational Semiotics [73] which provides a framework for analysis of information systems under six different layers. The SL structure allows the clarification of concepts and the analysis of information, considering from the Information Technology (IT) Platform (i.e., physical layer, empirical layer, and syntactic layer) to the Human information Functions (i.e., semantic layer, pragmatic layer, and social layer).

Analysis shows that the tools studied meet, mainly, the requirements associated with layers related to IT Platform (i.e., physical, empirical, and syntactic layers). The requirements of the Human Information Functions (i.e., semantic, pragmatic, and social layers) are hardly addressed. This suggests that new tools must take them into account to achieve goals related to these layers.

Table 2.1: Requirements for websites evaluation tools based on event logs instantiated in the Semiotic Ladder

Human Information Functions	
Social	Focus on the integration of accessibility and usability for the target audience of the evaluated website. Enabling remote testing during real use of the evaluated website. Interfere with the Web page as minimum as possible.
Pragmatic	Providing controls representing the status of the tool and user context during the test session. The operation of the tool should use two actions: one to start the capture, which can stay valid for more than one session, another to interrupt the capture, which may occur at any time.
Semantic	Providing high levels of abstraction without depending on task models, grammars, or specific events.

Information Technology Platform	
Syntactic	Using all available data (e.g., client-side events and server-side logs) in order to obtain correlations between them. The combination of the available data in different components can reveal information impossible to obtain in independent evaluations.
Empirical	Preventing that while processing or transmitting logs interfere with the use of evaluated interface. The tool should implement safe and effective techniques without impacting on the website usage.
Physical	Do not depend on resources or specific configuration of the participants devices (e.g., disk space, processor frequency, bandwidth, etc). The evaluation tool should include mechanisms to achieve their goals in different configurations of hardware and software.

2.4 Conclusion

This paper presented a study of websites evaluation tools based on event logs, as well as different solutions to problems encountered at the event capture, log transmission, etc. The main results of this work are related to the identification of requirements and promising investigation subjects. Thus, developers of new websites evaluation tools can avoid the weaknesses and address some unexplored aspects such as the integration of A&U, for example. Based on the characteristics present in websites evaluation tools based on event logs, a set of requirements was proposed and have to be addressed to avoid the limitations discussed in this work. The next steps of this investigation involves the development of a website evaluation tool based on logs that combine the solutions in order to fill as much as possible the identified gaps, taking into account the requirements listed.

Capítulo 3

An Asynchronous Client-Side Event Logger Model

3.1 Introduction

Several studies have been addressed Web usage, ranging from Usability and Accessibility (A&U) guidelines to tools that analyze code, content, or logs of websites. Data Mining has been defined as the "analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner" [41]. Websites logs are commonly used as data source by data miners that focus on studying Web usage. The data mining of Web usage emerged as a new domain called Web Usage Mining (WUM) which is "the process of discovering and interpreting patterns of user access to the Web information systems by mining the data collected from user interactions with the system" [69].

WUM algorithms and tools focus mainly on server logs. Server-side data makes possible to identify the user route in a website requiring less effort, since Web server logs are a natural product of its use. However, server-side logs do not contain representative data about the interactions between the user and the Web page [37]. Client-side data have more detailed information about user actions in a Web page, but require more effort of the system to capture and transfer data to a local where researchers could analyze them.

Nowadays, HCI community count on tools that keep track of users behavior using mouse tracks (e.g., MouseTrack [7]) and eye tracks (e.g., eyebox2 [70]), but literature lacks studies focusing on data captured automatically from the whole diversity of users. Accessibility evaluation tools need to address some aspects usually not covered by evaluation tools based on mouse events or interaction by using a visual display. How tools that keep track of mouse or user's eye movements would help in an evaluation of a University's website that has teachers and students that are screen readers users?

In this context, we present a model to log event data that gets as many different events as possible, since with a large vocabulary of events in the logged data, researchers could perform many different analyses. One application of this model is to replay what clusters of users remotely do; this type of result would be costly to do with the videos used in tests in controlled environments.

The model requires the acceptance of the user to participate of the evaluation; the events recording are started as soon as s/he accepts it. In addition, a policy of privacy is implemented to allow the control of the type of events triggered by the participant that will be logged and transferred (e.g., if the "x" key is pressed then the data recorded will inform only that some key was pressed in some time and context of use).

After defining the model, we present an implementation of it as a case of study. The model and its implementation are part of the WELFIT (Web Event Logger and Flow Identification Tool) project, a work in progress tool to identify barriers that screen reader users face during a visit to a website being evaluated.

This report is organized as follows: the next section presents works related to client-side events capture; section 3.3 details the presented model; section 3.4 discusses implementation issues and details of this model in the Web context; finally, section 3.5 presents conclusions and future works.

3.2 Client-side event loggers

User Interface events are natural results of using windows based interfaces and their components (e.g., mouse movements, key strokes, mouse clicks, list selection, etc) [43]. Event logs produce results as frequency of use of certain functions, places where users spend more time and the sequence that users complete their tasks [81]. Since it is possible to record these events and they indicate the user behavior during the interface usage, they represent an important source of information regarding usability.

In this section, we will present WET and WebRemUSINE, client-side event loggers of Web pages, and discuss their common characteristics. Both tools use empirical data and are Web-based; however, their models can be generalized to log other kinds of client-side data-logger. WET focuses in logging data during formal tests (i.e., in test controlled environments). In contrast, WebRemUSINE can be used in remote tests and in both formal and informal environments.

The evaluation and comparison of these Web-based tools, including the implementation of the model proposed here, are made to keep a theoretical unit and to exemplify how client-side logging can help evaluators to perform studies in different kinds of devices. Some of implementation issues are tightly closed to Web context, but the overall idea and model can be applied to other devices (e.g., mobile devices, set-top boxes, and video

games).

Etgen and Cantor developed an event capture tool called WET (Web Event-Logging Tool). This tool, in contrast to traditional techniques of logging test sessions that use manual record of user interactions, makes automatic capture of events that occur on the client-side, avoiding high costs of time and money present in manual data capture methods [37].

The log captured by WET is recorded in text format at client-side (i.e., in cookies). The available space in cookies is about 4 kilobytes. The capture starts when the user indicates the beginning of a task through clicking on the start option, which is visible in the pages that use the tool. The capture is interrupted when the user selects the stop option. The transmission of information to the server occurs only when the stop is triggered.

WET authors comment that future directions of the project involve the use of Java applet for logs transmission and the construction of a wizard to assist in the configuration of the tool [37]. This fact suggests that the aspects of WET that need improvement in dealing with the client-side logging are: some way to record more data, use a bigger event vocabulary, and do not depend on user actions.

WebRemUSINE [63] is a tool that makes automatic analysis of websites interaction logs in order to detect usability problems through remote evaluation. The analysis of records from WebRemUSINE is based on the comparison between the paths made by users and the optimum task model configured [63].

To capture events the tool makes use of a technique similar to the WET, but a wider range of events is used. The storage and transmission of the logs is done through a Java applet, which allowed the tool to avoid the storage capacity of cookies, scenario that may occur in WET in cases of longer sessions [63].

For the user, using this tool involves splitting its screen into two regions, one for the list of tasks that the participant must choose before starting each task, and the other containing the website being evaluated. Once the preparation phase is completed, the WebRemUSINE allows the number of sessions to grow without requiring more effort from specialists. For the authors, one of the improvements planned for the system is the automatic generation of task models of websites [63]. This fact suggests that the points of WebRemUSINE that need improvement in dealing to client-side logging are related to: not depend on plug-ins installed on participant's device, not interfere in a significant way with the user interface being evaluated, and not depend on user actions.

These different approaches still have interesting challenges to reach their goals. The reader may note that the functions these two tools have in common result in a blueprint of what a client-side data-logger should do. In short, it must record client-side events that usually result in a large amount of data, interact with the participant of the test

showing the status of the tool, and transmit the logged data to an eventual server. This was the starting point for the model specified in the next section. The model proposed in this technical report details other phases than just recording and transmitting, addressing other needs, and avoiding some of the limitations of the previous tools. More information regarding the well succeeded techniques, gaps, and limitations of websites client-side logger tools are found at [29].

3.3 The client-side event logger model

This section describes the structure and components of the proposed model. It was built to address the requirements of websites evaluation tools (Table 3.1) elicited in a study of client-side loggers [29]. The requirements are organized in the Semiotic Ladder (SL), an artifact of Organizational Semiotics [73] which provides a framework for analysis of information systems under six different layers, considering the Information Technology Platform (i.e., physical, empirical, and syntactic layers) and the Human Information Functions as well (i.e., semantic, pragmatic, and social layers).

The model was designed so that its set up and use require just one change in evaluated applications: a call to the client-side event logger. Thus, as soon as a participant starts the test session and accepts to participate of the test, the tool starts to record events occurred at the client-side until the participant cancels his/her participation.

The developed model started from the use case specification, the domain model elaboration, and responsibility assignment to conceptual classes. The model specification was designed from the refinement of the main goal of the tool: to ***capture*** **event at client-side and** ***transmit*** **the logged data to a server, where all analysis is made**. This statement already indicates two components of the model: the **DataLogger**, responsible for capturing event data, and, the **Communicator**, responsible for transmitting logs to the server.

User interface events have a duration ranging from 10 milliseconds to one second [43], evidencing that the number of events that occurs during few minutes can be huge. Then, any data-logger that must transmit logged data through an information channel should have to compact the data to fill the requirements mentioned before. This brought the need for a component to compact all the data, the **LogCompactor**.

To deal with the amount of data recorded we propose the use of asynchronous communication with the server as a strategy to avoid using significant resources (i.e., disk and memory spaces), and to interfere as minimum as possible with the use of the website being evaluated. However, this solution can represent a limitation since it relies on the client bandwidth connection to the server. We will discuss how critic this point is in section 3.5, when we present results of this model.

Table 3.1: Requirements for websites evaluation tools instantiated in the Semiotic Ladder [29].

Human Information Functions	
Social	Focus on the integration of accessibility and usability for the target audience of the evaluated website. Enabling remote testing during real use of the evaluated website. Interfere with the Web page as minimum as possible.
Pragmatic	Providing controls representing the status of the tool and user context during the test session. The operation of the tool should use two actions: one to start the capture, which can stay valid for more than one session, another to interrupt the capture, which may occur at any time.
Semantic	Providing high levels of abstraction without depending on task models, grammars, or specific events.
Information Technology Platform	
Syntactic	Using all available data (e.g., client-side events and server-side logs) in order to obtain correlations between them. The combination of the available data in different components can reveal information impossible to obtain in independent evaluations.
Empirical	Preventing that while processing or transmitting logs interfere with the use of evaluated interface. The tool should implement safe and effective techniques without impacting on the website usage.
Physical	Do not depend on resources or specific configuration of the participants devices (e.g., disk space, processor frequency, bandwidth, etc). The evaluation tool should include mechanisms to achieve their goals in different configurations of hardware and software.

To manipulate data and perform record, read, and remove functions, we used a **DataAcessObject**, responsible to access the data recorded in the users' device like logs and session information. In addition, we needed a way to interact with the user and show the status of the logger, responsibility of the **Facade** component. To address privacy policies we created a component responsible to check if the captured data can or cannot be sent to the server, the **PrivacyFilter**. Finally, we defined a **Factory** to create and build all the components together.

The following sections detail the function and main methods of each of these components; an overview of the tool structure is represented in Figure 3.1.

The model addresses the following requirements presented in Table 3.1:

- **Physical layer** requirement, since it is lightweight (see section 3.4 for implementation details) and depends on few resources of the users' devices, achieving its goal in different configurations of hardware and software;
- **Empirical layer** requirement, since it process and transmits logs without interfering with the use of the evaluated interface;
- **Syntactic layer** requirement, since it uses all event available data that do not impact on security problems;
- The **Semantic layer** requirement is partially addressed. The model aims to log usage data without depending on specific task models, grammars, or events, but it does not foresee the analysis of the data to reach high levels of abstraction;
- **Pragmatic layer** requirement, since it provides controls representing the status of the tool and user context during the user interface test session, which makes possible that the user interrupts the capture at any time;
- **Social layer** requirement, since it focuses on interfering as minimum as possible with the user interface being evaluated. In addition, it makes possible the integration of A&U, since it keeps track of the interaction through different devices.

The following sections detail the components of the model: **DataLogger**, **Communicator**, **LogCompactor**, **DataAccessObject**, **Facade**, and **PrivacyFilter**.

3.3.1 DataLogger

The **DataLogger** is the central component of the model. The class contains the logic to manage the logging of events. It communicates with other classes from data layer (e.g., **DataAccessObject**) and interface layer (e.g., **Facade**, **Communicator**). Initially, the

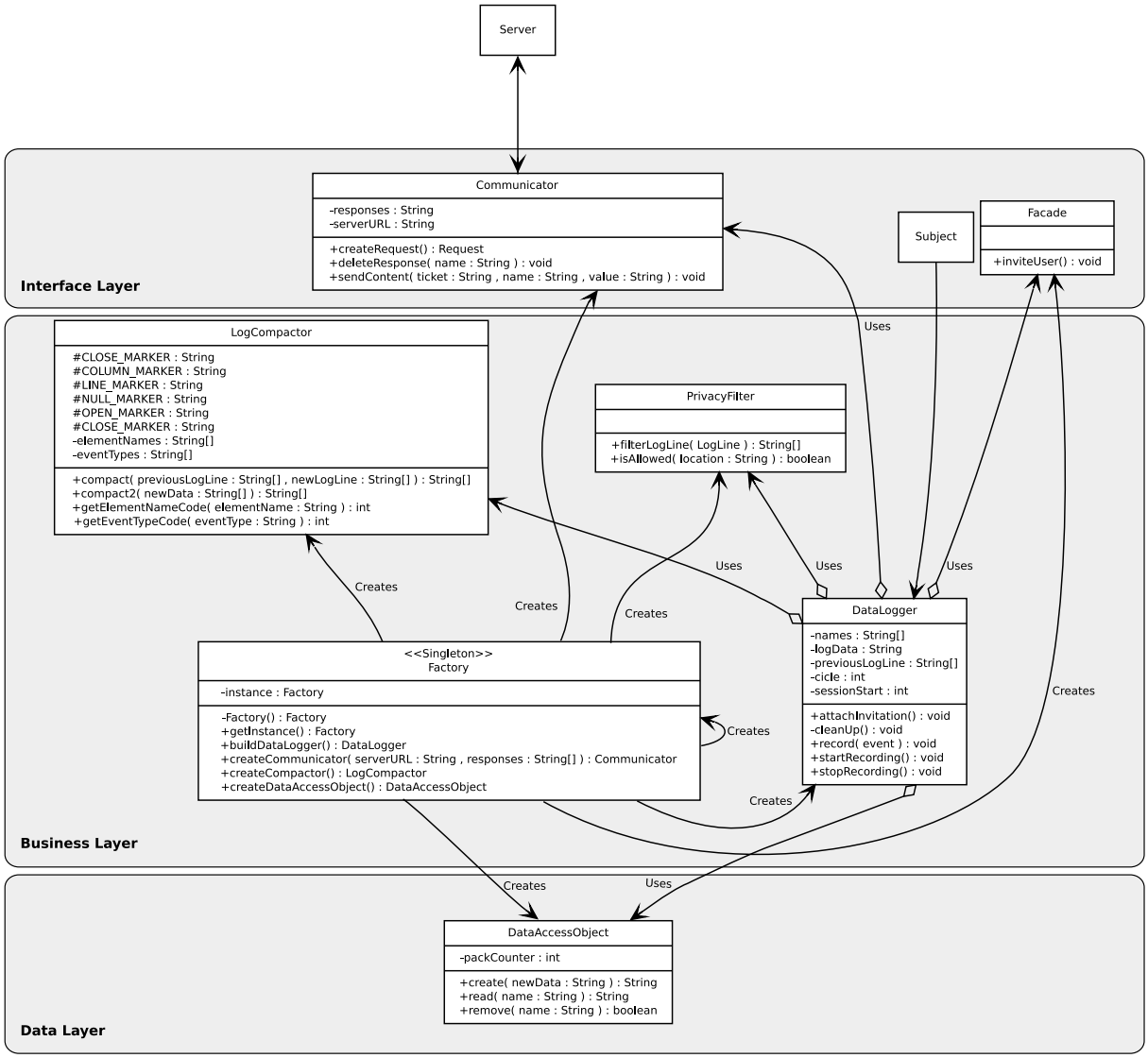


Figure 3.1: The overview of the client-side event logger model. This representation tries to simplify the whole design so it suppress some attributes, methods (e.g., setters, getters), and some data and abstract classes.

DataLogger is attached to the subject to be observed (e.g., Window object), so it can be notified to record all the events occurred. It is inspired on the GoF (Gang of Four) Observer Pattern. Observer is a behavioral pattern that defines a way to notify dependent classes when some change occurrs in the subject class [38].

The main attributes of this component are:

- *Log data* - contains the data of the current package being built. As soon as the logged data reaches a defined size, the **DataLogger** creates a package to be sent to the server;
- *Package size* - define the limit in bytes of the packages of logs;
- *Previous log line* - used to check what information changed between the previous event and the new one. It is used in the compaction procedure, detailed in **LogCompactor** section;
- *Clean up cycle* - define the interval in seconds in which the packages already confirmed by the server are deleted;
- *Session start, IP address, a ticket, and other information* - used to identify users' sessions and events stream, since the IP of different participations can be the same (e.g., when requests come from an corporative network).

As soon as it starts to run, **DataLogger** asks the **Facade** to show the invitation to the user. Then, if the invitation is accepted, the **DataLogger** is notified to start the event capture, when it attaches the record function to all event handlers of the subject object.

The **DataLogger** can also be stopped from recording events in two ways: first, when the user activates the button to stop recording; second, due to some problem occurred at the client-side (e.g., if the number of accumulated packages reaches a defined limit). It is also important to note that every change occurred in the status of the tool must be informed to the user.

While the participant interacts with the subject (e.g., window-based component), the events are triggered. Thus, for each event triggered the **DataLogger** records all information and build the log line. This line is passed to **PrivacyFilter**, further passed to the **LogCompactor**, and then, finally, added to the current package.

The **DataLogger** component looks for every defined event, since capturing all client-side events provide researchers ways of finding different patterns, a requirement to be addressed by the model.

As log lines are added to the current package the **DataLogger** verify if it has reached the defined package size. If it is the case, then the **DataLogger** asks the **Communicator**

to send the package to the server. Additionally, it calls the **DataAccessObject** to record the package just sent. This register is used if the **DataLogger** needs to send it again to the server.

Finally, at a define cycle; the **DataLogger** executes a method to verify if the server has sent a confirmation for the already sent packages. Thus, for every confirmation, the **DataLogger** asks the **DataAccessObject** to delete packages confirmed by the server and asks the **Communicator** to delete the confirmation messages that were already used. When cleaning up, if the **DataLogger** finds an error message or does not find a confirmation message, then the **DataLogger** calls the **Communicator** to send the respective packages again.

3.3.2 Communicator

The **Communicator** component is responsible for controlling the transmission of the logged data to the server. It keeps all the information regarding the identification of packages being sent and keeps all the responses emitted by the server. Additionally, it uses asynchronous transient communications to the server (i.e., by asynchronous we mean that the client sends a package to the server asynchronously while participants are using the Web pages; by transient we mean that the server is already waiting for requests). This characteristic is one of the improvements over the previous mentioned tools, enabling the recording of a wide variety of events without interfering with the website usage.

The **Communicator** must use secure methods to transfer the logged data. The packages can be ciphered or transferred through a secure channel, ensuring that other programs cannot intercept the packages. However, due to resources dealing with processing time or technologies, developers can leave the data less protected if they configure a less restrictive **PrivacyFilter**.

The main attributes of this component are:

– *Responses* - a collection of all responses sent by the server;

– *Server URL* - the address to where all packages are sent.

The **Communicator** initiates by establishing a connection with the server pointed by the server URL attribute. Then, for each package mounted by **DataLogger**, the **Communicator** creates a request to the server sending the package just built. Each package is identified by a ticket, a package name, and a value.

As soon as it receives the server response, it records the response that will be used when the **DataLogger** run the clean up procedure. Additionally, it also retrieves and deletes server responses as needed. These cases occur when the **DataLogger** cleans up, since it reads all messages and deletes confirmation messages that will not be used anymore.

3.3.3 LogCompactor

The **LogCompactor** was introduced to avoid the heavy consume of client's bandwidth connection, that may occur if the raw log is transferred to the server. Additionally, the compact technique to be used needs to be lightweight, so that the required processing is not noticed by users while they are using the artifact in evaluation.

In order to define the compaction technique to use, we reduced the search scope to the simple (and lightweight) compaction techniques. We tried using frequency table based compaction, in which we represent the most frequent values with short length codes. We got a compaction around 35%. But when analyzing the raw and the compacted data we found that the codes were commonly present in subsequent log lines. The Table 3.1 illustrates the sequence found in logged data in wich codes appear in subsequent log lines.

Table 3.1: Event flow data format example using codes 1 and 2 to events *mousemove* and *click*, respectively, and codes 10 and 11 to elements *[object HTMLBodyElement]* and *[object HTMLImageElement]*, respectively.

Timestamp(ms)	**Event code**	**X coordinate**	**Y coordinate**	**Element code**
400	1	50	50	10
500	1	60	60	10
600	1	70	70	10
700	1	80	80	11
800	2	90	90	11
900	1	100	100	11
1000	1	110	110	11
1100	2	120	120	11

After that analysis and due to event flow data format, we decide to experiment representing only the changed values (i.e., if an event is repeated many times at the same place, we just record the complete information of the first occurrence and for the next ones we record only the time they occurred, since the other values haven't changed). For an example, see Table 3.2. This technique brings compaction around 70% without making the compacting procedure a processor-consuming task.

Before detailing how the compactor works it is important to state that the log lines we use have fixed column length, even if the just recorded event does not have all the properties the data-logger is looking for. This approach was used to prevent the client to send control data to the server (e.g., property identification). Additionally, due to events flow data format and the compaction used, we prevent the null marker to represent null/undefined values in log lines, since they come as empty values until they change into event property data (see Table 3.2).

Table 3.2: An example of how 4 events can be compacted representing only data that changed. First, the participant moved the mouse over the logo image from coordinates 50 to 100 on the X-axis, then he clicked over the logo at 600 ms after the session has started, and then after a second he clicked again over the logo. The last column shows how null values are represented to maintain the fixed column length representation.

Time-stamp (ms)	**Event name**	**X coordi-nate**	**Y coordi-nate**	**Target ID**	**Element name**	**Referrer**
400	mouseover	50	200	logo	[object HTML Image Element]	null
500		100				
600	click					
1600						

The compaction has two steps:

1. Since the lines are fixed column length, the first step of compaction is a loop that verify, for each log line value, if new log line value is equal to the respective one in the previous log line. If this is the case, then this position on the compacted log line is replaced by an empty string. Finally, at the end of the loop the compacted log line is returned to the caller component;

2. The second step was added to improve the compaction obtained by the first step and reduce the overhead of markers used to separate empty values of the fixed length log lines. For this purpose, we defined a markup to replace repetitions of column separators (e.g., commas) by markers indicating the number of repetitions of the separator. Note that this will be done only if it is worth it (i.e., the number of commas is greater than the sum of repetition markers and the number used to represent the repetitions). An example of compaction of the Table 3.2 log data would be:

```
400,mouseover,50,200,logo,[object HTMLImageElement],-@
500,,100{3}@
600,click{5}@
1600{6}
```

Where comma (,) is the column marker, the hyphen (-) is the null marker, the at (@) is the line marker, and braces ({,}) are open and close markers. We should note that all the content must be escaped depending on the implemented markup

preventing conflicts between markers and event data. This means that all markers used to represent column (,), end of line (@), and so on, that appear in the logged data must be concatenated with another marker to avoid ambiguity between markup and data. For example, using the back slash to escape column and line markers we would get the "\," and "\@".

Complementing this approach, we defined a fixed codification to event types and to element names, reducing values as mousemove and click to 1 and 2, for example. In addition, we used the bigger base available to represent number values with less bytes (e.g., hexadecimal values, base 32, and so on). The following lines detail these techniques using the raw log data from Table 3.2:

1. Raw log data using markers to represent column, line, and null values (211 bytes):

```
400,mouseover,50,200,logo,[object HTMLImageElement],-@
500,mouseover,100,200,logo,[object HTMLImageElement],-@
600,click,100,200,logo,[object HTMLImageElement],-@
1600,click,100,200,logo,[object HTMLImageElement],-
```

2. Log data using event type codes, element name codes, and base 32 to represent numeric values (80 bytes, 62% of compaction):

```
cg,2,1i,68,logo,1,-@
fk,2,34,68,logo,1,-@
io,1,34,68,logo,1,-@
1io,1,34,68,logo,1,-
```

3. A first step compaction (50 bytes, 76% of compaction):

```
cg,2,1i,68,logo,1,-@
fk,,34,,,,@
io,1,,,,,@
1io,,,,,,
```

4. A second step compaction (44 bytes, 79% of compaction):

```
cg,2,1i,68,logo,1,-@
fk,,34{4}@
io,1{5}@
1io{6}
```

3.3.4 DataAccessObject

The **DataAccessObject** is based on the Data Access Object (DAO) J2EE Design Pattern. DAO provides a solution to abstract and encapsulate the access to the persistent storage, managing the connection with the data source to retrieve and record data [6]. The **DataAccessObject** component is located at data layer and its responsibility is to perform record, read, and remove actions as the **DataLogger** requests it. **DataAccessObject** is the component that access and retrieve data recorded in the users' device like logs and session information. In addition, the component defines a name for each package when creating registries.

3.3.5 Facade

The **Facade** is the user interface of the logger. It is inspired on the GoF (Gang of Four) Facade Pattern. Facade is a structural pattern that "defines a higher-level interface that makes the subsystem easier to use" [38].

The **Facade** is the class that shows the status of the tool and offer controls to the participant interact with the system (i.e., **DataLogger**). Initially, the **Facade** is asked by **DataLogger** to show the invitation and keep the participants aware of what is happening with the tool. Then, each activation of **Facade**'s controls calls the respective method at **DataLogger** component (e.g., stop recording).

3.3.6 PrivacyFilter

The **PrivacyFilter** is the component that uses previously defined policies (e.g., not record which key is pressed when a key press event occur, not record events occurred in a certain location). These policies can be defined in two levels: first level rules with priority are defined by the administrator of the tool and second level rules can be defined by the administrator of the application being evaluated, if they do not conflict with the former.

Two methods are used to define the rules while the tool is recording events:

1. Administrators can specify if some locations cannot be logged through the *isAllowed* method. Thus, when a participant accesses some location specified in some rule,

then the tool does not record events in that location (e.g., login.html). Before recording, the **DataLogger** asks permission for **PrivacyFilter** using the method *isAllowed*;

2. Administrators can specify which information of events cannot be logged through the *filterLogLine* method. With this method it is possible to define policies for each type of event. For example, if there's a rule to filter every key value when a participant presses a key, then when some key event is recorded then all values related to whichever keys were used will be filtered and, consequently, not sent to the server.

3.3.7 Factory

The **Factory** is the model creator class. The **Factory** contains the information to instantiate and build all components. First, the **Factory** instantiates itself, following the Singleton GoF pattern. Singleton is a creational pattern that provides a global point of access to it [38]. In addition, it uses two other GoF creational patterns: Factory Method, that "lets a class defer instantiation to subclasses" [38]; and Builder, that "separates the construction of complex object from its representation so that the same construction process can create different representations" [38].

This component was designed this way to allow the model to be extended allowing the definition of other classes used to assembly the **DataLogger**.

3.4 Implementation issues and results

This section discusses some of the implementation issues and results we achieved when applying the presented model. As mentioned before, this work is part of a project involving the evaluation of websites. Thus, the implementation was made using Javascript, a script language that is native of the newest Web browsers.

The use of this implementation requires only that websites being evaluated insert a reference to the Javascript in the header of Web pages. From that point, each time an user access the page, the server fills up the script with information unavailable from Javascript (e.g., client's IP, a global identifier for that session, etc.) before serving it. Then, as soon as the script is loaded at client's browser, the tool starts to run.

This model behaved well in the implementation using Javascript. However, some components faced limitations in dealing with security constraints. The main issues are related to the space available to record information on client's device and to transfer the data to a different domain from the website being evaluated.

The space available to record information in the client's device is restricted. The first solution was to use cookies, but the first negative point is that they are limited to a size of 4 kilobytes (kB) and each domain can specify only 20 cookies [60]. This results in a space of 80kB to a tool that captures approximately 4kB of raw log per second. This might be a problem, but the asynchronicity of the model dealt with that. Afterwards, the problem was the time required to record, retrieve, and delete the packages without interfering with the use of the website. Initially we used the same limits of cookies. Then, to avoid the time required to manipulate cookies we used the Web page structure in memory, also known as Document Object Model (DOM) tree. Finally, we used application cookies only to deal with error recovery and to maintain information valid for more than one session (e.g., the acceptance of the user to participate of the evaluation).

Another point that contributed to the solution of keeping the logged data in DOM tree is that some users deactivate cookie support, so the direct dependency of cookies might reduce the amount of cases to study. We used the same limits imposed by cookies the keep the tool as lightweight as possible, as stated in the requirements.

The data unit we are saving in cookies are the packages of log, then we are limited to the use of 20 packages in the DOM tree. Thus, either at clean up cycle or at reaching 10 accumulated packages the **DataLogger** tries to send it all again. If many errors occur and the limit of 20 accumulated packages in the client-device is reached, then the tool's **DataLogger** stops to record events. This solution to persist data showed to be efficient and effective.

The bigger issue implementing the proposed model in Javascript was the asynchronous cross-domain transmissions. The problem was to deal with security restrictions of the XMLHTTPRequest, a Javascript object widely used object in AJAX (Asynchronous Javascript And XML) applications, which just allows connection between Web pages/applications hosted at the same domain.

The security restriction is called Same Origin Policy, it "prevents document or script loaded from one origin from getting or setting properties of a document from a different origin" [67]. Then, if you try to make an XMLHTTPRequest from one domain to another, your Web browser probably will pop up a permission denied message.

At first sight, the Same Origin Policy may seem too restrictive, since it blocks the use of Web services directly via XMLHTTPRequests. However, according to Levitt [55], allowing scripts to access any domain from within a Web page opens up users to potential exploitation. Next we discuss some solutions to deal with this policy and finally comment about the solution we have chosen for this case study.

One of the best initiatives we found is the use of Signed Scripts. This involves "generating a digital signature and associating that signature with the script it signs" [66]. With this signature the scripts can obtain privileges to, for example, use cross-domain

XMLHTTPRequests. However, it is an initiative of the Mozilla project and depends on the security model implemented in Mozilla's compatible Web browsers, leaving aside all users of the other browsers, a price we do not want to pay.

Another interesting solution we found is to use server-side proxy programs, since the majority of server-side programming languages allow the cross domains connections. Thus, a participant viewing the Web page index.html at domain **www.example.com** would trigger events that will first be sent to the **www.example.com** Web server, and then a proxy program would dispatch the information to the location of the server of the tool (e.g., **www.toollocation.com**). This is indeed effective, but this solution came with two negative effects. First, all websites administrators using the logger would have to configure the proxy, a task that could be simple but also can even require the download of modules or installation of complements. Second, the Web server of the website using the tool would face a notorious overhead, since each package being sent to the tool's server would be passed first to the **www.example.com** Web server. This price, website administrators may not pay.

A very simple solution is the Iframe Proxy. In this solution, the information is exchanged by the URL of an *iframe* HTML element. This is possible because the document containing the *iframe* and the document being referenced in it can both update the iframe [35]. Thus, the **www.example.com** application can put a message in the URL of the iframe calling the **www.toollocation.com** and then the referenced page can take this message, process it, and put the response back into the iframe's URL. After that, **www.example.com** can use it at some timeout-defined cycle. This solution is interesting, however, as the iframe is updated some Web browsers (e.g., Microsoft's Internet Explorer) plays a sound, a price users should not have to pay.

Another client-side solution is the use of a Flash Proxy. Due to the policy used in movies in Flash to exchange data between domains it can be used as a proxy to send data to another server [57]. However the dependency on Flash plug-ins and plug-ins' versions would difficult the use of the logger, conflicting with the requirements presented before.

Some proposed solutions to the Same Origin Policy are: JSONRequest, module tag, content restriction header, W3C Access Control List (ACL) System, and Flash's crossdomain.xml [8]. However, the proposed solutions that would give to Javascript programmers the power to perform cross-domain requests are JSONRequest and module tag. Thus, we will detail these two approaches.

Before presenting JSONRequest, we need to introduce what is JSON (JavaScript Object Notation). JSON is a data interchange format based on a safe subset of Javascript to represent simple or complex structured data [23]. Therefore, "JSONRequest is proposed as a new browser service that allows data exchange with any JSON data server without exposing users or organization to harm" [23].

JSONRequest is a solution that acts like the XMLHTTPRequest with the following changes [8]:

- Uses a minimal set of HTTP headers, reducing the overall size of requests;
- Cookies would not be transferred, avoiding cross-site cookie issues;
- Accepts only JSON text, ensuring that code could not be sent for execution;
- Each communication failure is randomly scheduled to a retry, frustrating certain attacks;
- Returns a sequence number so that each response can be easily associated with the originating request;
- Supports duplex connections, enabling server to initiate communications.

The <module> tag proposes to divide a Web page into a collection of modules that are secure from each other and provide safe communication. This solution is somewhat similar to the iframe proxy presented before, but it specifies and foresees different uses. The communication between the page and modules is only allowed using send/receive functions and allows exchange data only in JSON text format. In addition, modules can cooperate only with the Web page, increasing the level of security between different modules. Thus, due to control over the send/receive communication, crossing domain between the module and the Web page is irrelevant [24]. According to Crockford [24], the module tag intent is to begin the process of reaching a consensus on a new Web browser security model, since Web applications are significantly ahead of Web browsers technologies.

The solution we are using in this implementation is based on an approach presented in Levitt [56].

This approach uses the fact that when the script tag is loaded, Web browsers execute the loaded script, also known as Dynamic Script Tag. Thus, manipulating the DOM tree and making requests through the creation of script tags dynamically allow asynchronous cross-domain communication.

When a script tag is created, an HTTP GET request is sent to the server module, which processes it and serves the response as JSON content to the caller Web page. HTTP GET method is used because this approach does not support HTTP POST method. JSON is used to avoid the parse of other formats (e.g., XML) since JSON can be used directly by Javascripts programs, what in fact makes this implementation an AJAJ (Asynchronous Javascript And JSON) application. Finally, the Web page can use the response and delete the respective script tag object from DOM tree.

Levitt [56]describes pros and cons of this approach compared to XMLHTTPRequest. Some of them are:

- Dynamic Script Tag runs identically on more Web browsers than XMLHTTPRequest. An example of that is the Microsoft's Internet Explorer implementation of XMLHTTPRequest, that it is different from other Web browsers and requires a compatibility layer or additional verifications;
- Dynamic Script Tag does not support HTTP POST method;
- Dynamic Script Tag cannot send and receive individual HTTP headers; consequently, it cannot handle errors gracefully through HTTP.

These points might suggest not using it. However, we are aiming not rely on plug-ins or any other components other than Javascript and Web pages. The dependence on plug-ins brings the need for installing other software, configuring it, etc. This is a temporary solution since the technology of most popular Web browsers does not support completely the model we proposed. Additionally, the security issues due to transmission of coded logs through HTTP GET can be facilitated through restrictive configuration of **PrivacyFilter**, as presented before.

Tests we made in the homepage of the website **www.todosnos.unicamp.br**, using random mouse movements over links and page elements, showed that each second of interaction results in approximately 1kB of compacted logs. Therefore, any participant using a connection that supports the transmission of 1kB per second plus the mean of bandwidth connection used by the participant to surf the Web will allow the logger to behave accordingly to the design and does not interfere with the use of the website. If it is not the case, then, as specified in the model, the tool must became inactive and does not record data if the available bandwidth does not allows the transmission of the logged data according to the defined requirements.

3.5 Conclusion and future works

The model presented in this technical report is an alternative to log client-side usage data. It can supply data to other applications focusing on discover patterns of what clusters of users do or just to replay user's actions during tests in controlled environments, like the well known video tools used in usability tests. The implementation showed to be lightweight and addressed the requirements stated for a client-side event logger tool.

During implementation and use of this model, maintainers and developers must always keep security and privacy in mind, since the information being captured and transmitted can be critic if it is not made in a safe manner. Accordingly, users must always be aware

of what is happening in their device and accept to participate before the logger starts to record events, since the free record of this kind of information without warning the user would characterize the tool as a spyware, for example.

Among the limitations of the model is the inability to register any action if the user agent does not support the language used to implement it (e.g., Javascript). Moreover, if the device using a data-logger based in this model has a connection that transmits data in a lower speed than data is collected, then the data-logger will not work properly and consequently stop recording events.

There are different approaches for compacting data, but as stated before we aimed to find a technique that could balance effectiveness and efficiency for events flow log data. Thus, one future work to this model is to test different compacting techniques to obtain an improved lightweight compaction. Another topic to be studied is the addition of detection and/or correction mechanism for the transmission of compacted data.

Another future improvement is to change the Dynamic Script Tag approach into a plug-in independent more adequate approach, as soon as it gets implemented (e.g., JSON-Request, module tag).

Finally, the completion of the WELFIT project foresees the implementation of the server module, complementing the data-logger. This module will perform WUM using the logged data to identify barriers that participants faced when navigating through websites. This project aims to infer high-level patterns of usage without depending on specific task models or grammars. Reaching this goal will make WELFIT address the Semantic Layer requirement.

Capítulo 4

Revealing Usage Patterns of Web Pages

4.1 Introduction

Several studies involving real use log data have been performed and implemented in website evaluation tools and WUM (Web Usage Mining) algorithms. However, most of them make use of server log as data source. This trend emerges from the fact that server logs are a natural product of Web servers functioning; thus the capture of this kind of data is straightforward. Moreover, they do not provide detailed information about what users do when they are interacting with UI (User Interface) elements of a Web page in a granularity level higher than the page-view level [7]. This occurs because Web server logs keep track of requested pages only, HTTP (HyperText Transfer Protocol) method used, time of the request, etc., while client-side event logs allow keeping track of all events triggered from users (e.g. mouse movements, mouse clicks, or pressed keys) or from the browser (e.g. image is loaded, form receives focus automatically, some error occurs).

Evaluation tools such as WebQuilt [44], LumberJack [45], Descubridor de Conhecimento en la Web [36], Web Utilization Miner [72], WebSIFT [18], for example, use server-side data logs. Few tools use client-side data event logs. Some examples are WebRemUSINE [63] and MouseTrack [7].

From now on, we use usage graph as a directed graph where nodes represent triggered events and flow as the representation of transitions of nodes of the usage graph (Figure 4.1).

In the Web context, accessibility barrier is anything that makes it difficult or impossible for people with disabilities to use the Web [79]. Usability problems can be defined as UI aspects that reduce system's usability for end users [26]. In some cases, both issues can occur at the same time. For example, if a Web page has an element, say a link, that

is too small and then too hard to be pointed by users with disability and users without disabilities, it represents an accessibility barrier and a usability problem.

Web usage patterns, at UI event level, represent how users interact with specific Web page elements. They represent an interesting source of information, opening new possibilities of scenarios of how to evaluate UI design of websites remotely, using real data. These characteristics acknowledge the fact that tests in controlled environment are artificial and may influence the results [65]. Finally, automatic UI evaluations using client-side event data allows to record how users use the Web page, making possible the detection of accessibility barriers and usability problems.

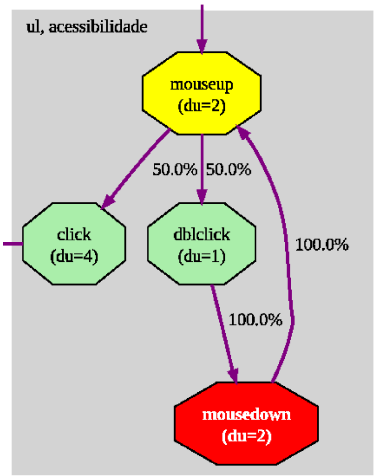


Figure 4.1: Transitions from the event *mouseup* to *dblclick* and *click*, both with edges labeled with the percentage of observed transitions, 50%. The element in which the events were triggered is an Unordered List (UL) with the ID attribute acessibilidade. This indicates that, in observed cases in a certain page, when at mouseup on UL "acessibilidade", the next events were *dblclick* or *click*.

This work is organized as follows: section 4.2 presents how the client-side data used in this work was captured; section 4.3 discusses characteristics of client-side event data and how to build the usage graph; section 4.4 presents how the usage patterns are revealed; section 4.5 presents the results achieved so far; and, finally, section 6 presents the conclusions.

4.2 Data acquisition

This work is part of the project of a tool called WELFIT (Web Event Logger and Flow Identification Tool). The project involves the development of a model to capture client-side event logs, transfer them to a server, and mine the captured logs in order to discover

usage patterns in Web pages. In addition, the tool aims at pointing out accessibility barriers and/or usability problems that assistive technologies users may face.

To use client-side event logs the capture should be lightweight and should not interfere with the website use. The user must be aware of the evaluation and must not feel disturbed by the system. In addition, the development of new evaluation tools based on client-side events must consider some specific requirements, since the data capture involves more tasks (e.g., compacting logs and transferring them to the server). The complete elicitation of the tool's requirements is discussed in [29].

WELFIT captures all types of standard Javascript client-side events remotely, during real use, when users are in their natural usage environment. The capture consider all types of events, since each one represents one kind of action performed by the user or by the browser, thus allowing the analysis of everything happened during the session.

When a Web page event is triggered, it travels from high level Web page's objects (e.g., Window, Document) to the target element (i.e., the page element in which the event was triggered). Consequently, the inclusion of a mechanism to capture all the events triggered at this root will allow keeping track of all events that pass through this element. The WELFIT's capture module specification is detailed elsewhere [28].

Server logs represent information dealing with requested URL (Uniform Resource Locator), HTTP method used, and so on. In addition to that, if a tool is capturing client-side events, a simple download of the page, without any device-triggered event, can contain time-stamped data representing the download time of each image, and window dimensions. It is worth noting that this simple example can be an interesting case to check the region or ads that are probably visible to users when visiting a Web page.

Depending on the assistive technology being used, events can be triggered outside of the Web browser (e.g. screen readers). In the case of screen readers, they take a snapshot of the page and place it in a virtual buffer, so that users can interact with elements that cannot receive focus [53]. With this approach some screen readers avoid triggering events at Web browser, thus hiding valuable information used by the tool. This can be an obstacle, since in some cases data may be not captured by WELFIT; however, the approach used by screen readers is due to a limitation of HTML (HyperText Markup Language) versions earlier than version 5. The new version of HTML will allow all elements to gain focus, since *tabindex* attribute became a global attribute [87]. In addition, the assistive technologies considered are not restricted to screen readers; we also take into account screen magnifiers, widgets for font resizing, high contrast color scheme, etc.

The data capture in WELFIT is performed according to the following steps: when a user accesses the evaluated website, s/he receives an invitation to be part of the evaluation. In addition, s/he is asked if s/he is using an assistive technology. This information is used

to mark the session and is used to compare and group users' sessions. After submitting the answers, if the user agreed to participate in the evaluation, then the capture starts.

The real data used in this work were captured by WELFIT during the interaction of users of a research group website called Todos Nós (www.todosnos.unicamp.br). Since one of the group focuses is the study of accessibility, part of the website's audience uses assistive technology. This website was chosen because WELFIT aims to identify usage flows including the barriers faced by users of assistive technologies.

The data captured during 47 days resulted in 75 sessions, 6 of them coming from assistive technologies users. All sessions are related to 256 different URLs (i.e., involve different pages and dynamic pages with different parameter-based content), 273,209 log lines, and over 80 MB of data. In addition, the average number of visited pages per session was 3.5 and the average number of events per page was 1,063.

4.3 Representing client-side event data

User interface events have a duration ranging from 10 milliseconds to one second [43], suggesting that the number of events that occurs during few minutes of interaction can be huge. Thus, the time measure unit used in client-side event logs must be milliseconds.

Representing these data involves the following elements: the Web page (`P`), the number of elements in the page (`El`), the number of events available in the language used (`Ev`), and the timestamp of the event (`T`). Initially, each node of the graph representing P was made using the following values separated by comas: the target element, the element's ID or name, if any exist, the triggered event, and the timestamp in which the event was triggered. Thus, a form element with "welfit" as ID attribute and receiving a focus event at 1000 milliseconds will result in the node named as "form, welfit, focus, 1000", for example. Additionally, we represent the usage graph not allowing cycles in the directed graph G, so we could represent all the event sequences to visualize as much information as possible. However, this approach resulted in graphs with a number of nodes ($\|V(G)\|$) depending on the number of events per session. This was ineffective since visual reports generated with this approach had poor usability due to the huge number of nodes representing all sequences of events.

Afterwards, not considering the timestamp information to give name to the nodes resulted in more concise graphs, now containing cycles for subsequent events of the same type, a common situation found in client-side event logs [28]. With this representation the generated graphs are dependent only on the number of page elements, resulting in graphs containing a maximum of $\|V(G)\| = El * Ev$. Thus, the number of nodes is not influenced by the number of tracked sessions and the edges representation is more compact. Finally, we also clustered events related to the same target element to represent client-side usage

flow in a consistent and reduced way.

Thus, using walks performed by users, which can be defined as non-empties alternating sequence of vertices and edges [34], we used the Algorithm 1 to build usage graph for a page P. To represent the graph structure we used the JGraphT [50], which is a free Java graph library that provides mathematical graph-theory objects and algorithms and also helps generating graphs in definition languages as Dot [1], for example.

4.4 Usage patterns

With the structure and data obtained with the usage graph, it is possible to extract information of walks with greater edge weight values or to check the most performed transitions between nodes. In dense graphs, simple filters for edge weight or transition percentage can reduce the overall graph representation, keeping only the most repeated patterns that show the main event sequences triggered when users interacted with evaluated Web pages.

Once having the usage graph structure and some usage patterns, it is necessary to evaluate the walks present in the graph in order to find differences between walks to analyze accessibility and/or usability problems. Thus, to highlight flows in order to represent differences present in the event sequences we followed the Sequence Alignment Method (SAM) approach presented in [42] and defined a heuristic to identify possible accessibility barriers and/or usability problems based on SAM properties.

SAM, also called Edit Distance, is a distance-based technique that represents the amount of operations necessary to equalize sequences [42]. Then, the lesser the distance between two sequences, the similar the sequences are. This method reflects structural information through the order of elements within sequences, contrarily to the commonly used distance measures based on Euclidean distance [42].

The heuristic we defined aimed at using the average distance to a certain node and how this measure changes across the flows in the outgoing nodes. This approximation allows the comparison of distances between more than two sequences at once and at every node with out degree grater than one.

First, be G the digraph representing page elements of a certain page P and a certain node of G, say u, has a average distance d_u. Additionally, be u's neighbors v_0, v_1, ..., v_k, and d_{v_0}, d_{v_1}, ..., d_{v_k} the respective average distances. Thus, if $d_u > d_{v_i}$, then the in degree of v_i is greater than 1, meaning that it exists some alternative walk to v_i that does not have u as precedent node and has a smaller average distance. Thus, the walk from the root to u is not the short one, indicating a performance difference and may indicate that u is part of a walk containing an accessibility barrier and/or usability problem.

Finally, to visually represent them we used a minimum and a maximum threshold. They are used to define colors for each node according to the following:

```
   Data: Website and page's URL
   Result: Usage graph structure
 1 read sessions of website and URL ;
 2 graph ← newgraph ;
 3 graph.addVertex(start) ;
 4 graph.addVertex(end) ;
 5 foreach session in sessions set do
 6 |   read events for session ordered by timestamp ;
 7 |   previousVertex ← start ;
 8 |   distance ← 0 ;
 9 |   events ← start + events + end ;
10 |   foreach event in events set do
11 |   |   vertex ← newvertex ;
12 |   |   if event.targetId != null then
13 |   |   |   vertex.name ← event.targetTag+",+event.targetId ;
14 |   |   else if event.targetName != null then
15 |   |   |   vertex.name ← event.targetTag+",+event.targetName ;
16 |   |   else
17 |   |   |   vertex.name ← event.targetTag ;
18 |   |   end
19 |   |   vertex.name ← vertex.name+",+event.type ;
20 |   |   if !graph.containsVertex( vertex ) then
21 |   |   |   vertex.n ← 1 ;
22 |   |   |   vertex.d_u ← distance ;
23 |   |   |   graph.addVertex(vertex) ;
24 |   |   else
25 |   |   |   vertex ← graph.getVertex(vertex.name) ;
26 |   |   |   vertex.d_u ← (vertex.n * vertex.d_u + distance)/(++vertex.n) ;
27 |   |   |   graph.setVertex(vertex.name, vertex) ;
28 |   |   |   if graph.containsEdge( previousVertex, vertex) then
29 |   |   |   |   e ← graph.getEdge(previousVertex, vertex) ;
30 |   |   |   |   graph.setEdgeWeight(e, e.weight + 1) ;
31 |   |   |   else
32 |   |   |   |   graph.addEdge(previousVertex, vertex) ;
33 |   |   |   end
34 |   |   |   previousVertex ← vertex ;
35 |   |   |   distance++ ;
36 |   |   end
37 |   end
38 end
```

Algorithm 1: Procedure used to build the usage graph

- If the average distance value is greater than the maximum threshold, then the node needs to be marked as a part of a potential barrier/problem (red);
- If the average distance value is greater than or equal the minimum threshold and is lesser than or equal the maximum threshold, then the node does not reveal any novel information since it has a value in the expected interval (yellow);
- If the average distance value is bellow the minimum threshold, then it represents a shortcut to other walks with greater average distances (green).

4.5 Preliminary results

The flows are compared based on the average distances walked by users from the first triggered event to all events triggered at all page's elements. With this, representative distances differences walked by users to a certain element, combined with the presented heuristic, reveal significant performance differences, indicating possible design problems (Figure 4.2).

To generate the visual graph reports we used the graph drawing software called Graphviz [1]. The usage graph structure allowed constructing the Dot language source file with two loops, one reading the vertex set and other reading edges set.

From the visual reports representing usage patterns it is possible to identify strategies used by assistive technology users. Additionally, some patterns found showed the sequence in which events are triggered in the evaluated website, representing how the user interacts with Web page elements, making possible studies of the effectiveness of some design decisions. Some of the patterns indicate that people that informed to be assistive technology users, access the accessibility toolbar containing skip links, font-sizes, etc, as one of the first navigational elements.

4.6 Conclusion

This work presented how to process logged data and automatically generate usage graphs summarizing statistic information present in the client-side event logs.

The visual representation of the graph structure and the filter parameters presented in this work allows the visualization of the flow of events of a Web page, representing more usual transitions in the usage graph, thus representing how users commonly interact with page elements.

The patterns and walks presented by the tool are an interesting data source to incorporate personalization features. It is possible, for example, to avoid repetitive tasks

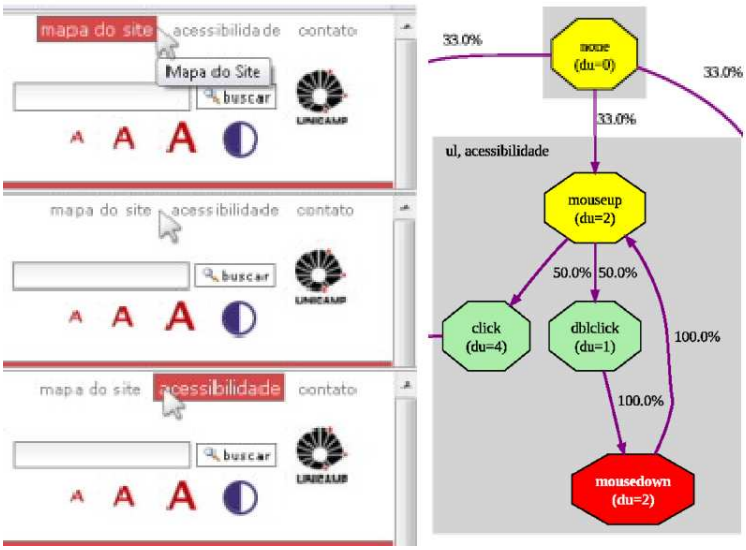


Figure 4.2: An issue found using the presented approaches. At the left, the UL element identified by "accessibilidade", that contains links using a *hover* Cascading Style Sheet (CSS) pseudo class to increase links' size (i.e., when the user passes the mouse over the links, they grow). At right, an extract of a usage graph representing part of a session of an assistive technology user. This extract shows events triggered at element UL with the ID "accessibilidade". The event sequence shows a failed attempt of double clicking at one of these links due to the resize, lacking the consistency of elements' dimensions.

like accessing the accessibility toolbar. In addition, it is possible to define global *tabindex* based on the most repeated patterns of navigation, thus giving a global order of elements based on empirical data representing most common users' real behavior.

Finally, one drawback of this work is that it requires a specialist, knower of user interface events and elements, to understand the report. Thus, a possible improvement in the tool is to incorporate an automatic generated summary of the report to make it more accessible too.

Capítulo 5

WELFIT: A Web Event Logger and Flow Identification Tool of Client-Side Logs

5.1 Introduction

Accessibility and Usability (A&U) are playing an increasingly important role in the achievement of a successful website [11]. As a result, the evaluation of A&U is an important phase of User Interface (UI) design. Some UI evaluation methods require capturing, gathering, or analyzing large data sets, tasks that can be time and resource consuming, leading to the use of automatic tools. The total or partial use of automatic usability evaluation methods can reduce time and costs involved in the development of Web applications, as it liberates specialists from repetitive tasks as manual analysis [63]. Thus, beyond the improvement that evaluation tools bring to both UI design and website development, they promote the consistency of evaluations and allow the number of test sessions to scale up, allowing more participants without increasing the evaluation costs.

Automatic evaluation tools provide an important support for UI designers and developers. They help maintainers of websites in many different tasks as validating code, applying guidelines verification, analyzing website's structures, and capturing and analyzing Web usage data. In websites development, UI evaluation gains a special responsibility since the design, development, and maintenance are influenced by other variables. The rapid pace and tight deadlines involving development and maintenance of websites, affect negatively the effectiveness and reliability of whole processes, from the detection of failures to the evaluation of benefits and costs of the proposed changes [11].

The UI evaluation can be remote or non-remote. Remote evaluation means that the user does not need to move to some test environment or lab to participate in the evaluation.

On the contrary, the non-remote evaluations require the user to be present in a controlled evaluation environment.

When evaluating UI, the participant can make informal use, when the evaluation requires the completion of freely chosen tasks, or formal use when requires the completion of specifically selected tasks) [49]. The automation of tools can involve **capture** (i.e., logging usage data), **analysis** (i.e., identification of problems), and **critique** (i.e., suggestion of improvements) [49]. These characteristics are interesting since they prevent the UI use to be biased during the evaluation. Furthermore, tests in controlled environments are artificial and may influence the results [65].

Events can be defined as effects resulting from user's or system's action. They can occur at client-side and at server-side and often the collection of these events is called, respectively, client-side logs and server-side logs. Besides, UI events are natural results of using window-based interfaces and their components (e.g., mouse movements, key strokes, mouse clicks, list selection, etc) [43]. Events triggered at client-side are more detailed data sources of how users use a website [63, 37] than server-side data that can lead to less accurate results [17]. On the one hand, server-side logs allow keeping track of users' flow in the website. On the other hand, client-side logs allow keeping track of users' flow in a Web page and in the website.

Usability is the "extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [48]. Web Accessibility means that people with disabilities can perceive, understand, navigate, interact, and contribute with the Web [79]. The integration of A&U is necessary, since evaluating them separately may result in different priorities for the problems found separately in A&U and then interfering in the way the target audience uses the evaluated website. The lack of such integration may result in usable sites with low accessibility or accessible sites with low usability [21].

The number of existing automatic evaluation tools of websites is representative and increases year after year. However, some points were not addressed yet. A tool can apply automatic usage capture and automatic analysis, but requires a setup and specific client configuration [29]. In addition, some tools can keep track of users' eyes or mouse movement, but lacks capture of usage data from the whole diversity of users, independently of device or method used to surf the Web [28]. Finally, few tools combine A&U evaluation or the capture and analysis of accessibility artifacts remotely, during informal use.

From these facts, the focus of our work is on the remote evaluation considering informal use, since it allows data capture during real use when users are interacting with the evaluated UI in their usual environments. Thus, UI evaluation in real situations points to the use of event logs, since it makes possible to capture data while users work on their regular tasks, in their regular environments [39]. Thus, this work presents a tool called

WELFIT (Web Event Logger and Flow Identification Tool) that uses both automatic capture and analysis, using client-side logs as data source.

This paper is organized as follows: the next section presents related works and evaluation tools, section 5.3 presents the WELFIT's model, section 5.4 presents and discusses the preliminary results, and section 5.5 presents the conclusion.

5.2 Related work

In this section, we will present works correlated to WELFIT that go in the following different directions: client-side logging tools, proxy-logging tools, Web page's content evaluation tools, and Web Usage Mining (WUM) tools.

WebVIP is one of the first efforts to capture client-side events automatically. The vocabulary of events, which stands for the number of different event types, is restricted to the events: key press/release, pointer press/release/move, enter/leave widget, and enter/leave window. The environment configuration requires a local copy of the entire website being evaluated so that the tool can include the event handling and the code responsible for capturing events [59].

WET is a logger for formal tests. It uses cookies to store logged data, influencing the reduction of the vocabulary of events due to available space issues. The user must indicate the starting and ending of the capture through tool's controls [37].

WebRemUSINE is a tool that makes automatic capture and analysis of websites interaction logs in order to detect usability problems through remote evaluation. The analysis of records is based on the comparison between the paths made by users and the optimum task model configured. The storage and transmission are made by a Java applet component. In addition, the user must select the tasks s/he is performing so that the events captured can be related to the task selected by the user [63].

WebQuilt is an automatic capture and analysis tool that uses page-view level logs as data source. It uses a proxy-logger that mediates between users and Web servers and stores the communication between them. This characteristic prevents changes in the client-side, enables the identification of a large number of people navigating between different websites, and allows tests to be made remotely. Thus, a participant must just start the test session at the website of the tool, so that the tool starts keeping track of the communication between the client and Web servers to which requests are sent. This makes possible usability tests on any website, even if the evaluators are not the maintainers of the websites visited by the participants [44].

MouseTrack is a proxy-based usability system that performs automatic capture and analysis. It provides an online configuration and visualization tool that shows the mouse path followed by website visitors. The tool fetches the Web pages being evaluated and

modifies them by inserting Javascript code responsible to capture mouse movements. Then, when a user clicks in one link of a page changed by the tool, the mouse movements coordinate set just recorded is sent with the request to the tool's proxy, which records the mouse movement coordinates and returns the requested page to the user [7].

NAUTICUS (New Accessibility and Usability Tool for Interactive Control in Universal Websites) is a content evaluation tool that performs automatic analysis and critique, checking if a website is usable for users interacting through screen readers. In addition, it is one of the first efforts combining A&U in an evaluation tool. NAUTICUS applies verifications according to a set of criteria stated by the tool's authors in order to help designers and developers to apply them consistently. The 19 criteria intend to be general principles that should be considered during the development phase of websites [21].

EvalIris is a web-based system that performs automatic analysis and critique using Web pages' code to check the conformity with configurable guidelines. The tool applies the guidelines and returns errors and warnings found in the evaluated Web page. The report can be presented as a list or attached to the evaluated Web page, putting the errors messages in the places they appear. It also provides the feature of evaluation as Web service, allowing its use as a component of other tools [3].

MAGENTA (Multi-Analysis of Guidelines by an Enhanced Tool for Accessibility) is a web-based system that performs automatic analysis and critique using Web pages' code to check whether a website is accessible and usable. The tool applies verifications using configured guidelines and returns suggestions of how to deal with problems found, helping in the task of modifying the implementation in order to improve website's A&U. In addition, the tool uses a Guideline Abstraction Language markup to allow changes in the guidelines set; also, it provides a visual editor for the guidelines [54].

Web Utilization Miner is a miner system for the discovery of navigation patterns in websites using server-logs as data source. The tool provides a declarative language that allows a human expert to define the criteria of how interesting is the information contained in the server logs. In addition, it uses directed graphs as representation for the patterns [72].

WebSIFT (Web Site Information Filter) is a miner system that uses server-side data and count on algorithms to identify interesting knowledge dealing with usage, content, and structure [18].

LumberJack is a tool that processes Web server logs and uses the content and hyperlinks to build a model of user activity. It applies clustering analysis and then computes a number of statistical analyses for each discovered group [45].

DCW (Descubridor de Conhecimento en la Web) is a tool that uses WUM over server-side logs in order to discover navigation patterns and association rules [36].

Some of these works have drawbacks and gaps discussed in [29]. Thus, to address some

of them, WELFIT aims to provide a simple environment configuration and uses client-side capture during real use of the broadest possible audience, without requiring specific devices or events vocabulary. In addition, it does not depend on user's action to perform primary functions (e.g., send data to the server or attach session's data to a task selected by the user). Moreover, it does not depend on technology support on the client-side other than Javascript, a native technology of the newest Web browsers. In addition to that, it integrates A&U in website evaluation. Finally, it uses WUM techniques to identify usage patterns in a granularity level higher than the page-view level without depending on task models to do that.

5.3 WELFIT: Conceptual model and system

The WELFIT is a tool that uses automatic capture and analysis in order to identify usage patterns and accessibility barriers and/or usability problems, using client-side event logs as data source. The system's model has two modules: client and server. In order to address gaps and provide mechanisms lacking in studied tools, WELFIT follows the requirements for websites evaluation tools based on event logs detailed in [29]. In the next sections, we will present characteristics of client-side log format used by the tool, the tool's model, WUM techniques used in the system, and, finally, reports generated.

5.3.1 Log format

Capturing client-side events bring more responsibilities and tasks to the automatic capture tool, other than just gathering logs as occurs when using server-side logs. Functions such as compact, build packages of logs, and send them to the server, are examples of tasks that should be performed when capturing client-side data [28]. These tasks result from the fact that the logs need to be recorded outside of the client device and, consequently, logs need to be transmitted using the minimum bandwidth as possible, leading to the use of data compression and other techniques.

Tools and studies using server-side log as data source already deal with huge volumes of data. Moreover, when changing from the page-view level to UI events level this volume tends to get bigger proportions, mainly because the time measure unit pass from seconds to milliseconds and what was a simple page-view may turn into a stream of hundreds of events [28].

From the fact that subsequent log lines contain correlated information of events or UI elements [28], we used a 2-phases incremental compression technique based on the Run Length Encoding (RLE), which is a lightweight method of data compression that avoids representing data that does not changed. The first phase uses a y-axis RLE compression,

keeping in the new log line only the data that changed in relation to the previous log line and empting columns that have the same value that the previous one. To save space in the representation of the log lines, a fixed length column was used. Due to this, we apply the second phase of the RLE-based compaction, now on the x-axis, replacing series of column separators of empty columns by the number of repetitions that they occur [28].

With this technique, we obtained an average compaction of 55% in packages already using codes for tag names and event types (e.g., the tag a is 1, the event type click is 4), resulting in a good starting point to a lightweight automatic client-side events capture tool, since it follows the requirements presented in [29].

5.3.2 Tool's model

WELFIT has two main modules: the client module was implemented in Javascript and follows the MVC (Model View Controller) pattern; the server module was implemented using Java related technologies, also following the MVC, using POJOs (Plain Old Java Objects) at Model, Java Servlets at Controller, and JSPs (Java Server Pages) at View.

The client module is responsible to capture events at client-side, compact the data, and transmit the logged data to the server. The server module receives the data sent by client modules and store them for future analysis.

The tool requires a user's browser supporting the technology used to implement the data-logger, in this case Javascript, and the user's acceptance to participate in the evaluation. The specification and implementation details of the client-side logger can be found in [28].

The environment configuration has tree steps:

1. The website administrator must register him/herself at WELFIT's Web administrative interface;

2. Once logged, the administrator can register the websites s/he wants to evaluate;

3. Once the website is registered, s/he includes the call to the client-module Javascript as the last component of the page (i.e., just before the </body> tag) in all website's pages.

We assume that the best place to include the client module call is between the tags <head> and </head> or just after the <body> tag, since this positioning allows the logger to capture events as soon as possible, loading the client module as one of the first page elements. However, the position we are using tries to keep the client module more browser independent as possible due to an Internet Explorer 7 bug.

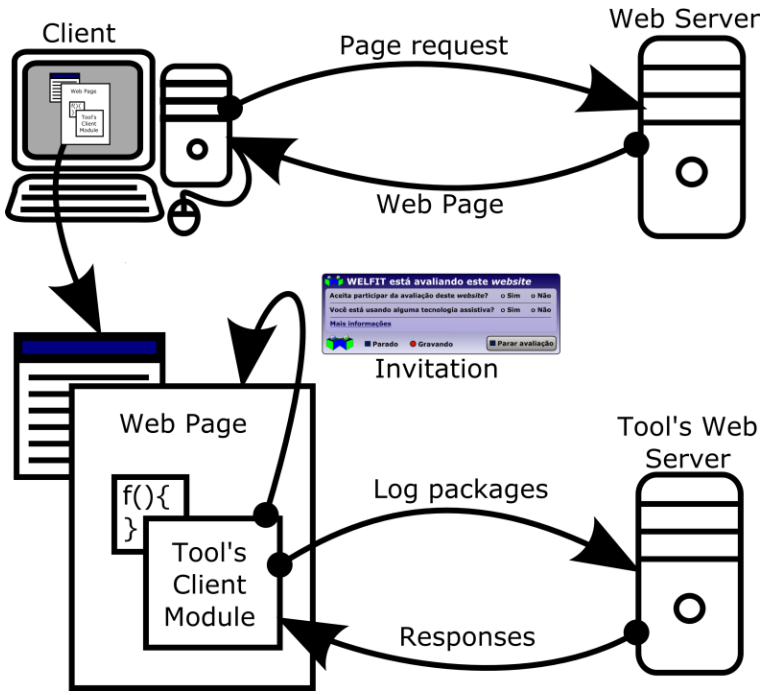


Figure 5.1: Overview of WELFIT's client-server model components, and message exchange.

This can cause the logger to lose some UI events since it is the last thing to be loaded. The rationale behind this decision is that the side effect of the bug is critical. The problem occurs due to the dynamic creation of page elements while the page is loading and its side effect is to close suddenly the referred browser just after presenting the message "Operation aborted" [19]. This bug was fixed in the new version of the Internet Explorer.

As soon as the website's administrator inserts the call to the client module in the website's pages, the WELFIT starts to work (see Figure 5.1 for an overview of WELFIT functioning). After the user request and download of a page in the website being evaluated, a request is made to the WELFIT server to download the client module. Then, the server module verifies if the request is coming from a registered website. If this is the case, then the server builds the client module, which has a fixed and a built-on-the-fly component.

The client module's fixed component has the implementation of all objects except by the Factory component, which is the component built differently according to the request. The Factory is responsible to instantiate itself and all other objects and, consequently, contains all the information necessary to create the client module.

When building the Factory, the server module associates a unique ticket number to each download of the client module and insert the respective JSessionID on the Factory component. At the client-side, we use application cookies to keep JSessionID and the first ticket obtained. Then, when starting up, the client module verifies if there is some

information for WELFIT and, if this is the case, reuses it.

If it is the first time the user sees the website being evaluated by WELFIT, then the invitation is presented to the user on the top of the pages and the logger remains stopped until the user accepts to participate. In this study, we attached to the participation acceptance, a question asking if the user is using assistive technology. This information is used at mining phase and will be detailed in the next section.

Finally, when the user submits the invitation form, the answers and the time the session started are recorded in application cookies and are sent to the server, which associates the session and ticket information with these data. Thus, if the user accepted to participate, the client module starts to record all triggered events defined for standard Javascript. Therefore, if the user accesses other page of a website that s/he agreed to participate, then the capture starts as soon as the client module is loaded.

During the event capture, all events are put into a package of configurable limited size. Then, as soon as the package being built reaches the configured limit, it is sent to the server. Moreover, each package is identified by a package counter, the ticket being used by the user, and the respective JSessionID. Each event that is part of the event stream is time stamped in milliseconds from the session start.

The limit used for the packages is 1 kilobyte and the limit of packages that are kept on the client is 20. Then, in order to follow the requirements in [29] and design the client module as lightweight as possible, we used the DOM (Document Object Model) tree structure to store the packages.

The client-server communication is performed according to the following policy: the client modules ask the server to store the logs and it answers the requests. Then, at a defined clean up cycle, the server responses are checked by the client module. The clean up cycle used was 10 seconds or 10 accumulated packages (i.e., not deleted at client due to some error or network delay).

At the client module, as soon as a package completes the size limit, it is sent to the server using the Dynamic Script Tag approach, which allows cross-domain AJAJ (Asynchronous Javascript And JSON). Details and the complete specification and justification of the client module's implementation can be found in [28].

At the server module, as soon as it receives a client module request, it verifies the referrer and the information sent. Then, we use the following server answers: DENIED@pack_number for packages coming from not registered websites, ERROR@pack_number if some error occurred during the data storage, and OK@pack_number if the package was recorded successfully.

When the client receives the server's answer, the module stores it and, in the next clean up cycle, the confirmed packages are deleted, the error packages are resent, and, if the denied message is received, the client module halts. Another halt situation occurs if

the user's bandwidth connection does not support the amount of recorded data, what will cause the 20 accumulated packages limit to be reached.

As a strategy to avoid losing the last packages of logged data, we set the Window's unload event handler to trigger the cleanup cycle and then resend pending packages. This means that, when the user is leaving a page, the client module tries to send unconfirmed packages again.

For users who do not use screen readers, the invitation appears at the top of the evaluated Web pages and, once the form is answered, the invitation is reduced to a small widget at the same position that the invitation form. This widget shows the tool's status (i.e., recording or stopped) and provides controls to stop/start the capture if the user decides to cancel the participation. In addition, screen reader users need to be aware of the evaluation and receive the invitation in similar manner. Regarding this, we allow the configuration of the *tabindex* of the invitation, so by setting the minimum value of *tabindex* for the form invitation, the user can perceive it as one of the first page elements. Thus, after the submission of the invitation, the widget being presented does not use the *tabindex* value, since it could represent a barrier for screen reader users on the consecutive accesses.

This model showed to be effective and efficient, since it does not interfere with the UI usage for users with average bandwidth greater than 1 kilobyte per second, which is the average amount of UI event log of compressed data found in the empirical results.

5.3.3 Web Usage Mining

WUM is "the process of discovering and interpreting patterns of user access to the Web information systems by mining the data collected from user interactions with the system" [69]. In the WELFIT project, WUM is used to classify sessions as coming from assistive technology or not, and to group the walks in a way to identify usage patterns and UI design problems, presenting reports visually to specialists.

The use of WUM in the project aims at reusing ideas already applied to server logs in client-side event logs, aiming to not depend on specific task models or grammars to infer high level actions from UI events.

While server logs studies use the main elements as Web pages, we split session evaluation to a detailed page-view evaluation. Thus, instead of page elements we used the concatenation of a certain event in a certain page element, with a certain ID or name, if this is the case. With this approach, it was possible to represent data in a consistent and reduced way.

Before applying WUM techniques on the logged data, we searched for methods, techniques, and algorithms to use as an experiment starting point. Thus, to discover clusters

with arbitrary shape we used distance measures [40]. Furthermore, distance-based experiments use initially Euclidean distance. However, when using Euclidean distance-based measures between two sessions' event flows, some structural information regarding the order in which the events occur may not be well reflected [42]. Moreover, the order of visited pages reveals important information for the purpose of supporting and increasing user satisfaction [42].

In an effort to take into account the structural information present on event flows, the approach we follow is based on SAM (Sequence Alignment Method) [42]. SAM, also known as Edit Distance, is a distance-based measure that gives a score representing the amount of operations necessary to transform a source sequence ($x[1..m]$) into a target sequence ($y[1..n]$) [42, 20].

Some allowed operations to transform $x[1..m]$ into $y[1..n]$, using an auxiliary array z, are the following [20]:

1. Copy a character from x to z by setting $z[j] = x[i]$ and then incrementing both i and j.

2. Replace a character form x by another character c, by setting $z[j] = c$, and then incrementing both i and j.

3. Delete a character from x by incrementing i but leaving j alone.

4. Insert the character c into z by setting $z[j] = c$ and then incrementing j, but leaving i alone.

5. Twiddle (i.e., exchange) the next two characters by copying them from x to z but in the opposite order; we do so by setting $z[j] = x[i + 1]$ and $z[j + 1] = x[i]$ and then setting $i = i + 2$ and $j = j + 2$.

From these operations, the SAM distance we used is calculated as the sum of the number of the operations, and each of them multiplied by a respective weight. We assume that the weight of copy operation is 0 so that, if two sequences are equal, then the SAM distance between them is 0. Moreover, we also assume that the weight of replace operations (w_r) is less than or equal the sum of deletion (w_d) and insertion weight (w_i), since it represents a deletion and an insertion (5.2). For the same reason, we assume that the weight of twiddle operations (w_t) is less than or equal 2 times the replace weight (5.3). Finally, the values we used are: $w_d = 1, w_i = 1, w_r = 2$, and $w_t = 3$, so that the operation of replacing is equal to a deletion and a insertion, and twiddle are $\frac{3}{4}$ of 2 replacements (5.1).

$$d_{SAM}(S_1, S_2) = w_d D + w_i I + w_r R + w_t T. \tag{5.1}$$

$$w_r \leq w_d + w_i \tag{5.2}$$

$$w_t \leq 2w_r \tag{5.3}$$

To classify the sessions as coming from technology assistive users or not, we calculated, for each page, the edit distance for all pairs of sessions (i.e., $d_{SAM}(S_i, S_j)$ for all i and j). It is worth noting that, for each of these pairs, this computation is made just once, and then the calculated distance is recorded in the tool's database. Thus, we find the maximum distance (d_{max}) between the most opposites event streams (say S_a and S_b), then all streams Si having $d_{SAM}(S_a, S_i) < \frac{d_{max}}{2}$ are grouped with S_a. In doing so, all streams S_i having $d_{SAM}(S_b, S_i) < \frac{d_{max}}{2}$ are grouped with S_b.

Once the groups are formed, we check the support in each one to be flagged as the assistive technology group, based on the number of participants that answered that are using assistive technology. These groups represent that the users in that group used the UI and triggered sequences of events in a similar way.

There are different possible sequences of operations to transform the source into the target [20]. The order of operations applied to sequences when calculating the d_{SAM} is copy, twiddle, replace, delete, and insert.

We compute the unknown distances as the administrator requests a report, since the known distances are already in the tool's database. However, for websites with big volume of sessions and users, this approach may not be effective and deserve more study to allow this classification to occur in real time.

From now on, we refer to digraphs for each Web page containing the event flows occurred in the respective page. Each of these digraphs contains two artificial nodes, namely start and end. As already mentioned before, each node is the concatenation of the element tag, element name or ID, and event. For example, a node representation for a focus event at the WELFIT's invitation form is: *form, welfit, focus*. Finally, in these digraphs we use the events flow as walk in graphs, as repeated nodes and cycles contain relevant information to the tool, since they may indicate UI design problems and/or usage patterns.

To identify UI design problems in a digraph of a certain page it would be necessary to search for significant distances between different walks up to a certain node, for each node. Although this could bring interesting results, it would be impractical to apply it on the fly, as the administrator requests a report. Due to this, we defined a heuristic that tries to get some properties of the SAM using the average distance to a certain digraph node as a metric, evaluating how this average distance changes in the outgoing nodes. Thus, differences in these distances will result in more delete and insert operations in a d_{SAM} comparison.

The SAM-based heuristic works as follows. First, consider G the digraph representing page elements of a certain page P and a certain node of G, say u, has an average distance

d_u. Additionally, consider u's neighbors $v_0, v_1, ..., v_k$, and $d_{v_0}, d_{v_1}, ..., d_{v_k}$ the respective average distances.

Since u is in all walks passing through u, then we assume that from root start node to the previous nodes of u, they are part of the same walk and the edit distance does not indicate a problem.

Thus, if $d_u > d_{v_i}$, then the in degree of v_i is greater than 1, meaning that there exists some alternative walk to v_i that does not have u as precedent and has a lesser average distance. Thus, the walk from the root passing through u to reach v_i is not the short one, what may indicate that u is part of a walk containing an accessibility barrier or accessibility problem (Figure 5.2).

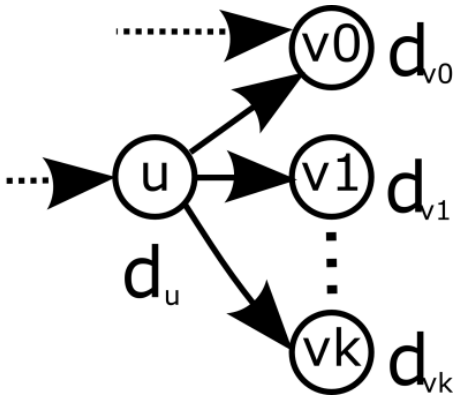


Figure 5.2: Properties used by the SAM-based heuristic. If $d_u > d_{v_0}$, then v_0 has another incident edge coming from a node different from u and with a less average distance.

Then, if d_u is greater than the average of $d_{v_0}, d_{v_1}, ..., d_{v_k}$ plus the standard deviation, which is the defined threshold, then it represents a significant difference at node u. This heuristic allows the approximated comparison of distances between more than two walks at every node. Thus, it is possible to find significant average distances that may indicate UI design problems.

5.3.4 Data visualization

WELFIT reports are visual and this is why the data visualization plays an important role for the system. From the graph structure used by the tool keeping all WUM results is possible to generate graph representations in different formats. The report generation was made through the experimentation of different graphing engines and structures to represent usage patterns and UI events. The software used to generate PDF (Portable Document Format) reports is Graphviz [1], an open source graph visualization software that counts on different graph engines and auxiliary tools.

The communication between WELFIT server module and the Graphviz is made by the graph source document, using the Dot language, passed to the graphing tool, which in turn returns the reports. The Graphviz's tested engines were Dot, Neato, FDP, and Circo. The most readable results were returned by Dot engine, which is the one WELFIT is currently using. However, for Web pages having a reasonable number of page elements the presentation has poor usability. Due to this and to the client-side log format [28], we clustered graph nodes by elements in which they were triggered.

5.4 Preliminary results

The real data used in this work were captured during evaluation of the website of the research group called Todos Nós (www.todosnos.unicamp.br). This website was chosen because part of its audience uses assistive technology. The data captured during 60 days resulted 85 recorded sessions, 6 of them coming from assistive technology users. All sessions are related to 269 different URLs, and 278,450 log lines. The results of this work point in two main directions: the viability of capturing data at UI event level during real use and identifying usage patterns from the collected data. The data captured using Javascript and the plug-in independent technique does not interfered with the usage, respecting the requirements elicited in [29] for data logging.

WELFIT's reports show usage patterns and strategies that users applied when interacting with Web pages elements. Moreover, with these information it is possible to identify possible usability and/or accessibility issues highlighted by the tool (Figure 5.3).

5.4.1 Discussion

The system and the model presented in this work open new possibilities of application of WUM techniques commonly used to extract novel information from server-side logs using client-side event logs, and to evaluate how users interact with UI elements of Web pages. However, in contrast to these contributions the tool and approach presented here still have some limitations.

The tool's visual reports require that a specialist knows events and Web page elements in order to understand the report. Moreover, since the graphs drawn by the system use page elements identified by an ID or name to create clusters of nodes, the lack of these IDs may make difficult the task of the specialist to find and change the specific page element. This situation is caused because using this approach the graphs are more compact and will represent how users interact with that type of element; as an alternative, the tool could generate IDs based on the element's position in the DOM tree, which would result in crowded graphs and making even more difficult the task of the graph drawing tool

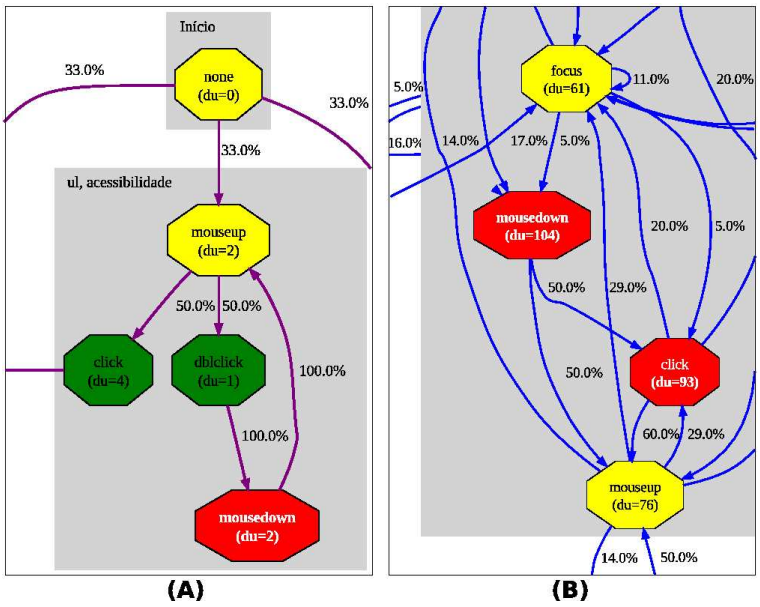


Figure 5.3: Patterns of accessibility from users: A) Attempts to click in a list element of accessibility toolbar; B) Highlighted click node shows attempts of clicking in an element.

used.

The issues found when drawing large graphs are related to the size, number of clusters, number of nodes, and, specially, the number of edges. In sessions containing many events it is necessary to apply filters to reduce restrictions of the drawing tool. The filters used are transition percentage between nodes and/or absolute number of transitions between nodes. With these filters we could show the more frequent patterns, but applying some of then may blur the overall usage flow, since the filtered edges may fragment the graph, which may not containing all users' flows from start to end.

The classification we used initially considered two groups, but more detailed information about these groups can be obtained by grouping them into more correlated ones. Thus, as future investigations we should considered other goals as, for example, divide the group until it reaches a specified support of 70%. This may lead to flows groups of more similar sequences of events.

Finally, in contrast to server logs studies, tracking session of UI event data allows the use of cookies and thus facilitates the task of reconstructing users' sessions [14].

5.5 Conclusion

The presented tool implements the developed model to capture client-side Web data and identify usage flows, highlighting patterns and possible A&U issues. In addition,

WELFIT focused in filling the requirements and gaps presented in [29]. The tool focus on the integration of A&U for the target audience, allowing remote testing during real use, interfering with the Web page presentation and functioning as minimum as possible. The tool avoids that the logs processing and logs transmission tasks interfere with UI usage though the use of lightweight and asynchronous techniques. It also provides tool's status and tool's controls and does not depend on specific plug-in on client's device. Finally, WELFIT uses WUM techniques to reveal usage patterns without depending on specific task models or grammars.

Usage patterns showed sequences of actions of participants that inform they were using assistive technology and that it was possible to improve the UI design and provide personalization features like automatic resizing or defining *tabindex* based on navigation patterns found.

Beyond the focus used in this study, WELFIT's model allows other focuses as evaluating websites performance according to specific users' characteristics (e.g., bandwidth connection, user agent, etc) and/or analysing other attributes present in the client-side event logs (e.g., mouse coordinates, shortcuts used, average time elapsed).

Capítulo 6

Conclusões

A Web conta com serviços cada vez mais importantes para os cidadãos. No entanto, a maioria dos *websites* não oferece serviços acessíveis para todo seu público. Decisões tomadas nas etapas de projeto de IU podem não considerar alguns contextos de uso ou usuários com deficiências específicas. Complementarmente, identificar barreiras de acessibilidade e problemas de usabilidade em *websites* depende da avaliação de código e de *logs* resultantes da utilização real do *website*.

A identificação de padrões de utilização da Web durante tarefas reais, utilizando uma configuração comum de software e hardware, é um ponto promissor para se obter dados sobre a utilização real da IU e informações sobre quais ações foram efetuadas pelos usuários durante a utilização de um *website*.

A avaliação de *websites* considerando dados reais envolve grandes quantidades de dados. Quando os dados utilizados são *log* de eventos disparados no lado do cliente esse volume é ainda maior, uma vez que um simples *page-view* representado em uma linha de *log* de servidor Web pode resultar em uma seqüência de centenas de eventos. O desenvolvimento de ferramentas que possam lidar com grandes quantidades de dados deve levar em consideração diversos aspectos desde os recursos computacionais necessários para executar tarefas de captura e transmissão, até aqueles que impedem que o funcionamento da ferramenta de captura atrapalhe o usuário durante a execução de sua tarefa.

Nesta dissertação foi apresentada a especificação de um conjunto de requisitos para ferramentas de captura de eventos disparados no lado do cliente. Segundo esses requisitos, foi definido um modelo contendo dois módulos, sendo um responsável pela captura de dados de eventos no lado do cliente e outro pela avaliação dos *logs* destes eventos. Por fim, foi desenvolvida uma ferramenta materializando o modelo e demonstrando como é possível capturar, transmitir e persistir *logs* de eventos disparados no lado do cliente, assim como obter informações de mais alto nível sem necessitar de gramáticas específicas para inferir problemas de design de IU.

A especificação dos requisitos apresentada no capítulo 2 considerou diferentes cenários e buscou tornar o funcionamento do sistema o mais transparente possível, apresentando controles e *feedback* necessários para os participantes. Ainda, foram levantadas características que podem fortalecer ferramentas de avaliação, tornando-as mais robustas, menos custosas e mais fáceis de usar e reusar. Foi possível identificar que as ferramentas avaliadas lidam principalmente com requisitos relacionados à plataforma de TI, deixando de endereçar requisitos relacionados às funções humanas.

O modelo de captura de eventos definido no capítulo 3 possibilita que avaliações semelhantes às apresentadas neste trabalho possam ser feitas em diversos dispositivos como celulares, *palmtops*, *video games* e *set-top boxes* utilizados em aplicações interativas para TV Digital, permitindo que a captura não interfira na utilização do *website* desde que conte com uma conexão que comporte a sobrecarga relativa à transmissão dos dados capturados, o que no contexto da Web representa uma média de 1kB por segundo para transmitir pacotes de *logs*.

Extrair informações de alto nível a partir do volume de dados obtido por ferramentas que utilizam *logs* de eventos disparados no lado do cliente continua sendo um desafio. Assim, técnicas que apóiem a extração e sumarização de padrões de utilização, como a apresentada no capítulo 4, podem ajudar na identificação de barreiras de acessibilidade e/ou problemas de usabilidade. A técnica apresentada é baseada em uma medida de distância utilizando as cadeias de eventos e considerando a ordem em que ocorreram, indicando diferenças significativas de performance até certo evento em certo elemento, o que pode indicar problemas de design e/ou barreiras de acessibilidade. Adicionalmente, foi apresentado como gerar gráficos representando os grafos de utilização com o software de geração de grafos chamado Graphviz.

Com a utilização da ferramenta apresentada no capítulo 5 foi possível demonstrar a viabilidade de capturar eventos disparados em páginas Web, no lado do cliente. Com isso possibilita-se envolver usuários de tecnologias assistivas em seus ambientes cotidianos de utilização, com configurações de software e hardware comuns.

A partir dos gráficos representando padrões de utilização é possível identificar estratégias utilizadas pelos usuários. Assim, uma vez que a seqüência em que os eventos são disparados representa como os usuários interagem com a IU, é possível aplicar estudos sobre a eficácia de decisões de design. Os dados capturados pela ferramenta representam 85 sessões de utilização do *website* do grupo de pesquisa Todos Nós. Deste número, 6 sessões são de usuários que informaram que estavam utilizando tecnologias assistivas durante a utilização do *website* estudado.

Com a avaliação continuada apoiada pelo WELFIT é possível identificar padrões de utilização da Web mediada por tecnologias assistivas, complementando outros tipos de avaliação. Este tipo de ferramenta tem um novo papel em relação à avaliação, uma vez

que seu objetivo é olhar como um determinado *website* é utilizado, de maneira isolada de quem usa e de quem desenvolve *websites*. Ainda, os resultados obtidos incitam outros estudos estatísticos sobre os dados capturados (e.g., tempo, coordenadas de movimentos do *mouse*, teclas de atalho) e abrem caminho para outros estudos como utilizar os padrões de uso identificados para incorporar recursos de personalização.

A partir das contribuições deste trabalho é possível aplicar estudos detalhados sobre como usuários de tecnologias assistivas usam *websites*, fomentando, por exemplo, o desenvolvimento de interfaces ajustáveis tendo como base os dados coletados e padrões identificados. Além disso, é possível analisar como usuários acessam o *website* enquanto executam tarefas reais e em ambientes cotidianos de utilização. A ferramenta contribui para o desenvolvimento de interfaces de usuário que sejam mais acessíveis e usáveis por todos indiscriminadamente.

Os relatórios resultantes do uso da ferramenta, contendo o gráfico representando o fluxo dos eventos, exigem um especialista conhecedor de eventos e elementos de páginas Web para compreender o grafo de utilização. Complementarmente, os grafos podem ser extensos e a tarefa do especialista de identificar problemas e entender o fluxo de eventos como um todo pode ser difícil dependendo da quantidade de dados e do algoritmo utilizado para gerar grafos.

Os grafos desenhados pelo sistema utilizam o ID ou nome dos elementos para identificar vértices desses grafos; a falta desses dados de identificação pode tornar mais difícil a tarefa de um especialista em encontrar um elemento específico para corrigir um problema. Esta estratégia torna os grafos de utilização mais compactos e representam como usuários interagem com certo tipo de elemento. Para estudos em que se deseja obter informações mais detalhadas sobre como usuários interagem com cada elemento, é possível gerar IDs automaticamente com base no posicionamento de cada elemento na árvore DOM. No entanto, isto resultaria em grafos mais densos e tornaria ainda mais difícil a tarefa da ferramenta responsável por desenhar grafos e do especialista em navegar nesses grafos.

Alguns problemas encontrados durante a geração das representações gráficas de grandes grafos de utilização (i.e., contendo centenas de nós e arestas) estão relacionados ao tamanho, número de elementos da respectiva página, número de vértices e, especialmente, o número de arestas. Existem algumas ferramentas desenvolvidas para desenhar grafos grandes, mas as ferramentas encontradas não permitem que os grafos contenham ciclos, clusters, ou rótulos nos vértices ou arestas. Assim, um dos trabalhos futuros envolve testes com outras ferramentas geradoras de grafos, aumentando a eficiência e eficácia com que o sistema apresenta as imagens contendo os grafos de utilização.

Para lidar com grafos grandes foram incorporados filtros para facilitar a tarefa do software gerador de grafos utilizado. Os filtros são: porcentagem de transições entre vértices e/ou número absoluto de transições entre vértices. Com eles é possível apresentar

padrões mais freqüentes de eventos; entretanto, certas combinações podem dividir o grafo em um número grande de componentes, impossibilitando a visualização do fluxo de eventos do começo ao fim.

A classificação utilizada neste trabalho considerou dois grupos, mas informações mais detalhadas sobre esses grupos poderiam ser obtidas através da divisão desses grupos em outros mais fortemente correlacionados. Então, como estudo futuro pode ser considerado como objetivo obter grupos contendo seqüências mais fortemente relacionadas ou dividir grupos até que alcancem um suporte específico de 70%, levando em consideração questões de desempenho e o tempo levado pela ferramenta para retornar o grafo de utilização para os avaliadores.

A próxima utilização da ferramenta resultante desta dissertação será feita no *website* Vila na Rede[1], que é um espaço para a comunidade de moradores da Vila União, bairro de Campinas, compartilhar produtos, serviços, eventos e idéias, levando para a Web a rede social já existente. O Vila na Rede é um sistema piloto desenvolvido pelo grupo de pesquisa e-Cidadania, que tem como objetivo estudar e propor soluções aos desafios do design da interação e IU em sistemas relacionados ao contexto do exercício de cidadania.

As principais contribuições deste trabalho são os requisitos para ferramentas de avaliação de *websites* baseadas em *logs* de eventos, especificação e implementação de um modelo de identificação de padrões de utilização da Web e a demonstração da viabilidade de se capturar dados no lado cliente durante a utilização real. Finalmente, o trabalho apresentado nesta dissertação contribui para a área de IHC ao tratar questões relevantes à avaliação de IU. O sistema desenvolvido auxilia na identificação de diferenças significativas entre cadeias de eventos, possibilitando ao especialista a identificação de padrões de utilização e problemas de design. O foco na identificação de fluxos de eventos que podem indicar barreiras para usuários de tecnologias assistivas é um dos principais diferenciais da proposta de solução apresentada, uma vez que coletar dados contendo a mesma representatividade em relação ao público alvo e tarefas seria mais difícil caso fossem realizados testes em ambientes controlados envolvendo tarefas pré-determinadas.

[1] `http://www.vilanarede.org.br`

Apêndice A

Casos de uso

A.1 Convida usuário

A.1.1 Informações características

Objetivo no contexto: Apresentar convite de participação aos usuários do *website* em avaliação.

Escopo: Módulo cliente.

Nível: Tarefa primária.

Pré-condições: Website em avaliação utiliza o módulo cliente do WELFIT, está cadastrado no sistema e o Javascript está habilitado no computador do usuário.

Pós-condição de sucesso: Convite é apresentado ao usuário.

Pós-condição de falha: Convite não é apresentado e, conseqüentemente, a captura não pode ser iniciada.

Ator primário: Módulo cliente.

Gatilho: Módulo cliente é carregado no dispositivo do usuário.

A.1.2 Cenário principal de sucesso

1. Módulo cliente apresenta o convite de participação (Figura A.1) com as perguntas "Aceita participar da avaliação deste *website*?" e "Você está usando alguma tecnologia assistiva?" e com o link "Mais informações", que apresenta mais informações sobre o funcionamento da ferramenta e sobre a avaliação.

2. Usuário acessa o convite.

Figura A.1: Convite apresentado aos usuários do *website* em avaliação.

A.1.3 Extensões

2.a - Usuário acessa link "Mais informações" : Um breve texto é apresentado ao usuário explicando o funcionamento da ferramenta e como a avaliação é feita, incluindo questões de e privacidade.

A.1.4 Informações relacionadas

Prioridade: Alta.

Meta de performance: Menos de 5 segundos para o convite ser apresentado.

Freqüência: Uma vez para cada usuário que visitar o *website* em avaliação, até que ele responda.

Canal até ator primário: Navegador.

A.1.5 Agendamento

Previsão: Versão 1.0

A.2 Responde convite

A.2.1 Informações características

Objetivo no contexto: Responder ao convite de participação.

Escopo: Módulo cliente.

Nível: Tarefa primária.

Pré-condições: Convite está acessível ao usuário.

Pós-condição de sucesso: Módulo cliente inicia captura de eventos.

Pós-condição de falha: Usuário não identifica o convite ou o ignora e inicia a utilização do *website* sem participar da avaliação.

Ator primário: Usuário.

Gatilho: Apresentação do convite.

A.2.2 Cenário principal de sucesso

1. Usuário aceita o convite.

2. O módulo cliente grava informações (e.g., número de *ticket*, respostas) do usuário em *cookie* de aplicação e inicia o registro de eventos.

3. A região da interface utilizada para apresentar o convite é então reduzida para apresentar o status da ferramenta (e.g., gravando, parado) e disponibilizar mecanismos para o usuário interromper a captura a qualquer momento (Figura A.2).

Figura A.2: Convite é reduzido para apresentar o status da ferramenta.

A.2.3 Extensões

1.a - Usuário recusa convite : Módulo cliente fica inativo e convite é fechado.

A.2.4 Informações relacionadas

Prioridade: Alta.
Meta de performance: Cerca de 10 segundos.
Freqüência: Uma vez para cada usuário que visitar o *website* em avaliação.
Canal até ator primário: Região superior das páginas do *website* em avaliação.

A.2.5 Agendamento

Previsão: Versão 1.0.

A.3 Captura eventos

A.3.1 Informações características

Objetivo no contexto: Capturar eventos disparados no lado do cliente.
Escopo: Módulo cliente.
Nível: Tarefa primária.
Pré-condições: Usuário aceitou participar da avaliação.

Pós-condição de sucesso: Eventos ocorridos no lado do cliente são capturados e gravados em memória.

Pós-condição de falha: Módulo cliente não captura eventos.

Ator primário: Módulo cliente.

Gatilho: Convite é aceito pelo usuário.

A.3.2 Cenário principal de sucesso

1. Módulo cliente captura eventos disparados no lado do cliente.
2. Módulo cliente monta a linha de *log* correspondente ao evento capturado.
3. Módulo cliente grava linha de *log* em memória.

A.3.3 Informações relacionadas

Prioridade: Alta.

Meta de performance: Execução em poucos milissegundos.

Freqüência: Durante toda a sessão de utilização do *website*, a cada evento ocorrido no lado do cliente.

Canal até ator primário: Páginas Web.

A.3.4 Agendamento

Previsão: Versão 1.0

A.4 Compacta *log*

A.4.1 Informações características

Objetivo no contexto: Compactar eventos capturados e gravá-los em um pacote.

Escopo: Módulo cliente.

Nível: Sub-função.

Pré-condições: Evento foi capturado e gravado em memória.

Pós-condição de sucesso: Linha de *log* é compactada e gravada no pacote sendo construído.

Pós-condição de falha: Linha de *log* é gravada sem compactação.

Ator primário: Módulo cliente.

Gatilho: Um evento é gravado em memória.

A.4.2 Cenário principal de sucesso

1. Linha de *log* é montada com o evento capturado, seja ela a linha i.
2. Grava-se o conteúdo de i em uma variável auxiliar.
3. Compara-se i com a linha gravada anteriormente no pacote (em memória), seja ela a linha $i - 1$, e todos os dados repetidos em i são trocados por vazio (i.e., “”).
4. Agora, a linha i pode ser compactada novamente substituindo caracteres que se repetem na linha de *log* por uma representação desse caractere imediatamente seguido por uma marcação representando o número de vezes que ele se repete. Exemplo: o delimitador de campos pode se repetir várias vezes para campos vazios, então, ele pode ser combinado com um número que representa seu número de ocorrências (e.g., o trecho “,,,,,,,,,,”, de 10 bytes, passa a ser “,{10}”, agora com 5 bytes).
5. $i - 1$ recebe o conteúdo da variável auxiliar.
6. Calcula-se o tamanho do pacote.
7. Se as i linhas de *log* formam um pacote de tamanho menor que o limite configurado (*default* 1kB), a i-ésima linha é adicionada ao pacote atual.

A.4.3 Extensões

1.a - Se linha i é a primeira do pacote : linha de *log* é gravada na integra.

5.a - Se as i linhas de *log* formam um pacote maior que tamanho configurado, um pacote contendo i-1 linhas é está pronto para ser transmitido e a i-ésima é inserida na íntegra em um novo pacote.

A.4.4 Informações relacionadas

Prioridade: Alta.

Meta de performance: Execução em poucos milissegundos.

Freqüência: Durante toda a sessão de utilização do *website*, a cada evento ocorrido no lado do cliente.

Canal até ator primário: Dados em memória.

A.4.5 Agendamento

Previsão: Versão 1.0

A.5 Envia *log* para o servidor

A.5.1 Informações características

Objetivo no contexto: Enviar dados armazenados em memória para o módulo servidor.

Escopo: Módulo cliente.

Nível: Tarefa primária.

Pré-condições: Pacotes estão gravados em memória.

Pós-condição de sucesso: Pacote é transmitido para o módulo servidor e o espaço que era ocupado pelos pacote é então liberado para novos pacotes.

Pós-condição de falha: Pacote deve ser retransmitido.

Ator primário: Módulo cliente.

Gatilho: Pacote é criado ou 10 pacotes acumulados na máquina do cliente.

A.5.2 Cenário principal de sucesso

1. Módulo cliente envia, de maneira assíncrona, os dados gravados para o módulo servidor, solicitando implicitamente o armazenamento dos pacotes.

2. Para cada pacote recebido o módulo servidor responde ao módulo cliente que a gravação foi bem sucedida.

3. Para cada resposta positiva vinda do servidor, o módulo cliente libera o espaço correspondente ao pacote gravado no servidor. A resposta do servidor deve conter o nome do pacote gravado, por exemplo, OK@[nome do pacote].

A.5.3 Extensões

3.a - Se a resposta do módulo servidor for negativa (e.g., ERROR@[nome do pacote]) : módulo cliente não deve apagar o respectivo pacote, pois vai reenviá-lo em ciclo definido.

3.b - A cada ciclo de 10 segundos ou 10 pacotes acumulados no cliente : módulo cliente deve reenviar pacotes dos quais não recebeu resposta positiva.

A.5.4 Informações relacionadas

Prioridade: Alta

Meta de performance: Envio de *logs* e processamento das respostas em cerca de 2 segundos.

Freqüência: Ao montar pacote ou se 10 ou mais pacotes estiverem acumulados.

Canal até ator primário: Dados em memória.

Atores secundários: Módulo servidor.
Canal até o ator secundário: HTTP.

A.5.5 Agendamento

Previsão: Versão 1.0.

A.6 Grava *log*

A.6.1 Informações características

Objetivo no contexto: Gravar os pacotes de *logs* no módulo servidor do WELFIT.
Escopo: Módulo servidor.
Nível: Tarefa primária.
Pré-condições: Módulo cliente envia pacote para o módulo servidor e módulo servidor está ativo e sem falhas.
Pós-condição de sucesso: Pacote é gravado e módulo servidor envia resposta para o módulo cliente.
Pós-condição de falha: Módulo servidor informa ao módulo cliente que houve erro na gravação dos *logs* enviados.
Ator primário: Módulo servidor.
Gatilho: Módulo servidor recebe dados do módulo cliente.

A.6.2 Cenário principal de sucesso

1. Módulo servidor recebe pacote do módulo cliente.
2. Módulo servidor verifica se o *website* ao qual o pacote se refere está cadastrado.
3. Módulo servidor grava o pacote e envia confirmação (e.g., OK@[nome do pacote]) para o módulo cliente, que pode então descartar o pacote que acabou de ser gravado.

A.6.3 Extensões

2.a - Se o *website* remetente não estiver cadastrado : módulo servidor envia a mensagem DENIED@[nome do pacote] para o módulo cliente, que então interrompe a captura no respectivo *website*.

3.a - Se houver algum erro na gravação : módulo servidor envia mensagem de erro (e.g., ERROR@[nome do pacote]) para o módulo cliente, que será responsável por reenviar o pacote que não foi gravado.

A.6.4 Informações relacionadas

Prioridade: Alta.

Meta de performance: Gravação de pacote e envio das respostas em menos de 1 segundo.

Freqüência: A cada segundo de utilização do *website* avaliado.

Canal até ator primário: Sistema de arquivos ou banco de dados.

Atores secundários: Módulo cliente.

Canal até o ator secundário: HTTP.

A.6.5 Agendamento

Previsão: Versão 1.0.

A.7 Autenticação

A.7.1 Informações características

Objetivo no contexto: Autenticar no sistema.

Escopo: Módulo servidor.

Nível: Tarefa primária.

Pré-condições: Usuário está cadastrado.

Pós-condição de sucesso: Usuário está autenticado para utilizar o sistema.

Pós-condição de falha: Usuário não pode acessar áreas restritas do sistema.

Ator primário: Administrador.

Gatilho: Acessar qualquer página de áreas restritas do sistema.

A.7.2 Cenário principal de sucesso

1. Usuário preenche e-mail e senha e envia o formulário de autenticação.

2. Sistema devolve para o usuário validado a página inicial da administração da ferramenta contendo os *websites* associados a ele, as opções para editar informações de cada *website* (e.g., restringir a captura de *logs* em regiões do *website*) e solicitar relatórios.

A.7.3 Extensões

2.a - Usuário não está cadastrado no sistema ou senha incorreta : Sistema devolve para o usuário uma página informando o ocorrido.

A.7.4 Informações relacionadas

Prioridade: Alta.
Meta de performance: Menos de 10 segundos.
Freqüência: Uma vez por sessão para cada administrador que acessar o sistema.
Canal até ator primário: Página de autenticação do sistema.

A.7.5 Agendamento

Previsão: Versão 1.0.

A.8 Cadastra administrador

A.8.1 Informações características

Objetivo no contexto: Cadastrar administrador.
Escopo: Módulo servidor.
Nível: Tarefa primária.
Pré-condições: E-mail do novo administrador não está cadastrado no sistema.
Pós-condição de sucesso: Cadastro do novo administrador está validado no sistema.
Pós-condição de falha: Cadastro não é concluído.
Ator primário: Administrador de *websites*.
Gatilho: Usuário acessa a página de cadastro de administrador de *websites*.

A.8.2 Cenário principal de sucesso

1. Novo usuário preenche nome, e-mail e senha.
2. Sistema verifica se e-mail não está cadastrado e validado.
3. Novo usuário recebe a confirmação do cadastro.

A.8.3 Informações relacionadas

Prioridade: Média.
Meta de performance: Menos de 1 minuto.
Freqüência: Uma vez para cada administrador cadastrado.
Canal até ator primário: Página de cadastro e e-mail de validação de e-mail.

A.8.4 Agendamento

Previsão: Versão 1.0.

A.9 Atualiza administrador

A.9.1 Informações características

Objetivo no contexto: Atualizar informações de administradores da ferramenta.

Escopo: Módulo servidor.

Nível: Tarefa primária.

Pré-condições: Administrador está autenticado.

Pós-condição de sucesso: Dados do administrador estão atualizados.

Pós-condição de falha: Dados do administrador não sofrem alterações.

Ator primário: Administrador.

Gatilho: Administrador acessa página de edição de informações de administrador da ferramenta.

A.9.2 Cenário principal de sucesso

1. Administrador altera suas informações (e.g., nome, e-mail, senha).

2. Administrador confirma alterações.

3. Sistema valida informações e apresenta mensagem de confirmação para o usuário.

A.9.3 Informações relacionadas

Prioridade: Média.

Meta de performance: Menos de 1 minuto.

Freqüência: Não é possível especificar.

Canal até ator primário: Página de edição de informações de administradores.

A.9.4 Agendamento

Previsão: Versão 1.0.

A.10 Cadastra *website*

A.10.1 Informações características

Objetivo no contexto: Cadastrar *websites* para que o WELFIT os avalie.
Escopo: Módulo servidor.
Nível: Tarefa primária.
Pré-condições: Administrador está autenticado e *website* não está cadastrado.
Pós-condição de sucesso: Website está cadastrado.
Pós-condição de falha: Website não foi cadastrado.
Ator primário: Administrador.
Gatilho: Administrador acessa página de cadastro de *websites*.

A.10.2 Cenário principal de sucesso

1. Administrador preenche informações (Nome, URL do nível mais abrangente e expressão regular para bloquear a captura de *logs* em algumas áreas do *website* avaliado, por exemplo, `^http:\/\/www\.exemplo\.com\.br\/.*\/login.html$`) e confirma.

2. Sistema verifica informações preenchidas.

3. Sistema retorna mensagem de confirmação para o administrador.

A.10.3 Informações relacionadas

Prioridade: Alta.
Meta de performance: Menos de 1 minuto.
Freqüência: Aproximadamente uma vez para cada administrador cadastrado.
Canal até ator primário: Página de cadastro de *websites*.

A.10.4 Agendamento

Previsão: Versão 1.0.

A.11 Atualiza *website*

A.11.1 Informações características

Objetivo no contexto: Atualizar dados de *websites* cadastrados no sistema.

Escopo: Módulo servidor.
Nível: Tarefa primária.
Pré-condições: Administrador está autenticado e *website* está cadastrado.
Pós-condição de sucesso: Dados do *website* estão atualizados.
Pós-condição de falha: Dados do *website* não são alterados.
Ator primário: Administrador.
Gatilho: Administrador acessa página de edição de informações de *website*.

A.11.2 Cenário principal de sucesso

1. Administrador altera informações (Nome, URL do nível mais abrangente e expressão regular para bloquear a captura de *logs* em algumas áreas do *website* avaliado, por exemplo, `^http:\/\/www\.exemplo\.com\.br\/.*\/login.html$`) e confirma.
2. Sistema verifica informações preenchidas.
3. Sistema retorna mensagem de confirmação para o administrador.

A.11.3 Informações relacionadas

Prioridade: Alta.
Meta de performance: Menos de 1 minuto.
Freqüência: Não é possível especificar.
Canal até ator primário: Página de edição de *websites*.

A.11.4 Agendamento

Previsão: Versão 1.0.

A.12 Solicita relatório

A.12.1 Informações características

Objetivo no contexto: Solicitar de relatório de avaliação.
Escopo: Módulo servidor.
Nível: Tarefa primária.
Pré-condições: Administrador está autenticado e existe algum *website* associado ao administrador solicitante.
Pós-condição de sucesso: Administrador recebe relatório.
Pós-condição de falha: Administrador é notificado da falha ocorrida.

Ator primário: Administrador.
Gatilho: Administrador acessa página da ferramenta e solicita relatório de *websites*.

A.12.2 Cenário principal de sucesso

1. Administrador seleciona *website*.
2. Administrador solicita relatório.
3. Sistema retorna relatório para administrador seguindo.

A.12.3 Informações relacionadas

Prioridade: Alto.
Meta de performance: Menos de 1 minuto.
Freqüência: Não é possível especificar.
Canal até ator primário: Página Web de solicitação de relatório.

A.12.4 Agendamento

Previsão: Versão 1.1.

A.13 Monta relatório

A.13.1 Informações características

Objetivo no contexto: Montar relatório para o administrador que o solicitou.
Escopo: Módulo servidor.
Nível: Tarefa primária.
Pré-condições: Relatório foi solicitado por administrador autenticado.
Pós-condição de sucesso: Relatório apresentado para o administrador que o solicitou.
Pós-condição de falha: Administrador é notificado de que ocorreu alguma falha.
Ator primário: Módulo servidor.
Gatilho: Administrador solicita relatório de *website* administrado por ele.

A.13.2 Cenário principal de sucesso

1. Módulo servidor classifica os *logs*.

2. Módulo servidor agrupa os fluxos de eventos (i.e., identifica barreiras que os usuários enfrentaram, tendo foco os fluxos relacionados aos usuários que utilizam tecnologias assistivas).

3. Módulo servidor gera os grafos com os fluxos de utilização e destacam as barreiras identificadas na fase anterior.

4. Módulo servidor reúne todas as informações do *website* relacionado à solicitação e apresenta o relatório para o administrador solicitante.

A.13.3 Informações relacionadas

Prioridade: Alta.
Meta de performance: Montar relatórios semanais em menos de 1 minuto.
Freqüência: Não é possível especificar.
Canal até ator primário: Sistema de arquivos ou bancos de dados.

A.13.4 Agendamento

Previsão: Versão 1.1.

A.14 Classifica *logs*

A.14.1 Informações características

Objetivo no contexto: Classificar os *logs* como sendo de usuários que utilizam ou não tecnologias assistivas.
Escopo: Módulo servidor.
Nível: Tarefa primária.
Pré-condições: Logs gravados no módulo servidor.
Pós-condição de sucesso: Logs classificados.
Pós-condição de falha: Logs não foram classificados.
Ator primário: Módulo servidor.
Gatilho: Administrador solicita relatório.

A.14.2 Cenário principal de sucesso

1. Logs são divididos em dois grupos: um contendo maior porcentagem de sessões de usuários que responderam que utilizam tecnologias assistivas, outro com as sessões restantes.

A.14.3 Informações relacionadas

Prioridade: Alta.

Meta de performance: Poucos segundos por sessão gravada.

Freqüência: Não é possível especificar.

Canal até ator primário: Bancos de dados ou sistema de arquivos.

A.14.4 Agendamento

Previsão: Versão 1.1.

A.15 Agrupa *logs*

A.15.1 Informações características

Objetivo no contexto: Agrupar *logs* de acordo com o fluxo de eventos neles contidos.

Escopo: Módulo servidor.

Nível: Tarefa primária.

Pré-condições: Logs estão classificados.

Pós-condição de sucesso: Fluxos de uso são agrupados de acordo com a classificação e de acordo com os dados relacionados aos seus eventos. Ainda, as informações resultantes do agrupamento são gravadas de forma que possibilitem a geração de grafos desses fluxos com a marcação das barreiras enfrentadas pelos usuários de tecnologias assistivas.

Pós-condição de falha: Fluxos de uso não estão agrupados, o que impossibilita a geração dos grafos de utilização com a marcação das barreiras enfrentadas pelos usuários de tecnologias assistivas.

Ator primário: Módulo servidor.

Gatilho: Administrador solicita relatório.

A.15.2 Cenário principal de sucesso

1. Fluxos de eventos são agrupados tendo em vista a identificação de barreiras encontradas por usuários.

2. Descrição do grafo de fluxos é gravada no sistema no formato utilizado pela ferramenta responsável por gerar os grafos (*default* Dot).

A.15.3 Informações relacionadas

Prioridade: Alta.

Meta de performance: Menos de 10 segundos.

Freqüência: 1 vez por semana.

Canal até ator primário: Banco de dados.

A.15.4 Agendamento

Previsão: Versão 1.1.

A.16 Desenha grafo de utilização

A.16.1 Informações características

Objetivo no contexto: Desenhar o grafo de utilização para que especialistas possam analisar as barreiras identificadas.

Escopo: Módulo servidor.

Nível: Sub-função.

Pré-condições: Representação do grafo de fluxos agrupados e classificados está disponível no sistema e programa externo para desenhar o grafo está acessível para o sistema.

Pós-condição de sucesso: Grafo com os fluxos de utilização está disponível para o administrador.

Pós-condição de falha: Grafo com fluxos de utilização não foi gerado.

Ator primário: Módulo servidor.

Gatilho: Administrador solicita relatório.

A.16.2 Cenário principal de sucesso

1. Módulo servidor lê informações resultantes do agrupamento de fluxos de utilização.

2. Módulo servidor utiliza programa externo e gera grafo representando os fluxos de utilização tendo destacadas as barreiras enfrentadas pelos usuários.

A.16.3 Informações relacionadas

Prioridade: Alta.

Meta de performance: Menos de 5 segundos por grafo.

Freqüência: p+1 vezes por relatório, tal que p é o número de páginas existentes em cada *website* avaliado. Assim o sistema retorna um grafo por página, em que os nós são os eventos, e um para o *website*, em que os nós são as páginas.

Canal até ator primário: Bancos de dados ou sistema de arquivos.

Atores secundários: Programa externo para geração de grafos, inicialmente Graphviz.

Canal até o ator secundário: Arquivo de descrição de grafo, inicialmente .Dot.

A.16.4 Agendamento

Previsão: Versão 1.1.

Apêndice B

Diagrama de casos de uso

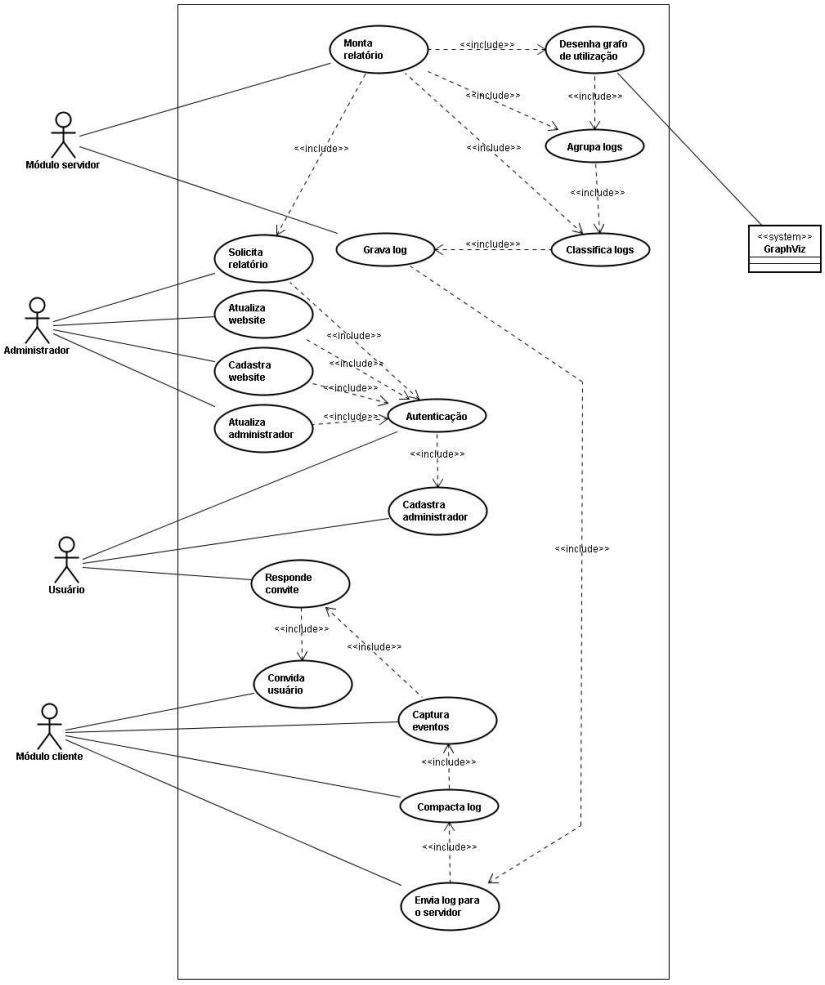


Figura B.1: Diagrama de casos de uso.

Apêndice C

Aprendendo sobre Acessibilidade e Construção de *Websites* para Todos

Artigo publicado na Revista Brasileira de Informática na Educação (RBIE), volume 16, número 3.
Vagner F. de Santana, Leonelo D. A. Almeida e M. Cecília C. Baranauskas

C.1 Introdução

Tecnologias de Informação e Comunicação (TICs) são parte essencial da vida social, econômica e educacional das pessoas na sociedade contemporânea. Assim, garantir que pessoas com algum tipo de limitação tenham acesso à informação é essencial [62]. O número de *websites* existentes na Internet é de cerca de 120 milhões [13]. Destes, mais de 90% falham ao serem confrontados com padrões mínimos de acessibilidade como fornecer descrições adequadas para gráficos ou possibilitar que usuários alterem o tamanho no qual as páginas são apresentadas [61]. Esta realidade não é diferente no contexto de ambientes educacionais na Web.

Websites das universidades têm conquistado um papel crucial no dia-a-dia dos seus estudantes, bem como de estudantes prospectivos. O uso da Web é cada vez mais requerido na vida universitária. Páginas Web das universidades contêm informação importante sobre recursos acadêmicos (catálogos de cursos, calendários, etc.), eventos no campus, estratégias administrativas, acesso a serviços como biblioteca, matrícula, etc. *Websites* que não são acessíveis podem prevenir certos usuários de participar das atividades da vida universitária. Para Kane *et al.* [51], páginas Web inacessíveis podem promover um "educational divide" no qual pessoas com algum tipo de deficiência são alijadas do acesso à educação e conseqüentemente a outros aspectos da vida em sociedade. Esses mesmos autores mostraram que a acessibilidade ainda é um problema para a maioria das 100

melhores universidades do mundo.

O Decreto 5.296/2004 regulamenta leis e estabelece critérios básicos para a promoção de acessibilidade. O artigo 24 deste Decreto aponta que "estabelecimentos de ensino de qualquer nível, etapa ou modalidade, públicos ou privados, proporcionarão condições de acesso e utilização de todos os seus ambientes ou compartimentos para pessoas portadoras de deficiência ou com mobilidade reduzida" [25]. No mesmo decreto está definido que "a concessão de autorização de funcionamento, de abertura ou renovação de curso pelo Poder Público, o estabelecimento de ensino deverá comprovar que [...] coloca à disposição de professores, alunos, servidores e empregados portadores de deficiência ou com mobilidade reduzida ajudas técnicas que permitam o acesso às atividades escolares e administrativas em igualdade de condições com as demais pessoas" [25]. Ainda, cerca de 14,5% da população brasileira possui algum tipo de limitação ou deficiência [47]. Dessa forma, a adequação de *websites* a todos indiscriminadamente envolve a promoção da cidadania, impactando na vida social, econômica e educacional da população, em especial da brasileira.

Até por força da legislação, nos últimos anos acessibilidade tem sido reconhecida como um dos principais requisitos para sistemas na Web baseados em conteúdo. A necessidade de prover os aprendizes com conteúdo baseado na Web que satisfaça suas necessidades e preferências em termos de acessibilidade, assim como o ajuste do conteúdo de aprendizado ao equipamento do usuário tem sido identificado como um problema importante na literatura sobre hipermídia educacional acessível [52]. Vários outros autores discutem o emprego de estratégias de desenvolvimento apropriadas ao tratamento da acessibilidade de forma a conseguir melhor qualidade da interação em tempo de uso de aplicações de software e serviços no contexto de e-learning e e-entertainment [68].

Acessibilidade na Web significa que pessoas com diferentes tipos de limitação podem perceber, entender, navegar, interagir com a Web e contribuir para a mesma [79]. Usabilidade, em suma, é a capacidade de um produto ser utilizado por usuários específicos para atingir objetivos específicos com eficiência e satisfação, dentro de um determinado contexto de uso [48]. Complementarmente, associado à importância do acesso às TIC está o entendimento de como essas tecnologias são utilizadas por todos seus usuários. Isso remete à combinação dos conceitos Acessibilidade e Usabilidade (A&U), uma vez que, em diversos casos, questões relacionadas a uma das disciplinas também contribuem para a outra e vice versa (e.g., teclas de atalho para regiões da tela aumentam a acessibilidade para usuários que usam leitores de tela e também auxiliam usuários a utilizarem uma página Web de maneira mais eficiente possibilitando acesso direto a áreas específicas com apenas um comando). Assim, o apoio à busca de acessibilidade na Web não exclui nenhum dos usuários e estende o conceito de usabilidade como um todo.

Designers, desenvolvedores e pessoas em geral que desejam criar *websites* acessíveis

contam com uma ampla variedade de recursos online, documentação técnica e ferramentas de software. Apesar da disponibilidade desses recursos, muitos continuam a produzir páginas inacessíveis e a tendência é que se tornem menos acessíveis ao longo do tempo [51].

Para melhorar A&U de *websites* existem diretrizes e heurísticas. No entanto, materiais que auxiliam na aplicação e avaliação das mesmas são fragmentados e tratam tecnologias (e.g., HTML, Javascript e CSS) e conceitos (e.g., A&U) de maneira pouco articulada. De forma a suprir essa lacuna, o presente trabalho apresenta um Processo para Adequação de *Websites* a Requisitos de A&U (PAWRAU) e sua implementação em um *website*, o WARAU. Para evidenciar a contribuição deste trabalho, é feita uma comparação preliminar entre a articulação do conteúdo utilizado no WARAU e em tutoriais e referências disponibilizados na Web para o mesmo fim.

O artigo está organizado da seguinte forma: a seção C.2 apresenta uma visão geral de recursos oferecidos para a construção e avaliação de acessibilidade na Web e discute suas principais limitações; a seção C.3 apresenta o processo PAWRAU, desenvolvido com a participação de webdesigners; a seção C.4 apresenta o WARAU, um ambiente na Web para apoio e colaboração no desenvolvimento acessível de *websites*, mostrando seu diferencial em relação a tutoriais online; a seção C.5 apresenta as conclusões e os trabalhos futuros.

C.2 Uma visão geral de recursos para apoio à construção Web-acessível

Nesta seção apresentamos alguns dos principais recursos disponíveis atualmente para a construção de *websites* acessíveis e usáveis. Em adição, apontamos algumas das principais lacunas de tais recursos conforme literatura da área. Os recursos foram divididos em duas categorias: materiais conceituais (e.g., tutoriais, diretrizes, hipertextos) e ferramentas (e.g., validadores, simuladores, ferramentas de autoria).

Entre os materiais conceituais é possível encontrar um grande número de tutoriais em formato de textos ou hipertextos (e.g., W3Schools [76], WebAIM [46]) e algumas iniciativas que visam o estabelecimento de princípios e diretrizes. O W3Schools é um dos portais de tutoriais de tecnologias Web mais utilizados por mantenedores de *websites*. Ele aborda diversas tecnologias Web, tanto livres (e.g., HTML, CSS) como proprietárias (e.g., Microsoft.Net). Em tecnologias como HTML e CSS, os tutoriais cobrem de maneira completa as definições das linguagens (definidas pelo W3C), além de oferecer recursos para que usuários possam interagir com os exemplos oferecidos nos tópicos desses tutoriais. Entretanto, essa abordagem, apesar de amplamente empregada em tutoriais online, possui lacunas no que diz respeito à integração das tecnologias Web (e.g., recomendações de uso de folhas de estilo CSS para separar o conteúdo da apresentação) e também estão restritas

a especificações de linguagens não endereçando questões relacionadas a técnicas que podem aumentar a qualidade de *websites* (e.g., uso de breadcrumbs para melhorar a navegação).

Dentre as iniciativas destacam-se o Web Acessibility Initiative (WAI) [79], o Section 508 [2], o Stanca Act [64] e, no contexto brasileiro, o e-MAG [33]. A iniciativa mais amplamente adotada é o WAI, promovido pelo World Wide Web Consortium (W3C), que visa o desenvolvimento de diretrizes e recursos que contribuem para tornar a Web acessível. O WAI concentra seus esforços em três focos, tal como descrito em [16]:

- **Navegadores Web**, players multimídia e tecnologias assistivas que permitem uma experiência completamente usável e acessível. Oferece o conjunto de diretrizes User Agent Accessibility Guidelines 1.0 (UAAG 1.0) [84] e a versão 2.0, ainda em estágio de desenvolvimento (UAAG 2.0) [85];
- **Ferramentas de Autoria de conteúdos Web** e ambientes de desenvolvimento que produzem conteúdo Web acessível e têm interfaces acessíveis. Oferece o conjunto de diretrizes Authoring Tool Accessibility Guidelines 1.0 (ATAG 1.0) [83];
- **Conteúdo Web concebido para ser acessível**. Oferece o conjunto de diretrizes Web Content Accessibility Guidelines 1.0 (WCAG 1.0) [82] e a versão 2.0, ainda em estágio de desenvolvimento (WCAG 2.0) [86].

Apesar de representar uma base sólida de princípios amplamente discutidos por uma comunidade bastante diversificada, existem diversas críticas sobre esses tipos de iniciativas. Sloan *et al.* [71] apontam algumas delas:

- Natureza teórica das diretrizes, ignorando o uso de tecnologias proprietárias; Dependência de outras diretrizes, o WCAG depende do uso de navegadores que atendam o UAAG; Ambigüidades na interpretação das diretrizes;
- Nível de compreensão necessário dos problemas de acessibilidade oriundo da dificuldade em entender o princípio que norteia um checkpoint.

Com a finalidade de oferecer o referencial necessário para o desenvolvimento de *websites* governamentais, o Governo brasileiro oferece o Padrão Brasil e-GOV [32], que conta com modelos, diretrizes e ferramentas. No contexto de acessibilidade, este projeto possui o Modelo de Acessibilidade de Governo Eletrônico (e-MAG). Esse modelo consiste no oferecimento de informações a desenvolvedores de *websites* governamentais para tornar seus *websites* acessíveis para a maior diversidade de usuários, em consonância com o decreto 5.296 de 2004.

Para a segunda categoria (ferramentas) destacamos a existência de ferramentas de autoria de código Web, ferramentas assistivas, navegadores, validadores e simuladores. Em

relação às ferramentas de autoria de código Web, o grupo responsável pelo ATAG disponibiliza os resultados de avaliação realizada com algumas das ferramentas mais utilizadas:

- Bluefish versão 0.6 - atendeu quase todos os checkpoints de prioridade 1, mas poucos das outras prioridades;
- DreamWeaver versão 4.0, da Macromedia - atendeu metade dos checkpoints de prioridade 1 e poucos de outras prioridades;
- FrontPage 2000, da Microsoft - atendeu poucos checkpoints em todas as prioridades.

Watanabe e Umegaki [77] realizaram pesquisa em que avaliaram algumas ferramentas assistivas segundo as diretrizes do UAAG. Nesse estudo foram avaliadas as ferramentas: PC-Talker XP versão 3.041, 95 Reader versão 6.02, IBM Home Page Reader 3.04 SP3 (versão Japonesa) e JAWS for Windows Professional versão 6.2 (versão Japonesa). Como resultados, Watanabe e Umegaki apontaram que as ferramentas tiveram bons resultados em relação a checkpoints de prioridade 1, no entanto falharam nos de prioridade 2, principalmente nos relacionados à manipulação de multimídia.

Ferramentas de validação e simulação são um importante recurso na criação de código acessível, seja pela facilidade em realizar uma varredura no código ou pela dificuldade que pequenas equipes encontram em ter contato com toda a diversidade de usuários existentes em cenários como o brasileiro. Algumas das ferramentas de validação amplamente utilizadas são o ATRC [15], o MAGENTA [54] e, no contexto brasileiro, o DaSilva [12] e o ASES [31], sendo esta última uma ferramenta que funciona localmente na máquina do usuário.

O ASES é uma ferramenta produzida no contexto do e-MAG que, além de oferecer mecanismos de validação segundo as diretrizes do e-MAG, ainda possui ferramentas de simulação do uso do conteúdo Web por pessoas com baixa visão (e.g., miopia, daltonismo) e leitor de tela. DaSilva é uma ferramenta online que permite a validação de *websites* segundo as diretrizes do WCAG e também do e-MAG.

Outras ferramentas simuladoras interessantes são o Color Laboratory [4] e o Colorblind Web Page Filter. Color Laboratory [80] é um simulador de paleta de cores que permite ao usuário ajustar a visualização segundo seu sistema operacional, monitor e deficiência visual. Já o Colorblind Web Page Filter permite a visualização de uma dada URL informada pelo usuário, considerando o tipo de deficiência visual.

Na educação temos diversos trabalhos que visam a utilização de hipertextos e ferramentas que apóiam a colaboração entre pessoas [9, 27]. No entanto, o papel da tecnologia como fator de integração entre pessoas, quando não consciente dos cuidados a se ter com relação à acessibilidade, pode ser distorcido, levando em certos casos, à separação entre pessoas sem ou com deficiência ou a exclusão desse segundo grupo.

Portanto, podemos verificar que há uma diversidade de materiais conceituais e ferramentas para apoiar os mantenedores de *websites*, mesmo que ainda não existam padrões de acessibilidade. No entanto, o conhecimento da existência de tais ferramentas, o entendimento das diretrizes e a articulação das diversas tecnologias disponíveis para a criação de conteúdo Web é, ainda, uma barreira a ser transposta. Nas próximas seções apresentamos nossa proposta para contribuir nesse cenário.

C.3 Um Processo para Adequação de *Websites* a Requisitos de Acessibilidade e Usabilidade (PAWRAU)

Como apresentado na seção anterior, muitas são as fontes de orientação para o desenvolvimento de código Web e existem diversas iniciativas que visam o estabelecimento de diretrizes e recomendações para a avaliação de A&U. No entanto, observam-se lacunas no que diz respeito à forma como esse conteúdo é apresentado aos mantenedores de *websites*. Sloan *et al.* [71] apontam por exemplo, como deficiências encontradas nas diretrizes do WAI: a subjetividade e ambigüidade na interpretação dos checkpoints, a complexidade em entender diferentes níveis de prioridades das diretrizes e a compreensão dos princípios de acessibilidade subjacentes aos checkpoints. Como resultado dessas deficiências encontramos um cenário em que a maioria dos mantenedores de conteúdo ignoram questões de A&U ou consideram tais questões secundárias e com custo demasiado em relação aos benefícios trazidos por elas.

Com o intuito de contribuir para mudar esse cenário, este trabalho propõe um processo que visa a integração das linguagens mais utilizadas para criação e manutenção de conteúdo Web (i.e., (X)HTML, CSS e Javascript) com conceitos, técnicas e recomendações referentes à acessibilidade e à usabilidade. Nesta seção apresentamos os princípios que nortearam a criação do conteúdo do PAWRAU, em seguida a dinâmica participativa empregada para a construção desse conteúdo e, por último, o processo e alguns exemplos de utilização.

A Figura C.1 apresenta a cronologia em que foram executadas as atividades de criação e divulgação do PAWRAU. Nos primeiros oito meses de atividades realizamos dinâmicas participativas com pessoas encarregadas da criação e manutenção de *websites*, em formato de oficinas, nas quais validamos e elicitamos novos requisitos e conteúdos que, posteriormente, compuseram a documentação do PAWRAU. A partir de fevereiro deste ano iniciamos a materialização do PAWRAU e sua disponibilização na Web (disponível em `http://warau.nied.unicamp.br`). O *website* vem sendo alimentado por meio de atividades de avaliação em outros projetos, de colaboração de usuários do *website* e de

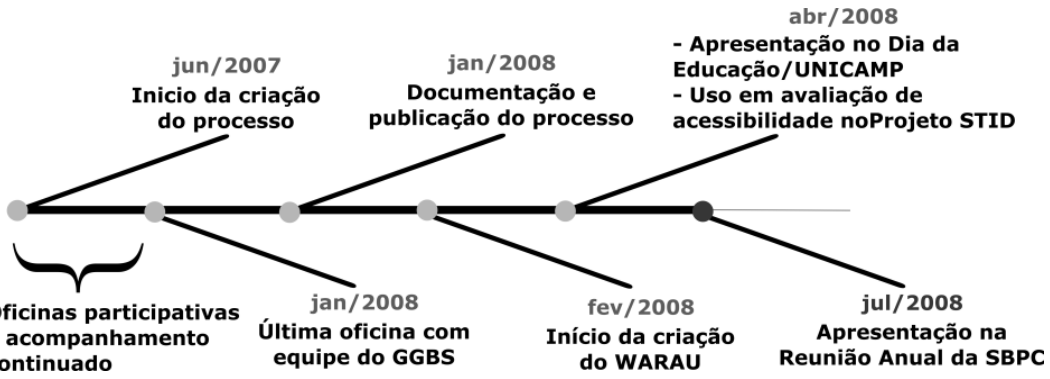


Figura C.1: Cronologia das atividades para o desenvolvimento do processo.

apresentações em eventos da área.

A elaboração do conteúdo do PAWRAU foi realizada de forma participativa e incremental, tendo como unidade as oficinas realizadas com a equipe de desenvolvimento de um *website* institucional da UNICAMP. Adicionalmente, todo o conteúdo foi norteado por princípios oriundos de experiências (profissionais e acadêmicas) e que acreditamos serem necessários para a criação e manutenção de *websites* acessíveis e usáveis. São eles:

1. **Definir padrão de codificação** - A seleção criteriosa de padrão de nomenclatura de elementos, variáveis e versões de linguagem contribui para a legibilidade do código, e para a divisão das atividades da equipe de desenvolvimento e, com isso, facilita a manutenção das funcionalidades de *websites* bem como a adição de novos recursos;

2. **Estruturar páginas e *websites* prezando o reaproveitamento de código** - Todo o código Web, seja ele documento HTML, folha de estilo CSS ou programa em Javascript, deveria ser escrito de tal forma que pudesse ser reaproveitado em diferentes áreas do *website* e, conseqüentemente contribuir para a manutenção e redução de consumo de recursos de tempo e financeiros;

3. **Prezar pela semântica no código** - O conteúdo de *websites* deve ser escrito considerando os elementos semânticos disponíveis pelas linguagens de marcação (e.g., títulos, parágrafos, tabelas, abreviações);

4. **Aplicar padrões e diretrizes de tecnologias e conceitos** - Linguagens Web contam com padrões e recomendações que, quando conhecidos por mantenedores, solucionam grande parte dos problemas comumente encontrados em *websites*. Com isso, possibilitam sua maior compatibilidade com os diversos dispositivos, navegadores, sistemas operacionais e outras aplicações utilizadas. Além disso, quando consideradas diretrizes de acessibilidade e usabilidade, espera-se um ganho em relação

à capacidade do *website* atender às necessidades específicas de cada usuário (e.g., navegação via teclado, sem recursos sonoros);

5. **Não se restringir a padrões e diretrizes de tecnologias e conceitos** - Apesar de trazer melhorias, padrões e diretrizes não são soluções suficientes para a garantia de qualidade de um *website*. Para tanto é necessário considerar as condições e restrições de uso específicas de cada *website* e levar em consideração o referencial teórico da área em questão;

6. **Considerar a diversidade de usuários** - Ao contrário do que geralmente é adotado por mantenedores de *websites*, desenvolver *websites* para o "usuário médio" não é garantia de ampla aceitação de *websites*. Portanto, o conhecimento da diversidade de usuários pode ser fator determinante para o sucesso de um *website*. Tal conhecimento complementa e, por vezes, redireciona diretrizes e padrões;

7. **Considerar diferentes formas de apresentação de páginas Web (dispositivos e configurações)** - *Websites* não são documentos estáticos e, portanto, seriam melhor construídos se fossem considerados como construções flexíveis a diferentes dispositivos, tamanho de display e preferências de visualização de usuários;

8. **Integrar tecnologias e conceitos durante todo o desenvolvimento** - Um dos grandes problemas no desenvolvimento de *websites* é a lacuna entre as recomendações técnicas e os conceitos que as norteiam. Um exemplo disso é a recomendação de acessibilidade sobre o fornecimento de texto alternativo a imagens. Apesar de prover texto alternativo e, portanto seguir a diretriz (e.g., checkpoint 1.1 do WCAG 1.0), mantenedores falham na escolha de qual informação deveria estar presente nesse texto e quais são os usuários que se beneficiam desse recurso. Para tanto é necessária uma abordagem integrada que permita a compreensão não somente das regras de desenvolvimento mas também das necessidades e dos benefícios gerados por sua aplicação;

9. Avaliação e validação - Devido à característica dinâmica de *websites*, mesmo quando mantenedores conhecem e empregam os padrões e recomendações, a tarefa de manter um *website* atendendo completamente a essas recomendações exige um monitoramento constante. Esse monitoramento pode ser obtido por meio de ferramentas automatizadas de validação de código ou por meio de avaliação manual.

O WAI conta com 10 recomendações breves que apresentam conceitos importantes de webdesign acessível e que servem como porta de entrada para as diretrizes WCAG 1.0 [82]. A Tabela C.1 apresenta a relação das 10 recomendações do WAI com os princípios levados em consideração no desenvolvimento do conteúdo do PAWRAU. Nela pode-se verificar que

Recomendações do WAI	**Princípios do PAWRAU relacionados**
1 - **Imagens & animações**: Use o atributo alt para descrever a função visual.	3, 4, 6, 7
2 - **Mapas de imagem**. Use mapeamento client-side e texto para regiões clicáveis.	4, 6, 7
3 - **Multimídia**. Forneça legenda e transcrições de áudio e descrições para vídeo.	3, 4, 6, 7
4 - **Links de hipertexto**. Use texto que faça sentido quando lido fora de contexto. Por exemplo, evite "clique aqui".	3, 4, 6, 7
5 - **Organização da página**. Use títulos, listas e estrutura consistente. Use CSS para layout e estilo onde for possível.	3, 4, 6, 7, 8
6 - **Gráficos & diagramas**. Resuma ou use o atributo longdesc.	3, 4, 6, 7
7 - **Scripts, applets & plug-ins**. Forneça conteúdo alternativo no caso em que funcionalidades ativas estiverem inacessíveis ou forem incompatíveis.	4, 6, 7, 8
8 - **Frames**. Use o elemento noframes e titles significativos.	3, 4, 6, 7
9 - **Tabelas**. Faça com que a leitura linha-a-linha seja sensata. Resuma.	3, 4, 6, 7
10 - **Verifique seu trabalho**. Valide. Use ferramentas, checklists, e diretrizes em `http://www.w3.org/TR/WCAG`	4, 6, 7, 8, 9

Tabela C.1: Relação entre recomendações do WAI [78] e princípios do PAWRAU.

os princípios utilizados no PAWRAU e as recomendações do WAI são consoantes. Ainda, é possível verificar que os princípios 1 e 2 que propusemos são mais abrangentes, pois consideram a construção e manutenção de *websites* e endereçam questões importantes do dia-a-dia de uma equipe de desenvolvimento. Complementarmente, consideramos que outro ponto importante representado no princípio 5 é o de não nos restringirmos às diretrizes e padrões, dado o dinamismo de tecnologias e técnicas utilizadas na Web como o desenvolvimento de novas formas de desenvolvimento de *websites*, novos dispositivos de interação, novas tecnologias assistivas, etc.

Em nossa proposta, além dos princípios apresentados anteriormente, foram levadas em consideração as seguintes premissas:

O perfil de mantenedores de *websites* não é bem definido e em equipes pequenas um profissional pode desempenhar mais de um papel (e.g., redator e designer ou desenvolvedor e designer). Além disso, o redator pode ser alguém com conhecimento no domínio sendo representado e pouca sofisticação técnica, o que é muito comum no contexto de ambientes educacionais, por exemplo. Assim, referências para mantenedores de *websites* devem

possibilitar o acesso às informações mais relevantes para uma certa combinação de perfis em um dado momento;

A rotina de trabalho de mantenedores de *websites* não é linear, ou seja, oscila entre atividades de design, codificação, padronização, avaliação, testes, entre outras. Desta forma foi identificada a necessidade de estruturar o conteúdo em tópicos auto-contidos, que explicitam quais são os resultados esperados de sua aplicação. Além disso, deve-se utilizar hipertexto para possibilitar a navegação para tópicos mais aprofundados (e.g., relacionados à avaliação ou testes) e outros mais fundamentais (e.g., relacionados a definições ou estruturas básicas).

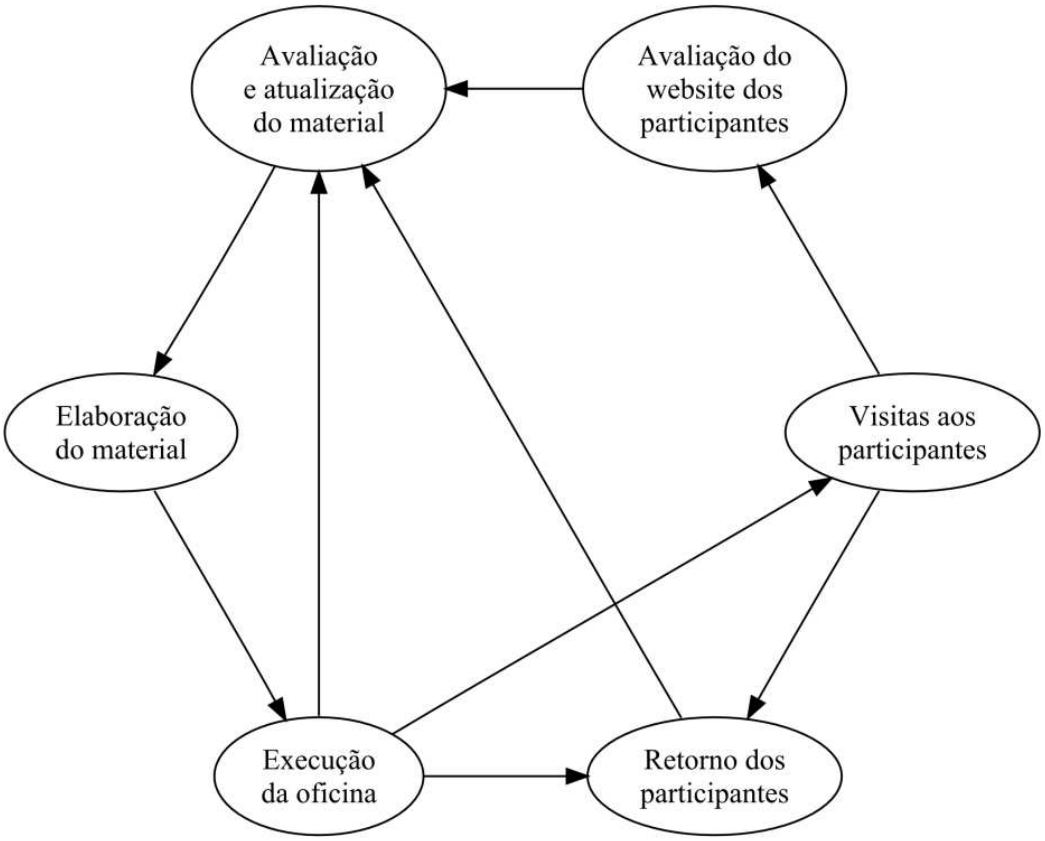


Figura C.2: Modelo participativo de criação do PAWRAU.

A Figura C.2 ilustra o modelo empregado na construção do material utilizado no PAWRAU. A dinâmica utilizada foi a seguinte:

1. Para cada oficina foi realizado um levantamento de materiais disponíveis na literatura tais como, padrões, diretrizes, boas práticas e artigos, além da inclusão de materiais resultantes da experiência profissional dos autores;

2. A partir desse material cada oficina foi executada juntamente com a equipe de desenvolvimento que participou do desenvolvimento do PAWRAU;

3. Durante as oficinas o material elaborado começava a ser avaliado pelos próprios autores. Assim, exemplos e explicações que não haviam sido previstas na elaboração e que foram apresentadas durante a oficina puderam ser incorporados na referência disponibilizada aos participantes. Neste ponto, outra entrada no processo de avaliação era o retorno dado pelos participantes na forma de comentários, dúvidas, explicações sobre a forma como trabalhavam, entre outros;

4. Ao final de cada oficina os participantes ficavam encarregados de aplicar o conteúdo da oficina nos *websites* sobre os quais trabalhavam formalmente;

5. Após uma ou duas semanas era marcada, de comum acordo, uma visita ao local de trabalho dos participantes para que fossem explicitadas as dificuldades e progressos em relação à aplicação do conteúdo trabalhado na oficina. A partir disto foi possível reforçar alguns conceitos e orientar o prosseguimento do trabalho;

6. Após a visita os participantes informavam a conclusão da aplicação do conteúdo da oficina no *website* que mantinham. Então, era possível verificar as alterações feitas no *website* dos participantes e, conseqüentemente, se os conceitos haviam sido aplicados de maneira adequada;

7. Por fim, a elaboração do material para a próxima oficina se iniciava, dando início a outra iteração.

A partir dos princípios e premissas que permearam a elaboração e execução das oficinas buscou-se organizar o material de forma que o PAWRAU pudesse ser utilizado por outras equipes mantenedoras de *websites*, como material de apoio na Web. Assim, os tópicos tratados em cada oficina foram agrupados para possibilitar que usuários com diferentes habilidades pudessem navegar livremente pelo conteúdo, pulando o que já conhecem para ir direto para a aplicação de técnicas e avaliação, ou revisando conceitos antes de realizar uma alteração e avaliação. Os conjuntos utilizados para organizar o material são baseados em categorias que refletem a semântica dos tópicos do PAWRAU. A Figura C.3 apresenta como esses conjuntos agrupam tópicos de organização (e.g., definição, padronização), implementação (e.g., codificação, técnica) e análise do conteúdo (e.g., avaliação).

O processo pode ser utilizado a partir de qualquer dos conjuntos de categorias apresentados na Figura C.3. Cenários mais lineares podem ser vistos quando mantenedores buscam alterar seus *websites* e seguem passos de padronização e definição, depois aplicam certas técnicas e, por fim, avaliam as modificações feitas. Neste exemplo a linearidade impede que termos utilizados não tenham sido definidos em etapas anteriores. Em outro

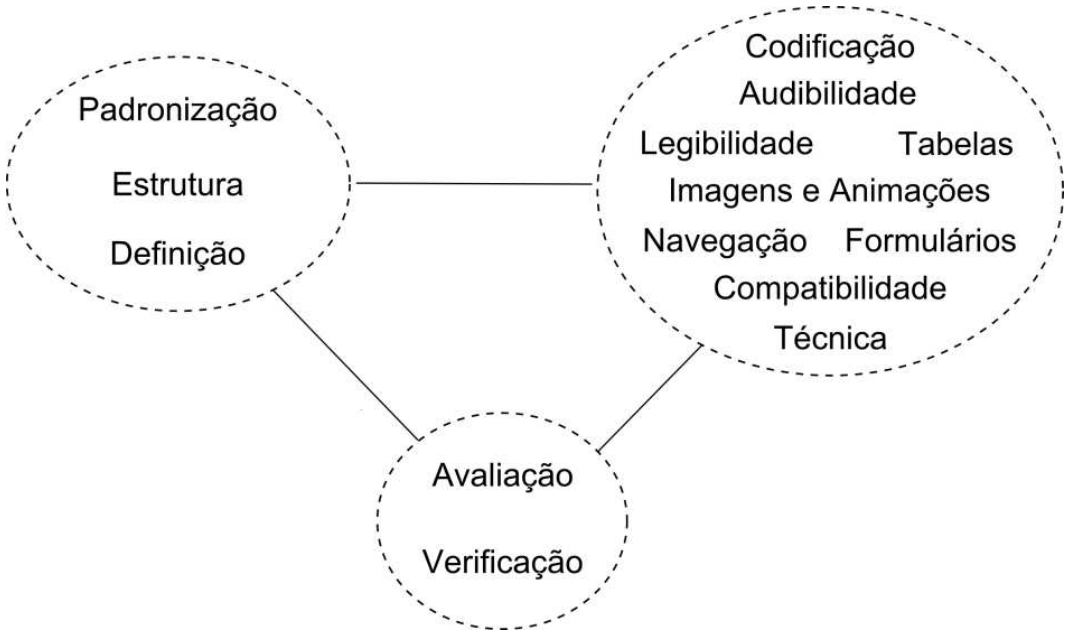


Figura C.3: Relação entre as categorias do material utilizado no PAWRAU.

cenário a utilização do PAWRAU pode ter início na avaliação de um *website*. Por exemplo, um mantenedor poderia receber uma mensagem de uma ferramenta semi-automática informando que textos alternativos em imagens são obrigatórios e que este recurso é essencial para usuário de ferramentas assistivas. Com isso, o mantenedor poderia verificar a definição de ferramentas assistivas para entender melhor a mensagem da ferramenta semi-automática e só então verificar qual técnica deveria utilizar para eliminar o problema retornado pela ferramenta semi-automática. Apesar do PAWRAU ter sido estruturado tendo em vista a replicação para outras equipes de desenvolvimento e, apesar de ter sido aplicado em um estudo de caso no qual obtivemos resultados significativos [5], verificou-se que a forma de apresentação do material desenvolvido não favorecia a flexibilidade idealizada no desenvolvimento do PAWRAU. Então, surgiu a necessidade de materializar o PAWRAU de forma a possibilitar filtragem do conteúdo de acordo combinações de temas (e.g., HTML, CSS, Javascript, Usabilidade e Acessibilidade) e diferentes perfis (i.e., desenvolvedor, designer e redator), o que levou à criação do WARAU, que será detalhado na próxima seção.

C.4 WARAU: Uma materialização do processo

Nesta seção apresentamos o WARAU (acrônimo de *Websites* Atendendo a Requisitos de Acessibilidade e Usabilidade) que é uma materialização do PAWRAU (Figura C.4). Um

dos objetivos do WARAU é a ampla divulgação do processo e a criação de um espaço para discussão do conteúdo do PAWRAU. Além disso, o WARAU permite que usuários façam buscas rápidas ao conteúdo do processo, buscas estas que consideram o contexto de trabalho e o foco de interesse de cada perfil de mantenedor de *websites*.

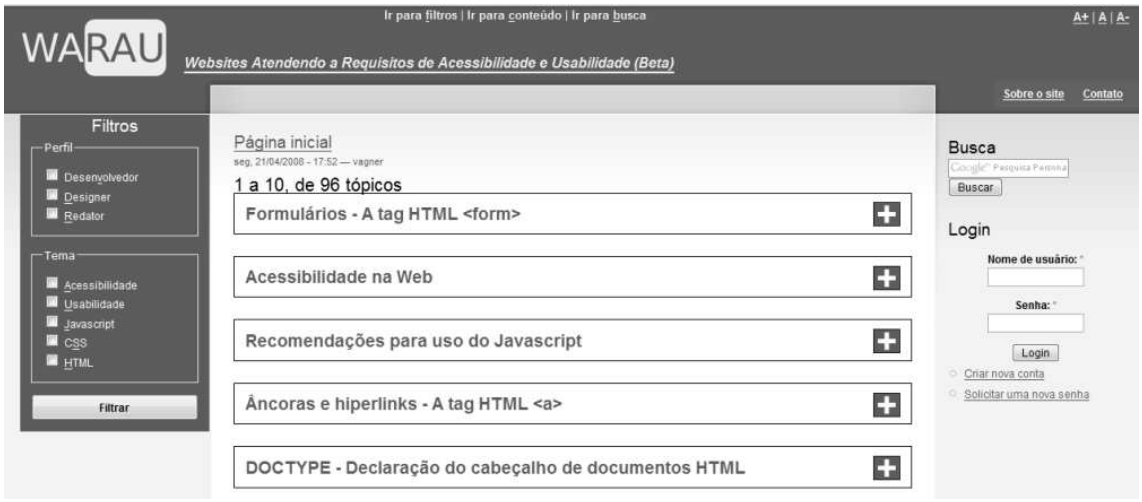


Figura C.4: Página inicial do *website* WARAU.

Para estruturar o conteúdo do PAWRAU de forma a oferecer suporte à extensão e flexibilidade na manipulação do conteúdo usando o WARAU utilizou-se uma estrutura em forma de tópicos descritos usando XML (eXtensible Markup Language) como linguagem de marcação. Esta estrutura de tópicos (ilustrada na Figura C.5) permite atualmente a representação estruturada de: a) o assunto que está sendo discutido no tópico; b) exemplos e contra-exemplos de código, incluindo renderizações; c) sugestões de ferramentas que podem ser utilizadas para alcançar o objetivo do tópico; d) sugestões de leitura prévia para a devida compreensão do texto (i.e., outros tópicos do WARAU); e) os resultados esperados do tópico.

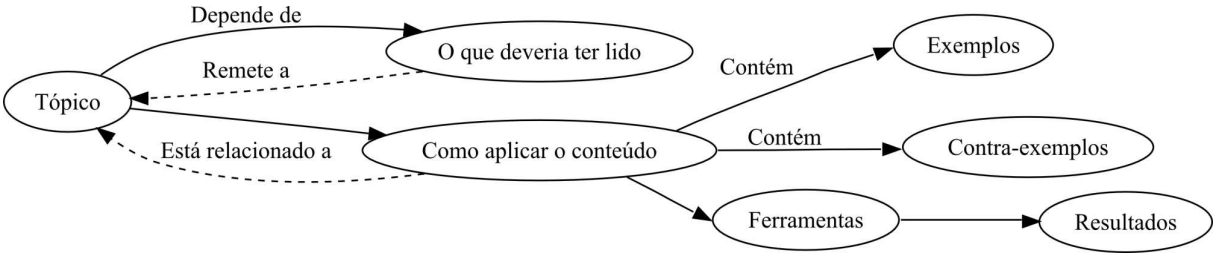


Figura C.5: Estrutura dos tópicos do WARAU.

O diagrama apresentado na Figura C.6 representa a DTD (Document Type Definition) do WARAU. Nessa DTD alguns metadados compõem os tópicos:

- **Perfis** - representa a atribuição de relevância do tópico para cada um dos perfis de mantenedores (i.e., designer, desenvolvedor e redator). Atualmente, a relevância pode assumir um de três valores: alta, média ou baixa;
- **Temas** - atualmente estão disponíveis HTML, CSS, Javascript, Acessibilidade e Usabilidade. Como um dos princípios do PAWRAU é o entendimento integrado das diversas tecnologias e disciplinas envolvidas no desenvolvimento de *websites*, cada tópico pode estar relacionado a vários temas;
- **Ferramentas** - estão presentes *websites* de avaliação, ferramentas de autoria, formulários de avaliação, entre outros;
- **Resultados** - atualmente estão disponíveis 14 categorias de resultados que abrangem desde padrões de codificação e estruturas de documentos até verificação e avaliação. Cada tópico pode endereçar mais de uma categoria de resultado;
- **Referências** - conjunto de artigos, *websites* e outras fontes utilizadas na construção do conteúdo do WARAU.

Além de metadados, a representação dos tópicos segundo essa marcação XML permite a descrição semântica do conteúdo segundo a estrutura apresentada na Figura C.5 e traduzida em 5 tipos de elementos: blocos de texto e elementos de texto (links, ênfase das palavras, citações a referências cadastradas no XML), lista de itens, tabelas, exemplos e contra-exemplos. Essa descrição usa marcações de conteúdo que podem ser posteriormente interpretadas em diversas linguagens, *websites* ou outros tipos de aplicações.

C.4.1 Diferenciais técnico-metodológicos do WARAU

Mais do que disponibilizar conteúdo em ordem alfabética ou por hierarquia de complexidade de entendimento o conteúdo do WARAU está organizado de forma a prover diversas trilhas de aprendizado conforme a necessidade de cada usuário. Para tanto usamos alguns recursos que permitem que a ordenação do conteúdo possa se ajustar a novos conteúdos de forma natural e semi-automatizada. O primeiro deles é a relação direta de dependência entre tópicos, usando o atributo “depends”. Quando um tópico possui um atributo “depends” para outro, esse último tópico é adicionado à lista de leitura prévia indicada e ganha maior peso no processo de ordenação. Dessa forma, quanto maior for o número de tópicos que dependem de um outro, este terá sua prioridade aumentada na ordenação.

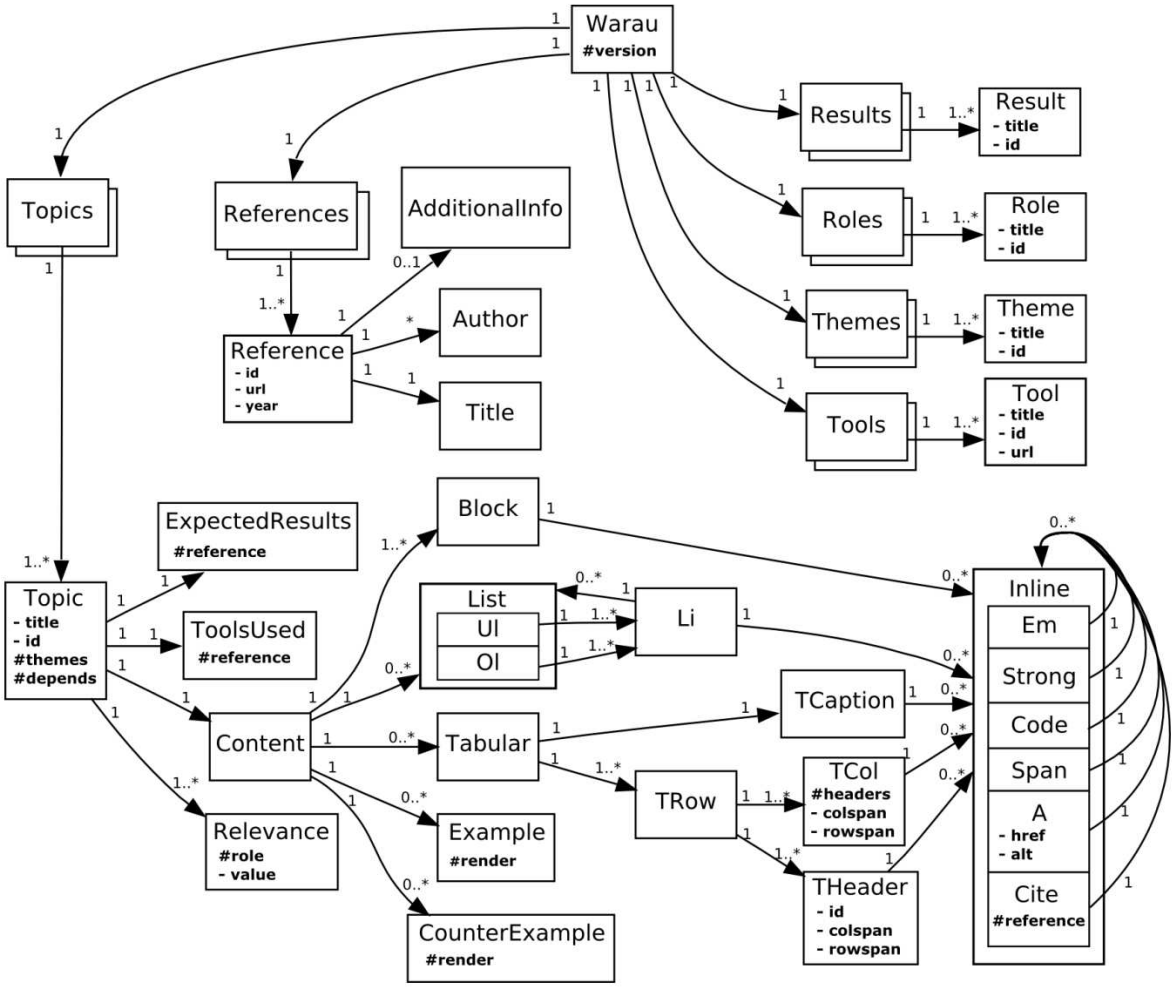


Figura C.6: Representação da DTD utilizada no WARAU.

Outros recursos que influenciam na ordenação dos tópicos são os links para outros tópicos que podem estar no elemento "block" e a relevância do tópico para o perfil do mantenedor.

Em resumo, podemos apontar como alguns dos benefícios obtidos com a utilização da marcação proposta:

- capacidade de extensão do conteúdo com baixo esforço e sem comprometer o conteúdo existente;
- adaptação natural da ordenação dos tópicos de acordo com a semântica do conteúdo;
- utilização da mesma marcação em diversos ambientes de sistemas e linguagens;
- extensão a novas tecnologias e disciplinas;

– suporte ao refinamento de papéis dos usuários;

– atualização de novas ferramentas de apoio à construção de código Web.

A partir da interpretação da estrutura apresentada, o WARAU oferece serviços de filtragem de conteúdo por temas e perfis de usuário e, também, dispõe de um formato acessível e usável de visualização de conteúdo. Ambos os filtros de perfil e tema permitem múltipla seleção, que é uma opção que mostrou-se necessária durante a execução das oficinas para a construção do PAWRAU. Nelas observamos que um cenário comum no contexto institucional da UNICAMP e, também, em parte dos *websites* disponíveis atualmente, é o de pequenas equipes (2 ou 3 pessoas) serem responsáveis e suficientes para a manutenção de *websites*. Nessas equipes, os papéis dos usuários se acumulam e se ajustam constantemente às demandas de trabalho.

Para ilustrar os benefícios obtidos com a integração dos temas envolvidos na manutenção de *websites*, utilizamos um cenário comum de utilização do WARAU: a procura por uma ferramenta de avaliação de acessibilidade, considerando um usuário que não possui conhecimento prévio do assunto (ver Figura C.7). Como ele tem o objetivo específico de avaliar a acessibilidade, geralmente buscando uma maneira rápida de alcançá-lo, o mantenedor filtra o tema "Acessibilidade" e acessa o tópico "Avaliação Simplificada de Acessibilidade" (ASA); lá ele encontra referências sobre o formulário para ASA, descobre que existe uma outra avaliação chamada de "Inspeção Heurística de Usabilidade" e compreende a forma de aplicação do ASA. Para melhor entender o que realmente se entende por acessibilidade em *websites*, o usuário acessa os tópicos "Acessibilidade" e "Acessibilidade na Web". Após entrar no tópico "Inspeção Heurística de Usabilidade" ele verifica que usabilidade também é importante no contexto de conteúdo Web, então ele acessa os tópicos "Usabilidade" e "Usabilidade na Web" para entender melhor esse conceito. Tanto em "Usabilidade" quanto em "Acessibilidade" ele descobre que existe uma intersecção entre esses conceitos e, então ele vai ao tópico "Integração Acessibilidade e Usabilidade". Assim o usuário pode construir o conhecimento fundamental para poder realizar e compreender uma avaliação de acessibilidade. Dessa forma, ele inicia a avaliação usando o formulário de ASA e verifica que algumas imagens do website que está sendo avaliado não possuem equivalente textual. Para saber como resolver esse problema, o usuário encontra o tópico "Imagens - A tag HTML <img>" onde ele aprende como, quando e o quê fornecer como equivalente textual a imagens. Além disso, neste tópico existe uma referência para um tópico sobre o uso de CSS para adicionar efeitos sobre textos, visando evitar o uso desnecessário de imagens.

No exemplo ilustrado pela Figura C.7 pudemos observar como o WARAU integra os diversos agrupamentos de tópicos (anteriormente apresentados na Figura C.3) e como o usuário teria navegado do tema Acessibilidade para outros (i.e., Usabilidade, HTML

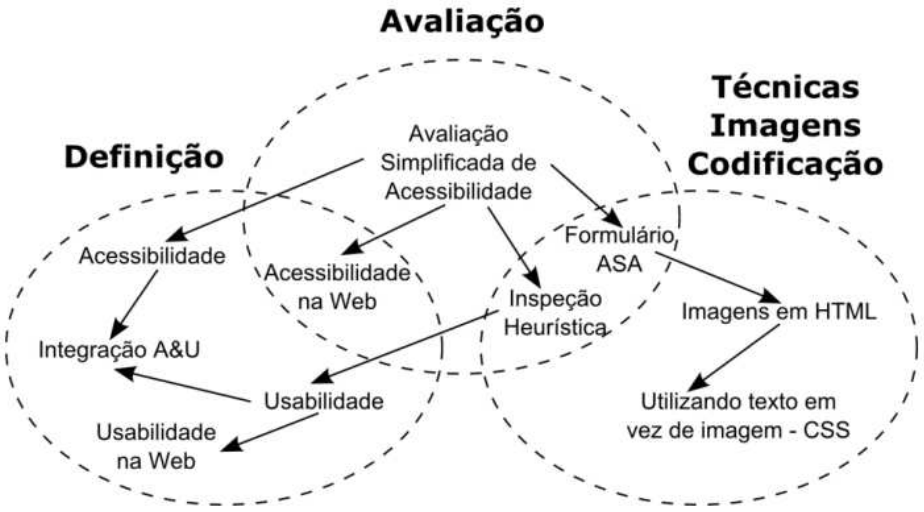


Figura C.7: Mapa conceitual de cenário possível de utilização do WARAU relacionado à Avaliação Simplificada de Acessibilidade.

e CSS) sem que para isso perdesse o foco inicial, que era executar uma avaliação de acessibilidade. Além disso, é possível observar que alguns tópicos contêm informações relacionadas a mais de uma categoria (e.g., de "Acessibilidade na Web" em "Definição"e "Avaliação"), o que torna mais natural a transição entre tópicos de grupos distintos.

Na próxima seção apresentamos uma comparação sintética entre a organização do conteúdo no WARAU e outros materiais disponíveis na Web, visando clarificar a contribuição oferecida por esta proposta.

C.4.2 Comparação com outras ferramentas de apoio e discussão

O conteúdo disponível no WARAU tem como principal diferencial a integração entre diferentes linguagens e conceitos. Esse tipo de integração possibilita a construção de elementos mais complexos que exigem a combinação de diferentes tecnologias e conceitos. A Figura C.8 apresenta a relação do conteúdo dos tutoriais de HTML, CSS e Javascript do W3Schools [76]. Nela é possível verificar que o material é bastante extenso, no entanto, suas páginas buscam detalhar, de maneira isolada, os elementos da tecnologia que tratam. A Figura C.8(A) mostra o relacionamento que encontramos entre páginas de tutoriais de assuntos distintos do W3Schools, envolvendo apenas páginas de índice que são referenciadas sem contexto, o que dificulta o uso de diferentes tecnologias em conjunto. Ainda, o relacionamento entre páginas do tutorial de uma mesma tecnologia traz pouca informação de contexto, uma vez que o relacionamento entre as páginas se dá através de

navegação do tipo "anterior" e "próximo" (Figura C.8, destaque B) ou através de páginas de índice (Figura C.8, destaque C). Por fim, é possível identificar claramente o isolamento dos conjuntos de tópicos de diferentes linguagens, além do relacionamento desses temas ocorrer somente em alguns locais.

A Figura C.9 apresenta o relacionamento dos tópicos das linguagens HTML, CSS e Javascript, e das disciplinas acessibilidade e usabilidade no WARAU. Os tópicos contidos no WARAU tratam estritamente de assuntos relacionados à adequação de *websites* a requisitos de acessibilidade e usabilidade. Por este motivo, o conteúdo do WARAU (Figura C.9) conta com menos elementos do que os tutoriais do W3Schools (Figura C.8). Ainda, é possível verificar na Figura C.9 que a ligação entre tópicos de diferentes linguagens e disciplinas está representada em diversas páginas, de maneira contextualizada, combinando diversos tópicos em vez de remeter somente a índices, sendo esta uma das principais características possibilitadas pela estruturação obtida com a DTD detalhada anteriormente. Complementarmente, tutoriais comumente tratam tópicos e temas de maneira independente; em vez desse tratamento, o conteúdo do WARAU possibilita que técnicas mais complexas sejam aplicadas a partir da construção de diferentes componentes (e.g., utilização de skip links depende apenas de conhecimentos básicos sobre âncoras e sobre acessibilidade Web).

As representações das Figuras C.8 e C.9 foram obtidas a partir do detalhamento da navegação pelo conteúdo das páginas, ou seja, sem levar em consideração links localizados em áreas específicas de navegação, de busca ou filtragem, possibilidades existentes nos dois *websites* representados, ou seja, duas páginas estarão relacionadas somente se o conteúdo de uma fizer referência explícita à outra, o fato de usarem a mesma navegação ou índice e de seus links estarem presentes nessa navegação ou índice não caracteriza um relacionamento em nosso estudo. Ainda, a Figura C.8 não contou com tópicos relacionados à acessibilidade e usabilidade, pois os tutoriais do W3Schools não integram esses conteúdos.

Para gerar as Figuras C.8 e C.9 foi utilizado o software Graphviz [1], que é uma ferramenta de código aberto de geração de imagens de grafos. Como entrada ela utiliza arquivos que descrevem a estrutura do grafo. Os arquivos de entrada utilizados neste trabalho foram criados manualmente, representando todos os links presentes no conteúdo das páginas analisadas e das dependências existentes entre os conteúdos, no caso do WARAU. Entre as opções de geração de grafos disponibilizados pelo Graphviz, foi utilizado o neato.

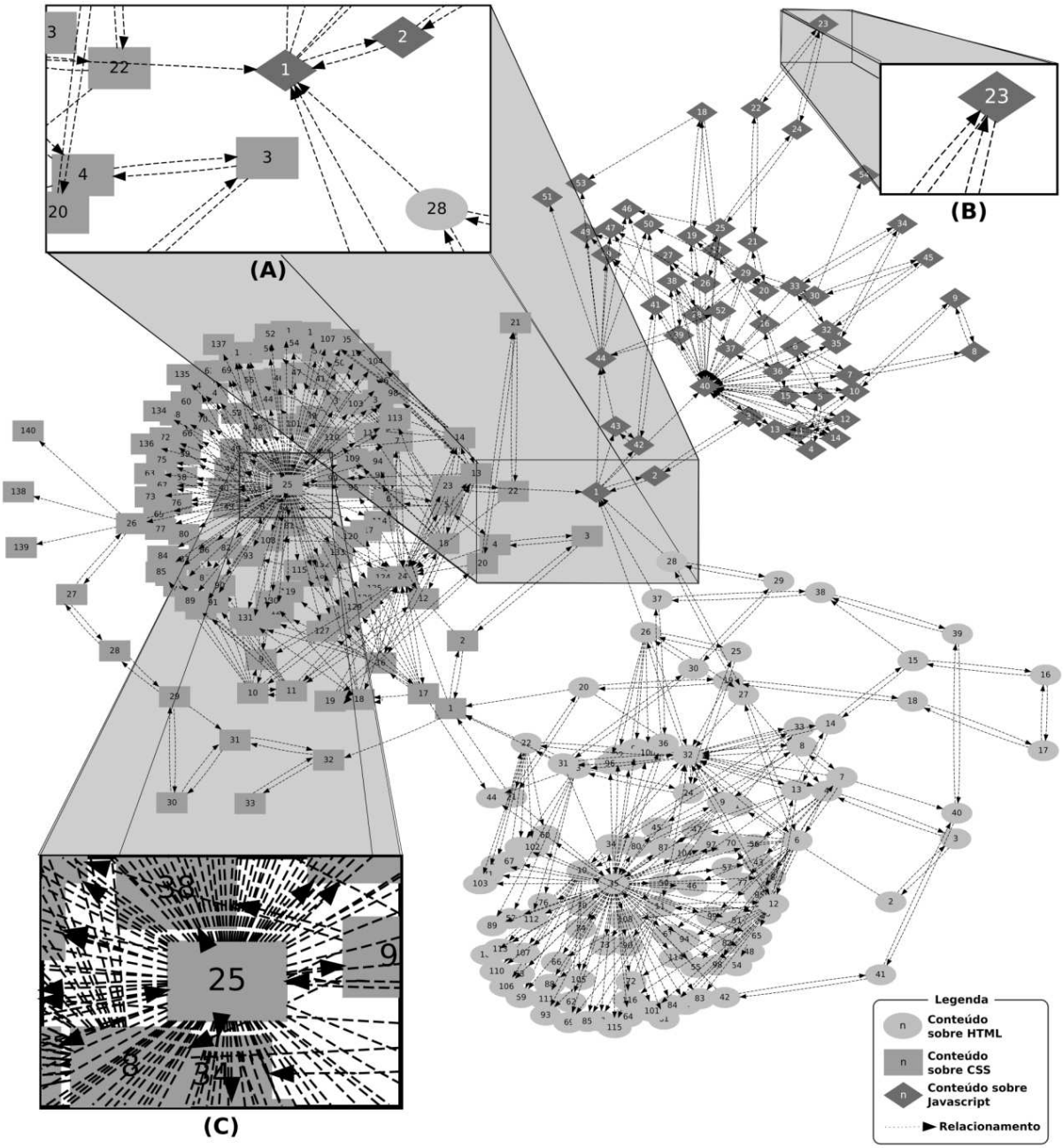


Figura C.8: Representação gráfica do conteúdo dos tutoriais do W3Schools. Com destaque para: (A) Relação entre páginas de tutoriais diferentes; (B) Representação da navegação com links "anterior" e "próximo"; (C) Representação da navegação em página de índice.

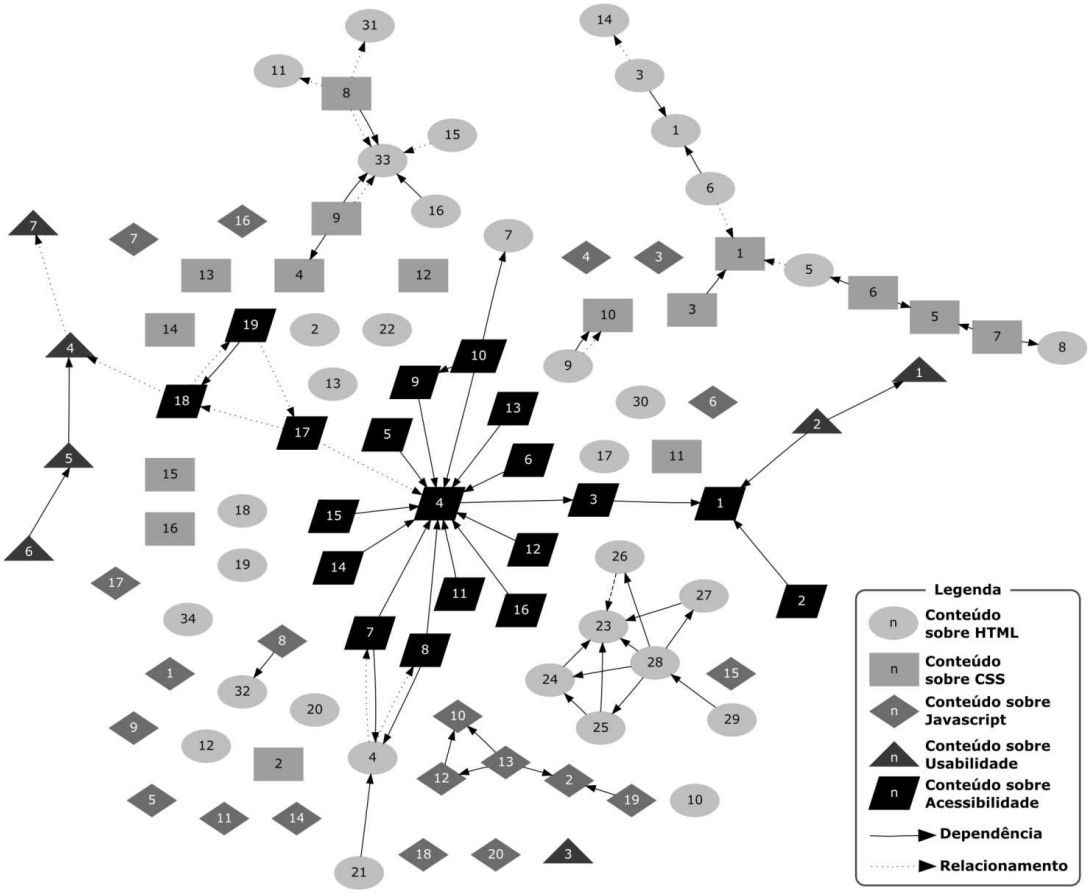


Figura C.9: Representação gráfica do relacionamento entre os tópicos de conteúdo do WARAU.

C.5 Conclusão

A Internet é parte essencial da vida social, econômica e educacional na sociedade contemporânea. Assim, garantir que pessoas com algum tipo de limitação tenham acesso à informação nessa mídia é essencial. A maioria dos *websites*, mesmo aqueles das melhores universidades no mundo, ainda apresenta problemas de acessibilidade, impedindo o acesso de todos à informação e serviços.

Aqueles que desejam criar *websites* acessíveis, em contextos educacionais, organizacionais ou pessoais contam com uma ampla variedade de recursos online, tutoriais e ferramentas de avaliação de acessibilidade. Diretrizes e heurísticas tendem a tratar aspectos tecnológicos de forma isolada dos aspectos conceituais. Além disso, apesar da disponibilidade desses recursos, muitos designers e desenvolvedores encontram dificuldade em situar

esse conhecimento na sua prática.

Este trabalho contribui na direção de facilitar o aprendizado de conceitos e técnicas para acessibilidade de *websites*, de forma articulada e contextualizada na prática de pequenas equipes de pessoas responsáveis pela criação e/ou manutenção de *websites*. Para isso apresenta o PAWRAU, um processo desenvolvido para o aprendizado de recursos de A&U necessários na criação e/ou manutenção de *websites* acessíveis. Esse processo é apoiado pelo WARAU, um ambiente na Web que oferece flexibilidade ao mantenedor de *websites* na construção de seu caminho de aprendizado. O WARAU está disponível na Web para uso e colaboração de todos aqueles que acreditam e buscam promover efetivamente o acesso de todos ao conhecimento via Internet.

Atualmente, tanto o documento XML com o conteúdo dos tópicos como o DTD que descreve o documento XML estão disponíveis para download gratuitamente na área de usuários cadastrados do *website*. Futuramente será disponibilizada para download uma ferramenta que possibilite a utilização do conteúdo sem estar conectado à Internet. Outros trabalhos previstos são a adição de outros filtros de busca (e.g., busca por resultados) e proporcionar maior participação dos usuários cadastrados no WARAU tanto para comentar e discutir conteúdos existentes como para sugerir novos tópicos a serem tratados.

Referências Bibliográficas

[1] Graphviz - Graph Visualization Software. `http://www.graphviz.org`.

[2] Section 508. `http://www.section508.gov/`.

[3] Julio Abascal, Myriam Arrue, Inmaculada Fajardo, Nestor Garay, and Jorge Tomas. The use of guidelines to automatically verify web accessibility. *Universal Access in the Information Society*, 3(1):71–79, 2004.

[4] AWARE (Accessible Web Authoring Resources and Education Center. Color laboratory. `http://colorlab.wickline.org/colorblind/colorlab/`, 2002.

[5] Leonelo Dell Anhol Almeida and Maria Cecilia Calani Baranauskas Vagner Figuerêdo de Santana. Um processo para adequação de websites a requisitos de acessibilidade e usabilidade. Technical Report IC-08-03, Instituto de Computação, Universidade Estadual de Campinas, January 2008.

[6] Deepak Alur, John Crupi, and Dan Malks. *Core J2EE Patterns: Best Practices and Design Strategies*. Prentice Hall PTR, 2nd edition, 2003.

[7] Ernesto Arroyo, Ted Selker, and Willy Wei. Usability tool for analysis of web designs using mouse tracks. In *Proceedings of ACM CHI 2006 Conference on Human Factors in Computing Systems*, volume 2 of *Work-in-progress*, pages 484–489, 2006.

[8] Brent Ashley. Shaping the future of secure ajax mashups. `http://www-128.ibm.com/developerworks/library/x-securemashups/`, 2007.

[9] Jorge Barbosa, Rodrigo Hahn, Solon Rabello, Sérgio Crespo Coelho da Silva Pinto, and Débora Nice Ferrari Barbosa. Computação móvel e ubíqua no contexto de uma graduação de referência. *Revista Brasileira de Informática na Educação*, 15(2):53–65, 2007.

[10] Bryan Basham, Kathy Sierra, and Bert Bates. *Head First Servlets and JSP: Passing the Sun Certified Web Component Developer Exam (SCWCD)*. O'Reilly Media, Inc., 2004.

[11] Giorgio Brajnik. Using automatic tools in accessibility and usability assurance processes. In *Lecture Notes in Computer Science Proceedings of the 8th ERCIM UI4ALL Workshop*. Springer Verlag, 2004.

[12] Acessibilidade Brasil. DaSilva - o primeiro avaliador de acessibilidade em português para websites. `http://www.dasilva.org.br`, 2006.

[13] BBC Brasil.com. Sites dobram em dois anos e beiram 120 milhões. `http://www.bbc.co.uk/portuguese/reporterbbc/story/2007/05/070516_internetcresce_pu.shtml`.

[14] Alan J. Broder. Data mining, the internet, and privacy. *Lecture Notes in Computer Science*, 1836:56–73, 2000.

[15] Adaptive Technology Resource Center. ATRC web accessibility checker. `http://checker.atrc.utoronto.ca`.

[16] Wendy Chisholm and Shawn Lawton Henry. Interdependent components of web accessibility. In Simon Harper, Yeliz Yesilada, and Carole A. Goble, editors, *Proceedings of the International Cross-Disciplinary Workshop on Web Accessibility, Chiba, Japan, May 10-14, 2005*, volume 88 of *ACM International Conference Proceeding Series*, pages 31–37. ACM, 2005.

[17] Mark Claypool, Phong Le, Makoto Wased, and David Brown. Implicit interest indicators. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pages 33–40, New York, NY, USA, 2001. ACM.

[18] Robert Cooley, Pang-Ning Tan, and Jaideep Srivastava. Discovery of interesting usage patterns from Web data. *Lecture Notes in Computer Science*, 1836:163–182, 2000.

[19] Microsoft Coorporation. What happened to operation aborted? `http://blogs.msdn.com/ie/archive/2008/04/23/what-happened-to-operation-aborted.aspx`, 2008.

[20] T.C. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. 1990.

[21] Francesco Correani, Barbara Leporini, and Fabio Paternò. Automatic inspection-based support for obtaining usable web sites for vision-impaired users. volume 5, pages 82–95. SpringerLink, 2006.

[22] Douglas Crockford. Private members in javascript. `http://www.crockford.com/javascript/private.html`, 2001.

[23] Douglas Crockford. Jsonrequest. `http://json.org/JSONRequest.html`, 2006.

[24] Douglas Crockford. The <module> tag. `http://www.json.org/module.html`, 2006.

[25] Presidência da República Casa Civil. Decreto n. 5.296 de 2 de dezembro de 2004. `http://www.planalto.gov.br/ccivil/_Ato2004-2006/2004/Decreto/D5296.htm`.

[26] Heloisa Vieira da Rocha and Maria Cecilia Calani Baranauskas. *Design e Avaliação de Interfaces*. NIED/UNICAMP, 2003.

[27] Antônio Carlos da Rocha Costa and Ana Carolina Bertoletti De Marchi. Cv-muzar - um ambiente de suporte a comunidades virtuais para apoio à aprendizagem em museus. *Revista Brasileira de Informática na Educação*, 14(3):9–26, 2006.

[28] Vagner Figuerêdo de Santana and Maria Cecilia Calani Baranauskas. An asynchronous client-side event logger model. Technical report, Institute of Computing (UNICAMP), 2008.

[29] Vagner Figuerêdo de Santana and Maria Cecilia Calani Baranauskas. A prospect of websites evaluation tools based on event logs. In Peter Forbrig, Fabio Paternò, and Annelise Mark Pejtersen, editors, *Human-Computer Interaction Symposium*, volume 272 of *IFIP International Federation for Information Processing*, pages 99–104. Springer Boston, 2008.

[30] Vagner Figuerêdo de Santana and Pollyana Notargiacomo Mustaro. Usabilidade é popular graças a seu retorno financeiro, diz jakob nielsen, 2007.

[31] Orçamento e Gestão Departamento de Governo Eletrônico do Ministério do Planejamento. Ases. avaliador e simulador de acessibilidade. `http://www.governoeletronico.gov.br/acoes-e-projetos/e-MAG/ases-avaliador-e-simulador-de-acessibilidade-sitios`.

[32] Orçamento e Gestão Departamento de Governo Eletrônico do Ministério do Planejamento. Padrões brasil e-GOV. `http://www.governoeletronico.gov.br/acoes-e-projetos/padroes-brasil-e-gov`.

[33] Orçamento e Gestão Departamento de Governo Eletrônico do Ministério do Planejamento. E-mag - modelo de acessibilidade de governo eletrônico. `http://www.governoeletronico.gov.br/acoes-e-projetos/e-MAG`, 2007.

[34] R. Diestel. *Graduate Texts in Mathematics: Graph Theory*. Springer, 2nd edition, 2000.

[35] Dojo Toolkit. Cross domain XMLHttpRequest using an IFrame Proxy. `http://dojotoolkit.org/node/87`, 2006.

[36] José M. Domenech and Javier Lorenzo. A tool for web usage mining. In Hujun Yin, Peter Tiño, Emilio Corchado, William Byrne, and Xin Yao, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2007, 8th International Conference, Birmingham, UK, December 16-19, 2007, Proceedings*, volume 4881 of *Lecture Notes in Computer Science*, pages 695–704. Springer, 2007.

[37] Michael Etgen and Judy Cantor. What does getting WET (web event-logging tool) mean for web usability? In *Proceedings of 5th Conference on Human Factors & the Web*, 1999.

[38] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object Oriented Software*. Addison Wesley, 1995.

[39] Mark Guzdial. Deriving software usage patterns from log files. Technical report, Georgia Institute of Technology, 1993.

[40] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[41] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. The MIT Press, 2001.

[42] Birgit Hay, Geert Wets, and Koen Vanhoof. Mining navigation patterns using a sequence alignment method. *Knowledge and Information Systems*, 6(2):150–163, 2004.

[43] David M. Hilbert and David F. Redmiles. Extracting usability information from user interface events. *ACM Comput. Surv.*, 32(4):384–421, 2000.

[44] Jason I. Hong, Jeffrey Heer, Sarah Waterson, and James A. Landay. Webquilt: A proxy-based approach to remote web usability testing. *ACM Transactions on Information Systems*, 19(3):263–285, 2001.

[45] Ed Huai hsin Chi, Adam Rosien, and Jeffrey Heer. Lumberjack: Intelligent discovery and analysis of web user traffic composition. In Osmar R. Zaïane, Jaideep Srivastava, Myra Spiliopoulou, and Brij Masand, editors, *WEBKDD*, volume 2703 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2002.

[46] Web Accessibility in Mind.

[47] IBGE Instituto Brasileiro de Geografia e Estatística. Censo 2000. `http://www.ibge.gov.br/`, 2000.

[48] International Standard Organization. Ergonomic requirements for office work with display terminals (VDTs). part 11: Guidance on usability, 1998.

[49] Melody Y. Ivory and Marti A. Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Comput. Surv.*, 33(4):470–516, 2001.

[50] JGraphT, 2005.

[51] Shaun K. Kane, Jessie A. Shulman, Timothy J. Shockley, and Richard E. Ladner. A web accessibility report card for top international university web sites. In Simon Harper and Yeliz Yesilada, editors, *Proceedings of the 2007 International Cross-Disciplinary Conference on Web Accessibility (W4A 2007), Banff, Canada, May 7-8, 2007*, volume 225 of *ACM International Conference Proceeding Series*, pages 148–156. ACM, 2007.

[52] Pythagoras Karampiperis and Demetrios G. Sampson. Designing learning systems to provide accessible services. In Simon Harper, Yeliz Yesilada, and Carole A. Goble, editors, *Proceedings of the International Cross-Disciplinary Workshop on Web Accessibility, Chiba, Japan, May 10-14, 2005*, volume 88 of *ACM International Conference Proceeding Series*, pages 72–80. ACM, 2005.

[53] Gez Lemon and Steve Faulkner. Making ajax work with screen readers. juicy studio. `http://juicystudio.com/article/making-ajax-work-with-screen-readers.php`.

[54] Barbara Leporini, Fabio Paternò, and Antonio Scorcia. An environment for defining and handling guidelines for the web. In Klaus Miesenberger, Joachim Klaus, Wolfgang L. Zagler, and Arthur I. Karshmer, editors, *Computers Helping People with Special Needs, 10th International Conference, ICCHP 2006, Linz, Austria, July 11-13, 2006, Proceedings*, volume 4061 of *Lecture Notes in Computer Science*, pages 176–183. Springer, 2006.

[55] Jason Levitt. Fixing AJAX: XMLHttpRequest considered harmful. `http://www.xml.com/pub/a/2005/11/09/fixing-ajax-xmlhttprequest-considered-harmful.html`, 2005.

[56] Jason Levitt. Json and the dynamic script tag: Easy, XML-less web services for Javascript. http://www.xml.com/pub/a/2005/12/21/json-dynamic-script-tag.html, 2005.

[57] Jason Levitt. Flash to the rescue. http://www.xml.com/pub/a/2006/06/28/flashxmlhttprequest-proxy-to-the-rescue.html, 2006.

[58] Kecheng Liu. *Semiotics in Information Systems Engineering*. Cambridge University Press, 2000.

[59] National Institute of Standards and Technology. WebVIP. http://zing.ncsl.nist.gov/WebTools/WebVIP/overview.html, 2002.

[60] Netscape Communications Corporation. *Client-Side JavaScript Reference*. 1999.

[61] BBC News. 'Most websites' failing disabled. http://news.bbc.co.uk/1/hi/technology/6210068.stm.

[62] World Health Organization. What is E-Accessibility?, 2006.

[63] Laila Paganelli and Fabio Paternò. Intelligent analysis of user interactions with web applications. In *IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces*, pages 111–118, New York, NY, USA, 2002. ACM.

[64] PubliAccesso.gov.it. Law n. 4. http://www.pubbliaccesso.gov.it/normative/law_20040109_n4.htm, 2004.

[65] Jeffrey Rubin. *Handbook Of Usability Testing: How to plan, design, and conduct effective tests*. John Wiley & Sons Inc, 1st edition, 1994.

[66] Jesse Ruderman. Signed scripts in Mozilla. http://www.mozilla.org/projects/security/components/signed-scripts.html, 2007.

[67] Jesse Ruderman. The Same Origin Policy. http://developer.mozilla.org/En/Same_origin_policy_for_JavaScript, 2008.

[68] Anthony Savidis, Dimitris Grammenos, and Constantine Stephanidis. Developing inclusive e-learning and e-entertainment to effectively accommodate learning difficulties. *Universal Access in the Information Society*, 5(4):401–419, 2007.

[69] Cyrus Shahabi and Farnoush Banaei-Kashani. A framework for efficient and anonymous web usage mining based on client-side tracking. In *Lecture Notes in Artificial Intelligence: Proceedings of WEBKDD 2001*, volume 2356 of *Lecture Notes in Computer Science*, pages 113–144. Springer-Verlag, 2001.

[70] D. Skeen. Eye-tracking device lets billboards know when you look at them. `http://www.wired.com/gadgets/miscellaneous/news/2007/06/eyetracking`, 2007.

[71] David Sloan, Andy Heath, Fraser Hamilton, Brian Kelly, Helen Petrie, and Lawrie Phipps. Contextual web accessibility - maximizing the benefit of accessibility guidelines. In Simon Harper, Yeliz Yesilada, and Carole A. Goble, editors, *Proceedings of the 2006 International Cross-Disciplinary Workshop on Web Accessibility (W4A 2006): Building the mobile web: rediscovering accessibility? Edinburgh, UK, May 22, 2006*, volume 134 of *ACM International Conference Proceeding Series*, pages 121–131. ACM, 2006.

[72] M. Spiliopoulou and L. C. Faulstich. WUM: A tool for Web utilization analysis. *Lecture Notes in Computer Science*, 1590:184–203, 1999.

[73] Ronald K. Stamper. A semiotic theory of information and information systems / applied semiotics. In *Invited papers for the ICL/University of Newcastle Seminar on 'Information', September 6–10, 1993*, 1993.

[74] Ronald K. Stamper. *Signs of Work: Semiosis and Information Processing in Organisations*. Walter de Gruyter, 1996.

[75] Ronald K. Stamper. Extending semiotics for the study of organisations. In *Proceedings of Conference on Semiotics and the Information Sciences*, 1998.

[76] W3Schools Online Web Tutorials. `http://www.w3schools.com/`, 1999.

[77] Takayuki Watanabe and Masahiro Umegaki. Capability survey of user agents with the UAAG 1.0 test suite and its impact on web accessibility. *Universal Access in the Information Society*, 6(3):221–232, 2007.

[78] Web Accessibility Initiative. Quick tips to make accessible web sites. `http://www.w3.org/WAI/quicktips/Overview.php`, 2001.

[79] Web Accessibility Initiative. Introduction do web accessibility. `http://www.w3.org/WAI/intro/accessibility.php`, 2005.

[80] Wickline.org. Colorblind web page filter. `http://colorfilter.wickline.org/`.

[81] Daniel Woo and Joji Mori. Accessibility: A tool for usability evaluation. In Masood Masoodian, Steve Jones, and Bill Rogers, editors, *Computer Human Interaction, 6th Asia Pacific Conference, APCHI 2004, Rotorua, New Zealand, June 29 - July 2, 2004, Proceedings*, volume 3101 of *Lecture Notes in Computer Science*, pages 531–539. Springer, 2004.

[82] World Wide Web Consortium. WCAG - Web Content Accessibility Guidelines 1.0. http://www.w3.org/TR/WCAG10/, 1999.

[83] World Wide Web Consortium. ATAG - Authoring Tool Accessibility Guidelines 1.0. http://www.w3.org/TR/ATAG10/, 2000.

[84] World Wide Web Consortium. UAAG - User Agent Accessibility Guidelines 1.0. http://www.w3.org/TR/UAAG10/, 2002.

[85] World Wide Web Consortium. UAAG - User Agent Accessibility Guidelines 2.0. http://www.w3.org/TR/UAAG20/, 2008.

[86] World Wide Web Consortium. WCAG - Web Content Accessibility Guidelines 2.0. http://www.w3.org/TR/2008/CR-WCAG20-20080430/, 2008.

[87] World Wide Web Consortium. HTML 5 differences from HTML 4. http://dev.w3.org/html5/html4-differences/, 2009.

Anexo A

Autorizações para publicação

Gmail by Google BETA

Vagner Santana <santana.vagner@gmail.com>

Permission to incorporate a book chapter into a Master thesis

Vagner Santana <santana.vagner@gmail.com> **19 de fevereiro de 2009 12:49**
Para: permissions.heidelberg@springer.com
Cc: cecilia@ic.unicamp.br

Dear Sir:

I am writing to request Springer's permission to incorporate into my Master thesis the following book chapter:

Title: A Prospect of Websites Evaluation Tools Based on Event Logs
Authors: Vagner Figuerêdo de Santana and M. Cecilia C. Baranauskas
Book title: Human-Computer Interaction Symposium - IFIP 20th World Computer Congress, Proceedings of the 1st TC 13 Human-Computer Interaction Symposium (HCIS 2008), September 7-10, 2008, Milano, Italy
Volume: 272/2008
Book series: IFIP International Federation for Information Processing
Editors: Peter Forbrig, Fabio Paternò, and Annelise Mark Pejtersen
ISBN: 978-0-387-09677-3
ISSN: 1571-5736
DOI: 10.1007/978-0-387-09678-0
Pages: 99-104

The material will be used as part of a Master thesis of the Institute of Computing at University of Campinas (UNICAMP), Brazil.

If you do not solely control copyright in the requested materials, I would appreciate any information you can provide about others to whom I should write.

Yours sincerely,

Vagner Figuerêdo de Santana
MSc Student
Rua Ana Cintra, 67, ap. 21
São Paulo - SP, Brazil, CEP 01201-060
Fax: +55 19 3521-5847
santana.vagner@gmail.com

M. Cecilia C. Baranauskas
Professor at the Institute of Computing
UNICAMP - Caixa Postal 6176
Campinas - SP, Brazil, CEP 13084-971
Phone: +55 19 3521-5845
cecilia@ic.unicamp.br

Gmail by Google BETA

Vagner Santana <santana.vagner@gmail.com>

Permission to incorporate a book chapter into a Master thesis

Essenpreis, Alice, Springer DE <Alice.Essenpreis@springer.com> **19 de março de 2009 07:24**
Para: santana.vagner@gmail.com, cecilia@ic.unicamp.br

Dear Vagner Figuerêdo de Santana,
Dear M. Cecilia C. Baranauskas,

Thank you for your e-mail.

With reference to your request (copy herewith) to re-use material on which Springer controls the copyright, our permission is granted free of charge, on the following condition:

* full credit (book title, volume, year of publication, page, chapter/article title, name(s) of author(s), figure number(s), original copyright notice) is given to the publication in which the material was originally published by adding: With kind permission of Springer Science+Business Media.

With best regards,

–

Alice Essenpreis
Springer
Rights and Permissions

–

Tiergartenstrasse 17 | 69121 Heidelberg GERMANY
FAX: +49 6221 487 8223
permissions.Heidelberg@springer.com
WWW.springer.com/rights

–

-----Original Message-----
From: SpringerAlerts@springeronline.com [mailto:SpringerAlerts@springeronline.com]
Sent: Monday, March 02, 2009 4:06 PM
To: Permissions Heidelberg, Springer DE
Subject: Permission to incorporate a book chapter into a Master thesis

Dear Sir:

[Texto das mensagens anteriores oculto]

[
sender name: Vagner Figuerêdo de Santana
sender email: santana.vagner@gmail.com
INTERNAL NAME: Rights and Permissions Department - Springer Verlag Heidelberg
ORIGINAL URL: http://www.springer.com/rights?SGWID=0-122-19-161426-0
]