UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

A3 - AVALIAÇÃO AUTOMÁTICA DE ACESSIBILIDADE: TÉCNICAS E RELATÓRIOS ADEQUADOS AOS NOVOS DESAFIOS WEB

Sérgio José Inácio das Neves

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

2012

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

A3 - AVALIAÇÃO AUTOMÁTICA DE ACESSIBILIDADE: TÉCNICAS E RELATÓRIOS ADEQUADOS AOS NOVOS DESAFIOS WEB

Sérgio José Inácio das Neves

DISSERTAÇÃO

Trabalho orientado pelo Prof. Doutor Luís Manuel Pinto da Rocha Afonso Carriço

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Sistemas de Informação

2012

Agradecimentos

A realização deste trabalho não teria sido possível se não fosse o meu orientador, o Professor Luís Carriço, e a minha colega Nádia Fernandes, que sempre me deram indicações no sentido de abordar e resolver os problemas de forma mais adequada. Muito obrigado aos dois.

Estendo os agradecimentos à minha família em geral, e à minha mãe em particular, que lutou com todas as forças para que o meu curso fosse concluído com sucesso.

Agradeço também ao Celso Sousa, amigo e camarada de luta, com quem fiz parceria na maior parte dos trabalhos de grupo e que me ajudou a resolver muitos problemas e a enfrentar novas situações. Sem ele, tudo seria bem mais difícil.

Resumo

Hoje em dia assiste-se à massificação na produção de conteúdos para a Web. Não existe nenhuma autoridade que dite quem pode ou não publicar conteúdos, comprometendo a qualidade destes. Por outro lado, um número crescente de pessoas, incluindo pessoas com necessidades especiais, acede à Web, pelo que é fundamental que os conteúdos sejam acessíveis.

Este relatório visa descrever os objectivos e o trabalho desenvolvido no âmbito da tese de mestrado. Genericamente, o trabalho destina-se a estender uma ferramenta de avaliação de acessibilidade Web, o QualWeb, segundo as seguintes linhas fundamentais: 1) a implementação de mais técnicas WCAG 2.0, tendo em vista uma avaliação mais exaustiva; 2) a concepção e desenvolvimento de uma interface facilmente usável onde serão apresentados os resultados da avaliação, enriquecida com referências explícitas ao código Web que gerou a notificação de erro, alerta ou conformidade.

Palavras-chave: acessibilidade web, avaliação automática, relatórios de resultados, interfaces

Abstract

Nowadays, we are witnessing a mass of production of web contents. There is no authority who oversees who can or cannot publish contents, compromising their quality. On the other hand, an increasing number of people, including people with special needs, access web, so it's fundamental that contents are accessible.

This report aims to describe the goals of the developed work in the scope of master thesis. Generically, the work aims to extend an automatic Web accessibility evaluation software, QualWeb, according to the following fundamental lines: 1) the implementation of more WCAG 2.0 techniques, in order to have a more thorough evaluation; 2) the conception and development of an easy to use interface, where the evaluation results will be presented, enriched with explicit references to the web code that generated the notification error, alert or compliance.

Keywords: web accessibility, automatic evaluation, results' reports, interfaces

Conteúdo

Lista de Figuras

Lista de Tabelas

Tabela 2.1: Ferramentas de Avaliação de Acessibilidade Web

Capítulo 1

Introdução

1.1 Motivação

Nos últimos anos, tem-se assistido a um aumento exponencial na produção de conteúdos para a web, relativos a todas as áreas da nossa vida. Desde o comércio, serviços de telecomunicações, media e todas as áreas de negócio, a todos os serviços estatais, passando pelas redes sociais, páginas pessoais e inúmeros blogs, a informação existente aumenta de dia para dia.

Sendo a Web um dos meios de comunicação mais importantes dos tempos actuais, torna-se vital que os conteúdos aí presentes sejam, para além de usáveis, acessíveis.

A acessibilidade Web visa estudar a adequação, quer dos conteúdos para a Web, quer das tecnologias utilizadas para os produzir, às pessoas com qualquer tipo de incapacidade (física, sensorial, cognitiva, entre outras).

As pessoas com necessidades especiais utilizam tecnologias de apoio, que são ferramentas que visam promover a sua independência nas tarefas que, de outro modo, seriam de execução difícil ou impossível, através da estimulação das faculdades existentes, procurando minimizar o impacto temporário ou permanente do défice mais ou menos elevado noutra(s) faculdade(s).

No que diz respeito à acessibilidade de conteúdos Web, as tecnologias de apoio têm de extrair a informação pertinente das páginas e apresentá-la aos utilizadores em pelo menos um modo perceptível (por exemplo, áudio, Braille e ampliação de texto). Daqui decorre que, para além da necessidade de adequação dos conteúdos às necessidades, as tecnologias subjacentes têm de ser compatíveis com as tecnologias de apoio.

De seguida apresentam-se as motivações fundamentais para a acessibilidade Web:

1. Do ponto de vista ético é o mais correcto: com efeito, o direito ao acesso à informação é fundamental e não deve ser vedado a nenhum ser humano. Para além disso, todos devemos ter consciência que ninguém está imune a sofrer de

alguma incapacidade, quer por acidente, quer pelo desgaste físico provocado pelo aumento da idade, e quando isso acontecer, ninguém gostará de se confrontar, de um momento para o outro, com barreiras no acesso à informação.

2. Possibilitar o aumento significativo de clientes das empresas e o aumento dos lucros: o número de pessoas com incapacidades e com um papel activo no consumo de bens e serviços é suficientemente grande para não se desprezar. Se estas pessoas se deparam com a inacessibilidade das páginas de uma empresa, rapidamente vão às páginas das empresas concorrentes, o que pode significar a perda de muitos clientes.

3. Promover o aumento de competências: para se compreender o conceito de conteúdo acessível é necessário, no mínimo, ter um conjunto de conhecimentos que abrange:
    a. Saber como os utilizadores interagem com as páginas, quais as suas preferências e o porquê da sua eleição. O modo de interacção depende, não só da incapacidade que se está a analisar, mas mesmo dentro de cada incapacidade existem inúmeros factores que alteram a forma de interacção, tais como a tecnologia de apoio e o navegador utilizados.
    b. Estar permanentemente informado, quer dos avanços no desenvolvimento das principais tecnologias Web e tecnologias de apoio (aparecimento de tecnologias novas, versões mais recentes de tecnologias já existentes), quer da compatibilidade entre elas.
    c. Conhecer, de um modo global, os padrões existentes para o desenvolvimento de conteúdos acessíveis e saber aplicá-los quando necessário.

    Todo este conhecimento é uma mais-valia para quem o possui, podendo tirar daí muitos benefícios.

4. Facilitar a indexação dos conteúdos pelos motores de busca: grande parte dos problemas enfrentados pelas pessoas com incapacidades são partilhados pelos motores de busca. Por exemplo: tal como as pessoas cegas, os motores de busca não compreendem imagens, a não ser que seja facultada uma descrição num formato perceptível, que no caso dos motores de busca tem de ser

textual; tal como as pessoas surdas, os motores de busca não reconhecem sons, a não ser que sejam descritos noutro formato, que para os motores de busca tem de ser textual.

Se estas regras forem cumpridas, todos os tipos de media poderão ser indexados, o que terá como consequência o aumento das classificações das páginas nas listas de resultados e, por conseguinte, um maior número de visitas.

5. Favorecer a acessibilidade dos conteúdos face à tecnologia utilizada: se os conteúdos são adequados às incapacidades, provavelmente utilizam padrões, o que provoca a diminuição de barreiras no acesso aos conteúdos, independentemente de estes serem apresentados em diferentes navegadores, em diferentes sistemas operativos com o mesmo navegador ou navegadores diferentes, em dispositivos móveis, em situações em que a largura de banda é reduzida, ou simplesmente porque uma tecnologia dentro de um navegador (por exemplo, javascript) não está disponível. Note-se, no entanto, que não é expectável que, ao ser alterada uma destas variáveis, o nível de experiência de utilização se mantenha, mas é fundamental que os conteúdos estejam disponíveis e acessíveis, seja qual for a riqueza visual e interactiva suportada pelo ambiente.

6. Cumprir as leis já existentes no campo da acessibilidade: apesar de ser um dos factores mais importantes com que nos devemos preocupar, não deverá ser o mais inspirador e motivador (pressupõe obrigação e não gosto), daí ter ficado em último lugar. Já existem em muitos países leis que regulamentam a acessibilidade dos conteúdos Web. Por exemplo, no caso de Portugal, deve referir-se a Resolução do Conselho de Ministros nº 155/2007 [14], que determina que os sítios da internet do Governo e dos serviços e organismos públicos da Administração Central devem ser acessíveis; no caso dos Estados Unidos, a lei mais conhecida é a “Section 508 of Reabilitation Act” [15].

Para promover uma correcta adequação dos conteúdos e tecnologias Web às diferentes incapacidades e a sua compatibilidade com as tecnologias de apoio, são utilizados standards. Os mais conhecidos e utilizados são desenvolvidos e promovidos através da Web Accessibility Initiative (WAI) do World Wide Web Consortium (W3C).

O W3C agrega vários grupos de trabalho, responsáveis pela elaboração de recomendações e outros documentos relacionados com os diferentes componentes de acessibilidade Web: conteúdo Web, ferramentas e standards de avaliação, formatos de relatório de testes, navegadores, media players e ferramentas de autoria.

Neste trabalho só serão considerados os documentos referentes às três primeiras componentes mencionadas. O documento responsável por garantir a acessibilidade dos conteúdos Web designa-se por Directrizes de Acessibilidade para o Conteúdo da Web e foi criado pelo W3C. Em 1999, foi lançada a versão 1.0 deste documento [2], que dava uma resposta aceitável às necessidades da altura. No entanto, com a evolução das tecnologias Web, o conhecimento entretanto adquirido sobre a aplicação das normas WCAG 1.0, e as novas formas de desenvolvimento (conteúdos cada vez mais dinâmicos), surgiu a necessidade de elaborar a versão 2.0 [3], que chegou ao estado de recomendação em Dezembro de 2008.

A versão 2.0 das directrizes baseia-se em quatro princípios que os conteúdos devem satisfazer para serem considerados acessíveis. Assim, o conteúdo acessível deve ser:

- Perceptível - A informação e os componentes da interface de utilizador têm de ser apresentados aos utilizadores de formas perceptíveis.
- Operável - Os componentes da interface de utilizador e a navegação têm de ser operáveis.
- Compreensível - A informação e a operação da interface de utilizador têm de ser compreensíveis.
- Robusto - O conteúdo tem de ser suficientemente robusto para ser interpretado, com precisão, por uma grande variedade de agentes de utilizador, incluindo tecnologias de apoio.

Distribuídas por estes princípios existem doze directrizes. Cada uma das directrizes tem vários critérios de sucesso (predicados). Todos os critérios (os que são aplicáveis à página em avaliação) têm de ser verdadeiros para que essa página esteja conforme com a directriz de que fazem parte. Para facilitar na satisfação dos critérios durante o processo de desenvolvimento e para verificar se foram satisfeitos, foi desenvolvido um outro documento [5], onde constam técnicas com base em linguagens específicas (HTML, CSS, javascript, entre outras). Estas técnicas dividem-se em técnicas suficientes (apesar de não ser obrigatório utilizá-las, existe a certeza que se forem usadas, o critério a que dizem respeito é satisfeito) e aconselháveis (apesar de não influenciarem o nível de conformidade, devem ser utilizadas nas situações em que for possível, a fim de ser proporcionado um nível maior de acessibilidade).

Estas técnicas constituem uma forma mais concreta de avaliação das páginas Web, ao invés de se usarem critérios muito genéricos, e potenciam o uso de ferramentas automáticas de avaliação. Estas ferramentas podem identificar os locais no código fonte onde as várias técnicas são bem aplicadas ou não, e apresentar os resultados numa interface amigável aos utilizadores, de modo a que seja possível corrigir com facilidade os problemas encontrados.

1.2 Enquadramento

O trabalho apresentado neste relatório enquadra-se em projectos do grupo HCIM (Human Computer Interaction and Multimedia) do LaSIGE (Laboratório de Sistemas Informáticos de Larga Escala). Os projectos de base da dissertação abordam precisamente a acessibilidade Web e a criação de ferramentas automáticas que sigam as recomendações do W3C. Destes projectos emergiu recentemente uma nova abordagem à avaliação de acessibilidade Web [16, 18], que aproxima a avaliação das recomendações vigentes, neste caso WCAG 2.0, e do que seria verdadeiramente efectuado se a avaliação fosse efectuada por peritos, i.e., a avaliação recai sobre o que é apresentado aos utilizadores. Como prova de conceito desta abordagem foi concretizada a ferramenta QualWeb [16], que implementa parcialmente algumas das técnicas WCAG 2.0, permitindo proceder de imediato a estudos em grande escala de propriedades da acessibilidade Web, que aliás demonstram, a validade da aproximação.

No entanto a ferramenta apresenta duas questões fundamentais a resolver: 1) a ainda limitada concretização das técnicas WCAG 2.0, válida enquanto prova de conceito; e a apresentação de resultados distante do suporte necessário à correcção do código, decorrente da aproximação que adopta. Estas são, por conseguinte, duas componentes a perseguir para obter uma ferramenta que simultaneamente sirva novas dimensões de investigação e uma aplicação prática e por conseguinte fortemente disseminada.

1.3 Objectivos

Este trabalho destina-se a estender uma ferramenta automática de avaliação de acessibilidade Web, o QualWeb evaluator, cuja implementação foi iniciada em 2010. O propósito é obter uma ferramenta robusta, abrangente e que permita simultaneamente a procura sistemática e expedita de problemas de acessibilidade em grandes quantidades de sítios Web, com o objectivo de estender estudos macroscópicos de acessibilidade, e a avaliação individual de páginas e sítios, com o intento de prosseguir a sua correcção. Em geral, pois, o resultado deste trabalho pretende servir: quer utilizadores finais, identificando a acessibilidade de um sítio ou página; quer investigadores ou entidades

reguladoras, encontrando tendências, valores médios detalhados de acessibilidade; quer programadores e designers, indicando onde no código e como se devem resolver os problemas de acessibilidade.

A extensão ao QualWeb norteia-se então por duas grandes linhas de acção:

1. Implementação validada de um conjunto mais alargado de técnicas dodocumento “Techniques for WCAG 2.0” [5]
    a. Actualmente, encontram-se especificadas 60 técnicas de HTML no documento “Techniques for WCAG 2.0” [5]. Dessas 60 técnicas foram implementadas 18 no trabalho realizado anteriormente. Neste trabalho foram desenvolvidas mais 7 técnicas HTML, muitas delas de maior complexidade. Foi também iniciado o desenvolvimento de técnicas CSS, tendo em vista uma avaliação mais exaustiva dos conteúdos Web, com a implementação de 7 técnicas de um total de 22 técnicas CSS existentes no documento referido.

    b. Paralelamente, procedeu-se ao alargamento da bateria de código Web de teste para as novas técnicas implementadas, para se manter a garantia da correcta implementação de todas elas.

2. Desenvolvimento de várias interfaces apresentadas nos navegadores Web:
    a. No trabalho realizado anteriormente, foi decidido guardar os resultados da avaliação da acessibilidade das páginas Web num formato denominado Evaluation and Report Language (EARL) [4]. Este formato, criado pelo W3C e baseado em XML, facilita o processamento por parte dos programas, promove a utilização dos resultados por outros programas que também o utilizem, para além de possibilitar a comparação de resultados de programas que executem as mesmas tarefas. No entanto, se para os programadores ou utilizadores em geral a leitura dos resultados neste formato é muito pouco prática, para a maioria dos utilizadores finais é impraticável ou impossível.

    Com o objectivo de resolver este problema foram criadas várias interfaces baseadas no navegador, que apresentam, de forma usável e acessível, os resultados obtidos na avaliação das várias técnicas.

b. Ao contrário da maioria das ferramentas de avaliação de acessibilidade Web, o QualWeb actua sobre o código Web depois de processado pelo navegador (vulgo browser), proporcionando assim uma avaliação mais correcta [16]. Esta vantagem, todavia, traduz-se numa na dificuldade de transmitir o local exacto do erro no código, tendo em conta o processamento, por vezes complexo, do navegador. Esta informação, não fazendo parte da especificação do formato originalmente escolhido no QualWeb, não foi desenvolvida.

Assim, no contexto deste trabalho foi implementado um modo de incluir a informação de localização do código original que deu origem ao relatório de cada problema encontrado numa página ou num sítio Web. É de referir que o processamento feito pelo navegador pode incluir diversas formas de código Web cliente, desde HTML a javascript.

1.4 Estrutura do documento

Este documento está organizado nos seguintes capítulos:

- O capítulo 1 visa estabelecer a motivação para a realização deste projecto, bem como os seus objectivos.
- O capítulo 2 fornece informação sobre o trabalho relacionado na área de enquadramento deste trabalho – avaliação de acessibilidade para a Web e interfaces de apresentação de relatórios de testes.
- Segue-se o capítulo 3, onde é descrito o avaliador QualWeb implementado anteriormente e tomado como base para a concretização da extensão e mencionados os requisitos do trabalho actual.
- O capítulo 4 visa descrever pormenorizadamente todo o trabalho realizado tendente à implementação dos requisitos mencionados no capítulo anterior, bem como o contributo do mesmo para a comunidade científica.
- Posteriormente é apresentado um capítulo de validação, tendo em vista dois grandes objectivos: por um lado, provar o funcionamento correcto das várias interfaces, mostrando os resultados de avaliação das páginas mais acedidas a nível mundial nas várias formas de visualização de erros; por outro lado, provar o correcto funcionamento do algoritmo de enriquecimento de informação de localização de erros de acessibilidade,

apresentando um conjunto de ficheiros de teste sob a forma original e sob a forma modificada.

- Este relatório termina com um capítulo de conclusão, onde são explanadas as conclusões decorrentes do trabalho realizado, nomeadamente o trabalho futuro a desenvolver.

Capítulo 2

Trabalho Relacionado

Este capítulo pretende dar a conhecer os trabalhos já realizados e em curso na área da acessibilidade Web, nos campos da avaliação da acessibilidade de páginas Web e construção de interfaces para apresentação de erros.

Em primeiro lugar serão descritos em pormenor os vários processos de avaliação de acessibilidade de páginas Web, sendo realçadas as vantagens e desvantagens de cada um e discutido o processo de avaliação automática, sendo confrontadas duas abordagens para o realizar. Seguidamente, será enfatizado um artigo onde são discutidas formas adequadas de implementar interfaces para apresentação de erros. Por último, serão descritas ferramentas de avaliação automática já existentes e as funcionalidades suportadas e não suportadas.

2.1 Avaliação de Acessibilidade de Conteúdos Web

A avaliação de acessibilidade de páginas Web pode ser realizada de três formas: por peritos, por utilizadores finais (avaliação manual), ou através de software (avaliação automática), tendo cada uma as suas vantagens e desvantagens.

A avaliação conduzida por peritos possibilita um conhecimento muito aprofundado sobre a acessibilidade das páginas em análise, no entanto não é escalável, dado o número abismal de páginas existente e o tempo que cada uma demora a ser analisada, para além de trazer algumas ambiguidades sobre se determinada funcionalidade é ou não acessível.

A avaliação realizada por utilizadores finais tem como grande diferencial o facto de o feedback recebido estar intimamente relacionado com as necessidades dos mesmos, mas tem um inconveniente que assenta na falta de conhecimento destes relativamente às tecnologias, o que poderá originar por vezes algumas dúvidas de os problemas encontrados serem mesmo de falta de acessibilidade, ou resultarem apenas do desconhecimento das tecnologias de apoio, navegadores ou outras tecnologias Web.

A avaliação por software tem como pontos fortes a grande escalabilidade e a inexistência de ambiguidades, mas peca por não ser tão exaustiva como a efectuada por peritos e por ainda existirem muitas tarefas que não são realizáveis por software, como a compreensão de tipos de media complexos, como a imagem, o vídeo e o som.

O ideal seria complementar estas três formas de avaliação, pois só o conjunto está isento de pontos fracos. No entanto, quando o número de páginas em estudo é considerável, como em [6], é necessário recorrer, forçosamente, à avaliação por software.

Quando se refere a avaliação por software, é importante definir qual o seu alvo. Até ao virar do milénio, ainda existiam muitas páginas estáticas: o navegador fazia o pedido, recebia o ficheiro HTML juntamente com outros recursos (estilos e vários tipos de media), e apresentava-os tal como vinham, sem qualquer processamento.

Hoje em dia, a maior parte das páginas que são apresentadas aos utilizadores não correspondem aos conteúdos que são transferidos para o navegador, porque existe código javascript que é executado depois de o ficheiro chegar e antes da apresentação e que pode alterar elementos estruturais (HTML) ou de apresentação (CSS).

Daqui decorre que, nos tempos actuais, já não faz sentido que a avaliação seja efectuada sobre o código fonte dos ficheiros transferidos para o navegador, mas sim sobre o conteúdo que é apresentado no navegador após o processamento, dado que é esse conteúdo que é visto e é alvo de interacção por parte dos utilizadores. No entanto, e como será realçado mais adiante, ainda existem muito poucas ferramentas a adoptar esta abordagem de avaliação automática.

2.2 Interfaces com os resultados da avaliação

Segundo Law et al. [7], os relatórios de avaliação de acessibilidade devem ser compreensíveis, tanto pelos gestores técnicos, como pelos programadores. Devem listar os problemas existentes, com uma ou duas instâncias para exemplificar, em vez de apresentarem uma longa lista de instâncias do mesmo problema. Referem os autores que estes utilizadores de ferramentas de avaliação gostam mais de lidar com números do que com situações abstractas. Sugerem então que os relatórios devem explicitar, para cada nível de conformidade, quantas directrizes estão satisfeitas e quantas não se aplicam (nenhum trabalho para o programador), quantas estão satisfeitas parcialmente (por falta de consistência, por exemplo) e quantas não estão satisfeitas (uma minoria).

Em geral, no entanto, o estudo não é definitivo. Primeiro aborda sobretudo programadores, não designers, que não são especialistas de acessibilidade. Ora essa situação, por exemplo em equipas dedicadas, tende a alterar-se gradualmente, expondo

um leque alargado de perícia de acessibilidade. Por outro lado com ferramentas adequadas muita da programação é feita por não programadores ou pelo menos programadores que dificilmente caem no perfil apresentado. Finalmente existe uma miríade de representantes de utilizadores e utilizações de ferramentas de avaliação de acessibilidade Web que não se coadunam com a imagem traçada por aqueles autores. Como exemplo temos as entidades reguladoras, ou os utilizadores finais.

Ferramentas como o Accessibility Evaluation Assistant (AEA), por exemplo, suportam uma maior flexibilidade de formas de apresentação que se coadunam com as diferentes perspectivas e usos que se pode fazer destas ferramentas [17]. Se considerarmos, para além disso, outros intervenientes, essa flexibilidade deverá ser ainda mais explorada.

2.3 Ferramentas de Avaliação de Acessibilidade Web

Na secção 2.1 foi referido que o alvo da avaliação por software deve ser o conteúdo que é apresentado após o processamento do navegador, dado que é esse conteúdo que é percepcionado pelos utilizadores e que necessita, efectivamente, de ser acessível. De facto, este é um dos requisitos que uma ferramenta de avaliação deve cumprir para que se mostre útil. Outros dois requisitos fundamentais são a produção de relatórios dos testes de acessibilidade em formato standard e o suporte à versão 2.0 das directrizes de acessibilidade de conteúdos para a Web.

O W3C disponibiliza uma página com muitas ferramentas de avaliação [1], mas nenhuma delas cumpre todos os requisitos acima descritos.

Começam já a aparecer algumas ferramentas que consideram a avaliação dos conteúdos após o processamento do navegador, mas também estas não cumprem todos os outros requisitos.

Na tabela 2.1 são apresentadas algumas (as mais relevantes) ferramentas de avaliação de acessibilidade Web, acompanhadas das funcionalidades suportadas.

Nome	**Avaliação no pós processamento**	**Directrizes WCAG 2.0**	**Relatórios em formato standard**	**Notas**
A-Checker [19]	Não	Não	Não	Utiliza as directrizes WCAG 1.0 e gera

				relatórios em HTML
TAW Standalone [20]	Não	Não	Sim	Utiliza as directrizes WCAG 1.0 e gera relatórios no formato EARL
Foxability [11]	Sim	Não	Não	Utiliza as directrizes WCAG 1.0
Functional Accessibility Evaluator [12]	Sim	Sim*	Não	* Utiliza técnicas alternativas às publicadas pelo W3C [13], encontra-se em actualização para as directrizes WCAG 2.0 e gera relatórios mas não num formato standard
WAVE - Web Accessibility Evaluation Tool [9]	Sim	Não	Não	Utiliza as directrizes WCAG 1.0
Hera-ffx [8]	Sim	Sim*	Sim	* Encontra-se em actualização para as directrizes WCAG 2.0 e gera relatórios em formato EARL, mas é uma ferramenta semi-automática que precisa de intervenção humana

Tabela 2.1: Ferramentas de Avaliação de Acessibilidade Web

Como se pode observar pela análise da tabela, das seis ferramentas apresentadas apenas quatro realizam a avaliação sobre o conteúdo resultante do processamento do navegador, apenas duas estão em actualização para as directivas WCAG 2.0 e apenas duas criam relatórios em formato standard. Para além disso, só uma delas cumpre os três requisitos explicitados, e mesmo essa é semi-automática.

O software QwalWeb, detalhado de modo pormenorizado no próximo capítulo, vem revolucionar o modo como é realizada a avaliação. Num dos modos de funcionamento desta ferramenta, a avaliação é efectuada após o processamento das páginas pelo navegador e dentro deste e não através de extensões, o que diminui a probabilidade de incompatibilidades do programa com novas versões do navegador. Para além disso, já foi realizado um estudo por alguns membros da equipa de desenvolvimento do QualWeb [16], em que se comprova que a avaliação da acessibilidade das páginas tal como são apresentadas pelo navegador após o processamento produz resultados mais credíveis do que a avaliação sobre o código fonte dos ficheiros que vêm do servidor como resposta aos pedidos. Para além disso, utiliza as directrizes de acessibilidade WCAG 2.0 e gera relatórios em formato standard (EARL).

2.4 Resumo

Este capítulo focou-se no trabalho relacionado que tem sido realizado na área de acessibilidade Web.

Foram discutidos pormenorizadamente os vários processos de avaliação de acessibilidade de páginas Web, referidas as vantagens e desvantagens de cada um e descrito detalhadamente o processo de avaliação automática de acessibilidade Web.

Seguidamente, foram mencionadas abordagens de implementação de interfaces para apresentação de problemas de acessibilidade.

Finalmente, foram apresentados os requisitos de uma ferramenta de avaliação automática de acessibilidade Web, descritas ferramentas de avaliação automática já existentes e o motivo pelo qual foi necessário implementar e estender a ferramenta objecto deste trabalho, o avaliador QualWeb.

O próximo capítulo concentrar-se-á na descrição detalhada da ferramenta QualWeb tal como foi implementada originalmente.

Capítulo 3

Descrição do Sistema

Este capítulo visa descrever a arquitectura original da ferramenta QualWeb, a interacção entre os vários componentes e as limitações desta ferramenta que estão na base do trabalho desenvolvido no âmbito desta tese.

3.1 Arquitectura da ferramenta QualWeb

A ferramenta QualWeb é constituída por quatro grandes componentes:

- Módulo fases de processamento;

- Módulo QualWeb;

- Módulo técnicas;

- Módulo formatadores (serialização de dados).

3.1.1 Módulo fases de processamento

Este módulo possibilita que a aplicação seja executada em dois modos distintos: antes e depois do processamento do navegador.

Caso a aplicação seja executada na fase pré-processamento, então a avaliação das páginas far-se-á sobre o documento HTML original, tal como vem do servidor Web; caso a aplicação seja executada na fase pós-processamento, então a avaliação será executada sobre o conteúdo das páginas após o processamento do navegador. A implementação destas duas fases de execução permitiu a realização de um estudo que teve como objectivo verificar que a avaliação pós-processamento é mais apropriada do que é realizada antes do processamento do navegador [16].

A principal função deste módulo é, portanto, passar ao módulo QualWeb o conteúdo adequado, consoante se trate de uma avaliação pré ou pós-processamento. Para que o módulo QualWeb possa ser executado independentemente do modo de avaliação, o conteúdo é sempre enviado sob a forma de uma árvore. A grande diferença

reside no conteúdo da árvore gerada, que no caso da avaliação pré-processamento contém todos os elementos do ficheiro HTML original e no caso da avaliação pós-processamento contém todos os elementos presentes no DOM gerado pelo navegador (estrutura em árvore construída pelos navegadores após o descarregamento dos ficheiros fonte, que representa todos os elementos presentes nas páginas e as relações entre eles, e que é utilizada pelos navegadores para efectuar toda e qualquer operação sobre as mesmas) após o processamento (elementos originais modificados pela adição, alteração ou remoção de novos elementos presentes em código javascript).

3.1.2 Módulo QualWeb

Este módulo é responsável pela preparação e envio do conteúdo alvo de avaliação ao módulo técnicas e pela execução do módulo formatadores.

No caso do ambiente de execução baseado no navegador, este módulo é mais complexo, baseando-se em duas componentes:

- Componente cliente: injecta as bibliotecas necessárias no contexto da página a avaliar e chama o módulo técnicas;

- Componente servidor: é constituída por dois servidores, um dos quais tem a função de fornecer as bibliotecas à componente cliente (servidor estático) e o outro destina-se à invocação do módulo formatadores, após serem enviados pelo cliente os resultados da avaliação (servidor dinâmico).

3.1.3 Módulo técnicas

Este módulo agrega todas as técnicas WCAG 2.0. Graças à flexibilidade de implementação do módulo, é possível desenvolver e adicionar novas técnicas a qualquer altura, para além de permitir escolher quais as técnicas a executar em cada avaliação.

3.1.4 Módulo formatadores (serialização de dados)

Este módulo é o responsável pela serialização e armazenamento dos resultados da execução do avaliador no formato pretendido. Neste momento apenas existe um formato implementado, o EARL, que é o formato padrão para o armazenamento de asserções de acessibilidade. No entanto, podem ser implementados outros formatos no futuro (por exemplo, pdf ou xls), sem repercussão no código já existente.

3.2 Limitações da ferramenta QualWeb

A ferramenta QualWeb original apresentava algumas limitações que serviram de ponto de partida para os requisitos do trabalho desta tese, limitações essas que se explicitam de seguida.

- Número reduzido de técnicas WCAG implementadas: só foram implementadas técnicas da categoria HTML e, dentro destas, apenas foram desenvolvidas 18 de um total de 60 especificadas.

- O formato EARL, sendo baseado em XML, é de difícil análise por todos os interessados na verificação e correcção dos problemas de acessibilidade.

- A nova abordagem utilizada na avaliação da acessibilidade traz um inconveniente, que é o facto de não ser possível aceder, de um modo directo e fácil, à informação relativa à localização dos problemas de acessibilidade.

3.3 Requisitos

Face às limitações referidas em 3.2., surgiu a necessidade de definir requisitos para as resolver, que se mencionam de seguida:

- Continuação do desenvolvimento de mais técnicas WCAG 2.0 na categoria HTML e início do desenvolvimento de técnicas noutras categorias que se julgassem pertinentes;

- Inclusão de informação adicional relativa à localização de problemas de acessibilidade;

- Desenvolvimento de mecanismos de apresentação de resultados da avaliação.

A implementação destes requisitos será detalhada no próximo capítulo.

3.4 Resumo

Neste capítulo foi descrita a arquitectura do avaliador QualWeb original, mencionadas as suas limitações e enunciados os requisitos deste trabalho.

O próximo capítulo apresenta uma descrição detalhada da implementação destes requisitos.

Capítulo 4

QualWeb-i (QualWeb interactivo)

Este capítulo é o mais importante do relatório, porque é nele que é englobada toda a descrição do trabalho realizado na implementação dos requisitos indicados na secção 3.3., sendo referidas todas as opções tomadas ao longo do desenvolvimento e tecnologias utilizadas.

4.1 Técnicas WCAG 2.0

No âmbito deste projecto, foram desenvolvidas 14 técnicas WCAG 2.0, das quais 7 são de HTML e 7 são de CSS. A escolha das técnicas a implementar foi efectuada tendo em conta que deveriam ser implementadas técnicas com um grau de complexidade superior ao das desenvolvidas no trabalho anterior, sem desprezar o facto de existirem técnicas cuja implementação exigiria bastante tempo de investigação, e que por isso não foram consideradas.

As tabelas 4.1 e 4.2 apresentam, respectivamente, as técnicas HTML e CSS implementadas, mostrando para cada uma a identificação da técnica e a sua descrição.

Técnica	**Descrição**
H24	Fornecer alternativas em texto para os elementos área de mapas de imagens
H27	Fornecer alternativas em texto e em formato não textual para object
H35	Fornecer alternativas em texto em elementos applet
H39	Utilizar elementos caption para associar legendas da tabela de dados a tabelas de dados
H43	Utilizar os atributos id e headers para associar células de dados a células de cabeçalho em tabelas de dados
H63	Utilizar o atributo scope para associar células de cabeçalho a células de dados

	em tabelas de dados
H73	Utilizar o atributo summary do elemento table para fornecer uma descrição geral das tabelas de dados

Tabela 4.1: Técnicas HTML WCAG 2.0 implementadas

Técnica	Descrição
C8	Utilizar letter-spacing CSS para controlar o espaçamento numa palavra
C12	Utilizar percentagem para tamanhos de letra
C13	Utilizar tamanhos de letra identificados
C14	Utilizar unidades 'em' para tamanhos de letra
C19	Especificar o alinhamento à esquerda ou à direita em CSS
C22	Utilizar CSS para controlar a apresentação visual do texto
C28	Especificar o tamanho de caixas de texto utilizando unidades 'em'

Tabela 4.2: Técnicas CSS WCAG 2.0 implementadas

Para garantir a correcta implementação das técnicas foram desenvolvidos e adicionados à bateria de testes unitários já existente pequenos ficheiros de em HTML que exercitam todos os resultados de cada técnica (conformidade, alerta ou erro).

Para além disso, e para servir como extensão futura, foi incluída a referência para o elemento que gerou cada problema na estrutura de resultados criada pelas técnicas. Deste modo, será possível aceder a todos os atributos de todos os elementos e efectuar qualquer tipo de operação sobre eles.

Foram também inseridas em cada técnica as descrições dos vários problemas/alertas/notificações encontrados, sendo todas essas descrições adicionadas à estrutura de resultados e armazenadas num elemento da especificação EARL, a fim de serem apresentadas nas interfaces.

De seguida são referidas as decisões tomadas nas técnicas mais complexas ou com alguma peculiaridade.

4.1.1 Técnicas HTML

H43: Utilizar os atributos id e headers para associar células de dados a células de cabeçalho em tabelas de dados

Dada a complexidade desta técnica, Foi desenvolvida uma versão simplificada, em que são analisadas tabelas só de uma entrada (só um nível de cabeçalho) e tabelas de dupla entrada (só um nível de cabeçalho, mas que abrange a primeira linha de cima e a primeira coluna da esquerda da tabela).

Para as primeiras é pesquisado apenas se um dos ids do atributo headers de cada célula de dados condiz com o id da respectiva coluna, enquanto para as segundas é

pesquisado se pelo menos dois dos ids do atributo headers de cada célula de dados condizem, um com o id da primeira célula do cabeçalho de linha (a contar da esquerda), o outro com a primeira célula de cabeçalho da coluna (a contar de cima).

Esta implementação só funciona com tabelas muito simples e não tem em conta a combinação de outros factores que a seguir se explicitam:

- Tabelas só de uma entrada, mas com o cabeçalho em coluna;

- Tabelas com várias linhas de cabeçalhos, em que seria preciso averiguar se nos ids de cada célula de dados (ou de cabeçalho) estão todos os ids dos cabeçalhos acima (e/ou à esquerda) dessa célula;

- Tabelas com células de cabeçalho ou de dados com o atributo rowspan ou colspan (por exemplo, abaixo de uma célula de cabeçalho cospan = "2" poderão existir duas células de dados, as duas com o mesmo id da célula de cabeçalho.

H63: Utilizar o atributo scope para associar células de cabeçalho a células de dados

Esta técnica só foi implementada tendo em conta os valores row e col do atributo scope. Os valores rowgroup e colgroup não foram considerados por introduzirem mais complexidade na implementação e por haver outras componentes a desenvolver.

4.1.2 Técnicas CSS

Na implementação destas técnicas foi preciso utilizar duas abordagens, consoante o objectivo da técnica.

Se a técnica apenas exigia verificar se determinada propriedade existia, a abordagem a utilizar é mais simples, bastando percorrer os vários elementos HTML no DOM e verificar o estilo de cada um. No caso das técnicas que exigiam verificar, por exemplo, em que unidades estava determinada propriedade, esta abordagem não funciona, sendo necessário realizar o parsing do array de estilos document.styleSheets, e para cada selector existente aplicar a função cssQuery1 que devolve os elementos que satisfazem esse selector.

4.2 Interfaces de visualização de erros

Como referido anteriormente, os conteúdos dos ficheiros EARL não são de fácil compreensão e análise pelas pessoas interessadas na avaliação e correcção de problemas de acessibilidade. Assim, e com o objectivo de responder aos distintos interesses dos

[1] http://dean.edwards.name/my/cssQuery/

diferentes tipos de utilizadores, foram desenvolvidas três interfaces que englobam a informação mais relevante presente nestes relatórios de erros.

4.2.1 Geração da interface

Para aceder facilmente aos dados pretendidos para apresentação na interface houve necessidade de desenvolver um parser para o conteúdo EARL, que foi implementado utilizando a biblioteca jssaxparser1, baseada em eventos SAX (simple API for XML).

Para a implementação do parser poderia ter sido usado uma qualquer biblioteca orientada ao DOM, mas essa alternativa foi excluída à partida porque, dado que este último tipo de bibliotecas constrói uma árvore com todo os elementos XML, o tempo de construção da árvore e os recursos de memória gastos são enormes, tendo em conta que pode haver relatórios de erros bastante extensos. Para além disso, a aproximação SAX mostrou-se adequada para os propósitos que se pretendia, pois apenas era preciso um parsing sequencial, do início ao fim do documento, área em que os parsers baseados em SAX não ficam a perder em relação aos orientados ao DOM.

4.2.2 Conteúdo e descrição das interfaces

As três interfaces apresentam os mesmos dados, mas diferem no modo como estes são apresentados.

Cada interface contém duas tabelas: uma contém os elementos comuns a

toda a avaliação (o número de elementos da página em avaliação e o seu

URL, o modo de teste, a data da avaliação e o nome, a localização e a versão do avaliador); a outra contém os elementos específicos de cada asserção de acessibilidade encontrada (o número da técnica, o URL do W3C com a descrição da técnica, u resultado da técnica e a descrição do resultado). Esta interface foi, na segunda fase do projecto, enriquecida com informação da localização das várias asserções, nomeadamente a linha e o ficheiro onde estas ocorreram, sendo esta informação acrescentada à segunda tabela.

Na primeira fase do trabalho, o avaliador QualWeb era executado no navegador, sendo a interface de visualização dos erros embebida na própria página em avaliação. Já na segunda fase, a execução do avaliador passou para o lado do servidor, sendo a interface separada do conteúdo da página. No entanto, a forma modular como a aplicação está construída possibilitou que o código da interface não sofresse alterações

[1] `http:// code.google.com/p/jssaxparser/`

da primeira para a segunda fase, a menos do enriquecimento supracitado e de alterações simples inerentes à mudança da arquitectura.

Como referido acima, as três interfaces apenas diferem no modo de apresentação dos dados.

Uma delas apresenta os dados na própria página; uma outra apresenta os dados numa iframe e por fim foi desenvolvida uma que apresenta os dados num novo separador/janela do navegador.

O desenvolvimento destas três interfaces oferece aos interessados na acessibilidade um leque mais alargado de escolhas de modos de visualização dos erros, tentando colmatar desvantagens presentes em cada um deles.

Na segunda fase do projecto, foi implementada a possibilidade de alterar facilmente entre os três modos de visualização, sendo apenas necessário escolher o modo pretendido numa caixa combinada e reavaliar a página novamente. Após a reavaliação, a interface anterior antiga é removida e é apresentada a interface desejada.

Esta interface, bem como a percepção da alteração entre interfaces, são completamente acessíveis aos leitores de ecrã. Em particular, foi tido em consideração que, dado o modo como os leitores de ecrã consultam o DOM na busca de alterações à árvore para carregarem eventuais novos dados no buffer de apresentação das páginas, se as alterações forem efectuadas de uma forma muito rápida, o leitor de ecrã não actualiza o buffer, apesar de visualmente os dados aparecerem na página.

Para resolver este problema, foi implementado um tempo de espera entre a remoção dos controlos da interface anterior e a adição dos controlos da interface escolhida posteriormente, possibilitando que os leitores de ecrã actualizem o buffer de modo correcto.

4.2.3 Tecnologias utilizadas

Para além do parser XML SAX referido acima, foi utilizada a linguagem javascript para o desenvolvimento do parser EARL. Na construção das interfaces foram utilizadas as linguagens javascript e HTML e a Framework jquery.

4.3 Localização de erros/alertas/notificações de acessibilidade

4.3.1 Problema

A ferramenta QualWeb adopta uma nova abordagem à avaliação de acessibilidade Web, que recai sobre o conteúdo que é realmente mostrado aos utilizadores e não sobre

o conteúdo das páginas após serem descarregados os ficheiros das mesmas. Esta avaliação baseia-se, assim, no conteúdo das páginas após a execução de código javascript, que pode, entre outras acções, inserir, eliminar, alterar, mostrar e esconder elementos HTML, introduzir, alterar e eliminar conteúdo de folhas de estilo, actualizar as páginas recorrendo a XmlHttpRequest (ajax).

A abordagem utilizada apresenta vários pontos fortes [16], no entanto coloca uma questão que se prende com a dificuldade no fornecimento de informação relevante no que concerne à localização dos problemas de acessibilidade para efeitos de correcção dos mesmos.

Nesta nova abordagem, a execução das técnicas WCAG 2.0 é efectuada sobre o DOM (Document Object Model) das páginas gerado pelo navegador. Do ponto de vista do navegador, a partir do momento em que este analisa todos os ficheiros fonte (HTML, javascript e CSS), extraindo deles toda a informação dos elementos aí presentes para a construção do DOM, deixa de ser relevante a associação entre cada ficheiro e os elementos que este contém ou a localização desses elementos dentro do ficheiro, pois é informação desnecessária à renderização da página que, em última análise, é a função fundamental do DOM. Essa informação perde-se e, após o processamento do navegador, a sua recuperação é impossível no estado actual da tecnologia e das funcionalidades apresentadas pelo DOM.

4.3.2 Alternativas discutidas

Numa fase muito embrionária, foi pensado resolver o problema do lado do cliente.

Na altura da discussão, o avaliador era executado no navegador, tendo como parâmetro de entrada um bookmarklet que injectava o código javascript responsável pela avaliação no DOM da página a avaliar. Em seguida, a avaliação era executada e os resultados eram enviados para um servidor que gerava e guardava o relatório dos erros em formato EARL. A fim de realizar a avaliação, o DOM da página era convertido, através de uma função javascript disponibilizada pelo navegador, numa grande string contendo todos os elementos da página, incluindo a maior parte das mudanças de linha presentes no ficheiro HTML original. Para além disso, o parser HTML utilizado na construção do DOM dispunha de uma funcionalidade que incluía, sem grande esforço, o número da linha em cada elemento HTML. Pensou-se, assim, que uma das abordagens seria a utilização dessa funcionalidade e a consequente modificação da biblioteca EARL de forma a extrair o número de linha de cada elemento e guardá-lo no relatório de erros. No entanto, esta aproximação mostrou-se pouco útil, dado que, para além da ineficácia do parser HTML em fornecer os números de linha dos elementos que eram criados a partir de código javascript sob a forma de inserção na árvore DOM, esses números de

linha diziam respeito a toda a string derivada a partir do DOM e não a cada ficheiro fonte, o que não fazia sentido, visto que os interessados na análise e correcção de erros precisavam de uma localização mais rigorosa dos mesmos, tão perto quanto possível do local que lhes deu origem. Acresce ainda a este inconveniente o facto de muitas páginas não permitirem a injecção de código do lado do navegador, o que impossibilitava a avaliação das mesmas.

Considerou-se, igualmente e ainda segundo a mesma arquitectura (avaliador executado do lado do navegador), a construção de uma extensão para o navegador Firefox orientada ao elemento HTML que, através da contagem do número de elementos presentes na string inferida a partir do DOM e do conhecimento do índice de cada elemento e da apresentação da referida string e da tabela de erros em controlos apropriados, assinalaria nessa string o elemento a que correspondia determinado erro, aquando da selecção desse erro na tabela de erros. Esta alternativa também não se mostrou adequada, pelas mesmas razões mencionadas na alternativa anterior, acrescida do facto de a utilização da ferramenta ficar dependente de um só navegador.

Finalmente, foi considerada uma terceira hipótese que se prendia com a modificação dos ficheiros fonte originais da página (HTML e javascript) em avaliação, introduzindo, em cada elemento HTML, informação sobre o seu número de linha e o ficheiro em que se encontrava. Apesar de, em conjunto com outros motivos de seguida explicitados, implicar uma alteração da arquitectura da ferramenta QualWeb, esta foi a alternativa que pareceu mais adequada e portanto escolhida para implementação. Com efeito, e conforme mencionado na discussão deste problema, a inclusão da informação acima referida nos ficheiros fonte teria de ser realizada antes de qualquer processamento efectuado pelo navegador, porque é a única fase em que é possível aceder a todos os ficheiros sem quaisquer modificações. Daqui decorre que o avaliador não poderia ser executado no navegador, mas sim do lado do servidor, onde existe mais margem de manobra para efectuar as operações pretendidas, sem depender do navegador cliente. As únicas componentes que permaneceriam do lado do cliente seriam a interface com os resultados dos problemas de acessibilidade e uma nova funcionalidade que pediria ao utilizador o URL da página a avaliar e o modo como os resultados seriam apresentados.

4.3.3 Solução

Para a concretização desta tarefa foi utilizada a linguagem nodejs, uma linguagem orientada a eventos de entradas e saídas e escalável para servidores que recebem muitas ligações ao mesmo tempo, e a ferramenta phantomjs1, uma ferramenta de linha de

[1] http://phantomjs.org

comandos que utiliza o motor de renderização WebKit (motor usado em navegadores conhecidos como o Google Chrome e o safari), funcionando como um navegador baseado em WebKit, a menos da interface que não existe, e possibilitando a manipulação do DOM gerado através de javascript, o que serviu os propósitos da funcionalidade desenvolvida.

Para melhor compreensão do todo, segue-se um resumo do fluxo resultante da introdução desta funcionalidade.

Um cliente acede ao servidor onde está alojada a ferramenta QualWeb e, no ecrã que é apresentado, coloca o URL da página a avaliar e o modo como pretende que os resultados sejam apresentados na interface e submete a informação (só o URL é submetido).

Do lado do servidor, recebido o URL do cliente, são, em primeiro lugar, descarregados os URLs de ficheiros javascript referenciados na página a avaliar. Depois, são Descarregados dos ficheiros indicados por esses URLs. Seguidamente, é executado do algoritmo de pré-processamento sobre os ficheiros descarregados, introduzindo, para cada elemento HTML, um atributo contendo o número da linha do ficheiro onde esse elemento se localiza e o nome do ficheiro a que o elemento pertence. Após esta etapa, são modificados os URLs presentes nas tags script para passarem a referenciar os novos ficheiros fonte. Finalmente, é executado o avaliador utilizando os ficheiros modificados pela execução do algoritmo anterior, e é gerado o relatório de erros (formato EARL) que é enviado como resposta ao pedido do cliente. Por seu turno, o cliente executa o parsing do EARL a fim de retirar a informação relevante e mostra-a na interface de apresentação de erros segundo a opção escolhida.

A implementação da funcionalidade de localização de erros é o contributo mais notório e relevante deste trabalho, pelo que os passos para a sua realização serão descritos detalhadamente em seguida.

I) Descarregamento dos URLs de ficheiros javascript referenciados na página a avaliar

Para a concretização deste passo foram consideradas duas alternativas.

Uma primeira hipótese era descarregar o ficheiro HTML e fazer o seu parsing, pesquisando todos os URLs para ficheiros javascript. Esta hipótese não foi seguida, devido à dificuldade em encontrar todos

Os URLs, que podem estar presentes em várias zonas do ficheiro e sob diferentes formas. Para além de poderem ser encontrados nos atributos src das tags javascript, podem também existir em código javascript, quer na criação de scripts (funções de inserção de elementos na árvore DOM), quer em funções que imprimem código

javascript, para além de as strings de URL resultantes poderem ser geradas pela concatenação de literais com variáveis e concatenação de variáveis.

Rapidamente foi descoberta uma alternativa mais simples e com resultados mais rigorosos, que foi a extracção dos URLs a partir dos atributos src dos elementos script presentes no DOM da página gerado pelo phantomjs. Deste modo, foi aproveitada a funcionalidade de parsing existente no DOM para descarregar todos os URLs, com a garantia de não haver URLs não processados.

Assim, para a elaboração desta funcionalidade é executado o phantomjs tendo como parâmetro o URL da página a avaliar. De seguida acede-se ao DOM gerado e, para cada elemento script aí presente extrai-se o valor do seu atributo src.

II) Descarregamento dos ficheiros indicados pelos URLs descarregados

Neste passo, procedeu-se à preparação e descarregamento de todos os ficheiros HTML e javascript.

Na preparação do descarregamento, foi implementada uma estrutura de pares (chave, valor), em que a chave é o URL e o valor é o nome que é dado ao ficheiro aquando do seu armazenamento após a execução do algoritmo de pré-processamento.

O primeiro par é constituído por (URL da página inicial, file_0.HTML).

Os pares seguintes contêm nas chaves os URLs para os ficheiros de extensão js e nos valores os respectivos futuros nomes de ficheiros correspondentes a esses URLs: (URL_1, file_1.js), (URL_2, file_2.js),...(URL_n, file_n.js), em que URL_1, URL_2, ..., URL_n são os nomes reais dos URLs dos ficheiros javascript e file_1.js, file_2.js,...,file_n.js são os nomes escolhidos para os ficheiros javascript que posteriormente serão guardados temporariamente na máquina onde corre o QualWeb.

Esta estrutura de pares (chave, valor) evita que um ficheiro seja descarregado mais do que uma vez, no caso de um mesmo URL ter sido descarregado mais do que uma vez no passo anterior (esta possibilidade pode ser real no caso de sites mais complexos ou que contenham erros de estrutura e indiquem um ficheiro mais do que uma vez).

Após a construção da estrutura, esta é percorrida e cada ficheiro correspondente ao URL é descarregado.

Numa implementação inicial, cada ficheiro era guardado no disco duas vezes: antes e após a execução do algoritmo de pré-processamento. No entanto, percebeu-se que bastava guardar o ficheiro apenas uma vez, após o algoritmo de pré-processamento, evitando uma escrita e uma leitura desnecessárias por cada ficheiro. Assim, as strings

contendo o conteúdo dos ficheiros são descarregadas utilizando a ferramenta curl1 e passadas directamente ao algoritmo de pré-processamento.

Relativamente à estrutura temporária de armazenamento dos ficheiros, havia duas hipóteses: ou a estrutura de directórios com base nos URLs dos ficheiros era replicada e os ficheiros poderiam manter o mesmo nome que tinham originalmente, ou os ficheiros eram todos descarregados para a mesma directoria. Optou-se pela segunda hipótese para simplicidade de posterior remoção dos ficheiros. No entanto, esta escolha obrigou a que fossem dados nomes únicos aos ficheiros.

Para além disso, de forma a possibilitar várias avaliações em paralelo, e utilizando-se a capacidade inerente da linguagem do lado do servidor em manter estado entre pedidos HTTP, é criado, para cada pedido do cliente, uma directoria diferente para cada avaliação.

III) Execução do algoritmo de pré-processamento sobre os ficheiros descarregados

O algoritmo de pré-processamento é o responsável pela introdução, no ficheiro HTML e nos ficheiros javascript, de informação respeitante à localização dos problemas de acessibilidade.

Em particular, para cada elemento de cada ficheiro, é adicionada informação que contempla a linha desse ficheiro onde o elemento se localiza, bem como o nome do ficheiro onde o elemento se encontra.

Para a concretização do algoritmo foram consideradas 3 alternativas descritas seguidamente.

Uma primeira alternativa passava pela inclusão de um atributo class em cada elemento, da forma class = "line_NUMERODALINHA file_NOMEDOFICHEIRO", em que NUMERODALINHA e NOMEDOFICHEIRO são substituídos, respectivamente, pelo número da linha e pelo nome do ficheiro. Exemplo: class = "line_4 file_funcoes.js".

Esta alternativa apresenta um inconveniente, que é evidenciado quando já existia um atributo class no mesmo elemento antes da inclusão do novo atributo class, em que, na presença de dois atributos class, apenas um deles é incluído no DOM e, perante esta situação, ou é incluído o elemento que já existia e perde-se a informação da localização do elemento, ou é incluído o atributo com a informação do elemento e perde-se informação importante da página original.

[1] http://curl.haxx.se

Outra alternativa passava pela inserção de um elemento não modificador da apresentação e funcionalidade da página, por exemplo, um span, antes do elemento em questão, da forma

line_NUMERODALINHA file_NOMEDOFICHEIRO<TAG></TAG>, em que NUMERODALINHA, NOMEDOFICHEIRO e TAG são substituídos, respectivamente, pelo número da linha, pelo nome do ficheiro e pela tag do elemento

considerado. Exemplo:

line_4 file_funcoes.js> .

Esta alternativa também não se mostrou a mais apropriada, dado que a introdução de elementos span duplicaria o número de elementos no DOM, o que aumentaria o tempo de avaliação das páginas, sendo este um processo impraticável caso se tratassem de páginas muito complexas.

A alternativa considerada para implementação foi a inserção de um atributo único em cada elemento HTML, a fim de se conseguir, por um lado, não aumentar a complexidade do processamento da avaliação, e por outro lado que fossem carregados no DOM todos os atributos da página original e o novo atributo, evitando a possível perda de informação.

Nesta situação, o nome de atributo que se mostrou mais adequado e que melhor cumpre os requisitos indicados acima foi um atributo presente no HTML5, que é composto por uma string fixa concatenada com uma string de escolha arbitrária: o atributo data-STRINGARBITRARIA, em que data- é a string fixa e STRINGARBITRARIA é uma string escolhida pelo programador. Neste caso, julgou-se feliz a introdução do atributo data-qualweb, sob a seguinte forma:

data-qualweb = "line_NUMERODALINHA file_NOMEDOFICHEIRO", em que NUMERODALINHA e NOMEDOFICHEIRO são substituídos, respectivamente, pelo número da linha e pelo nome do ficheiro. Exemplo, um atributo p na linha 4 do ficheiro funcoes.js ficaria:

<p a1 = "valor" a2 = "valor" ... an = "valor" data-qualweb = "line_4 file_funcoes.js">TEXTO em que a1, a2,...,an são eventuais outros atributos de p.

Na implementação deste algoritmo foram utilizadas expressões regulares, considerando elementos HTML nas seguintes condições:

- Elementos HTML em strings passadas como argumentos da função document.write e em strings presentes na atribuição à propriedade elemento.innerHTML;

- Elementos HTML presentes em código javascript responsável pela criação e inserção de elementos na árvore DOM. Em particular, para cada linha de cada ficheiro, foram pesquisados os locais onde existiam funções appendChild e insertBefore (funções de inserção de elementos no DOM) e, após a chamada de cada uma delas, foi introduzido código javascript (função setAttribute), que coloca o atributo data-qualweb nesses elementos.

- Todos os outros elementos HTML, nomeadamente aqueles que não estão embebidos em código javascript.

Foram tidos alguns cuidados especiais no desenvolvimento do algoritmo, em particular foram colocadas barras de escape de aspas nas strings javascript com atributos HTML e foi tida em consideração a possibilidade da existência de vários scripts numa única linha de um ficheiro, sendo os atributos correctamente adicionados.

No entanto, o algoritmo apresenta alguns pontos fracos, não de muito fácil resolução, que são a incapacidade em lidar com atributos presentes em strings que resultem da concatenação de literais com variáveis (nestes casos não há modificação) e a impossibilidade de lidar com atributos introduzidos por funções presentes em Frameworks javascript (jquery, Ext JS, MooTools, etc.).

IV) Alteração dos URLs presentes nos ficheiros fonte que referenciam ficheiros javascript

Para garantir que os ficheiros alvo do algoritmo de pré-processamento utilizados na avaliação são os presentes na máquina onde corre o QualWeb, foi necessário alterar os URLs que referenciavam estes ficheiros para passarem a referenciar os ficheiros guardados no local temporário.

Para isso, o algoritmo pesquisa, em todos os ficheiros (html e js) cada atributo src existente em elementos script, quer presente directamente no HTML, quer presente em código javascript de criação de scripts (atribuição de valor à propriedade elemento.src), e substitui o URL original pelo URL novo.

O URL novo é determinado em dois passos. Primeiro, é utilizada uma função de resolução de URLs que, dado o URL da página inicial e o URL antigo de um atributo src, devolve o URL absoluto para o ficheiro em questão. Em seguida, é efectuado um lookup na tabela de URLs, passando como argumento o URL obtido no passo anterior e recebendo como valor o URL a colocar no atributo src do ficheiro novo.

Deste modo, a avaliação pode ser executada sem problemas e o DOM pode ser gerado recorrendo aos ficheiros locais.

V) Execução do avaliador utilizando os ficheiros modificados

Após a execução do algoritmo de pré-processamento sobre todos os ficheiros fonte, os ficheiros resultantes são alvo de avaliação. Para tal, é executado o phantomjs uma segunda vez e executadas as técnicas sobre o DOM gerado. Houve, assim, a necessidade de alterar a estrutura de resultados gerada pelas técnicas, de forma a conter a informação da linha e nome de ficheiro.

VI) Geração do relatório de erros e envio deste para o cliente

A biblioteca de geração do EARL é utilizada para gerar a string EARL contendo os erros, alertas e conformidades de acessibilidade. Nesta fase, foi necessário modificar esta biblioteca, de forma a conter a nova informação introduzida.

Para cada asserção, o número da linha é guardado num elemento ptr:lineNumber e o nome do ficheiro é guardado num elemento earl:info. O nome deste segundo elemento não está relacionado com o nome do ficheiro, no entanto, devido à ausência da especificação do EARL de um elemento específico para guardar nomes de ficheiros, este foi um dos elementos que ainda não continha dados e que se julgou adequado para colocar esta informação.

A resposta ao cliente é uma string que se encontra dividida nas seguintes componentes, com o objectivo de facilitar o parsing do lado do cliente:

Conteúdo EARL<<SNF_ERRORS>>URLs correspondentes a ficheiros que estão referenciados em atributos src de elementos script mas não foram encontrados (separados pelo caracter de mudança de linha \n)<<OTHER_ERRORS>>Erros relacionados com a não localização da página ou do alojamento

Os elementos rodeados de << e >> são os separadores das várias componentes e estão sempre presentes na string.

A string é enviada para o cliente que faz o parsing do EARL, extrai os elementos importantes e apresenta-os na interface de visualização de erros. Caso haja algum problema no processamento da avaliação, será apresentado na interface o motivo que levou à existência do mesmo.

4.4 Resumo

Neste capítulo foi apresentado todo o trabalho efectuado, as opções tomadas e a sua justificação, os aspectos menos positivos das várias componentes e as tecnologias utilizadas na implementação.

O próximo capítulo visa validar a implementação realizada e mostrar os vários modos de visualização na avaliação da acessibilidade de alguns sites mais populares.

Capítulo 5

Validação

Este capítulo serve dois grandes objectivos. Por um lado, pretende-se mostrar as interfaces de visualização dos resultados de acessibilidade e provar o seu correcto funcionamento, recorrendo à avaliação de acessibilidade de alguns sítios Web mais utilizados. Por outro lado, pretende-se provar a correcta implementação do algoritmo de pré-processamento nos aspectos relativos à inserção de informação de localização de problemas e à modificação de URLs.

5.1 Apresentação das Interfaces de visualização dos resultados

Para apresentar as interfaces optou-se por escolher, da página Alexa Top 500[1], dois dos sítios Web mais populares (www.wikipedia.org e www.yahoo.com) e executar o avaliador sobre eles. As figuras seguintes mostram, para cada um dos sítios Web, as três formas de visualização de problemas de acessibilidade.

Todas as formas de visualização são formadas por duas tabelas, uma abaixo da outra.

A tabela de cima contém os elementos comuns a toda a avaliação e é constituída por duas colunas. Na coluna da esquerda estão as propriedades e na coluna da direita os respectivos valores.

As propriedades presentes nesta tabela são: número de elementos da página em avaliação, URL da página em avaliação, modo de teste (toma o valor "automático" em todas as avaliações),

data da avaliação, nome do avaliador (The QualWeb WCAG 2.0 evaluator), localização do avaliador (http://qualweb.di.fc.ul.pt/) e versão do avaliador (0.1).

[1] HTTP://WWW.ALEXA.COM/TOPSITES

A tabela de baixo contém os elementos específicos de cada asserção de acessibilidade e é constituída por seis colunas, cada uma correspondendo, da esquerda para a direita, às seguintes propriedades: número da técnica, URL do W3C com a descrição da técnica, resultado da técnica, descrição do resultado, linha do ficheiro onde ocorreu a asserção e nome do ficheiro onde ocorreu a asserção.

A diferença entre os três modos de visualização reside na forma como as tabelas são apresentadas.

Nas figuras que apresentam o modo de visualização, quer na mesma página, quer numa iframe, pode observar-se, acima das tabelas, um formulário do qual fazem parte:

- Uma caixa de texto preenchida com o URL da página previamente avaliada;

- Uma caixa combinada onde está seleccionado o modo de visualização de erros escolhido;

- Um botão que dá início à avaliação da página.

No modo de visualização na mesma página, as tabelas são apresentadas na mesma frame do formulário, enquanto no modo de visualização numa iframe as tabelas são apresentadas na mesma janela do formulário, mas numa iframe.

Nas figuras que apresentam o modo de visualização num outro separador, o mesmo é preenchido apenas com as duas tabelas, ficando o formulário no separador antigo, e portanto não visível nas figuras que apresentam esta situação.

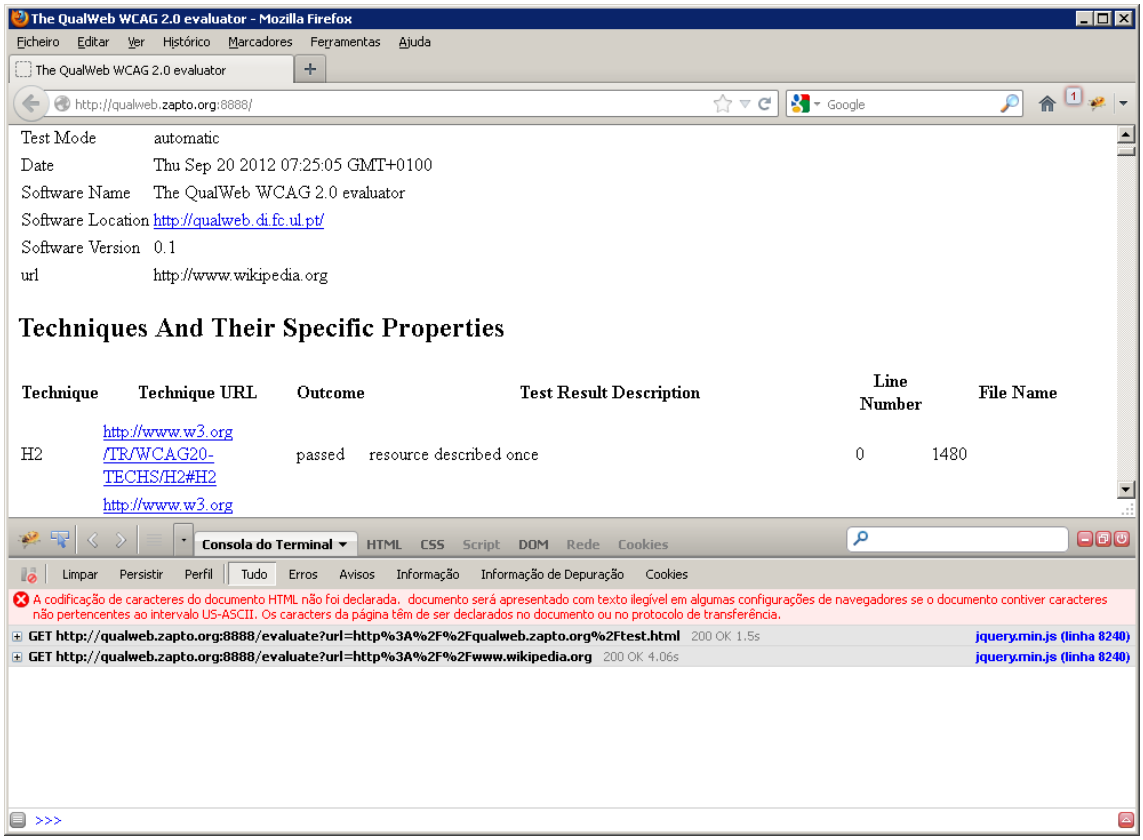

Figura 5.1: wikipedia.org - interface na mesma página

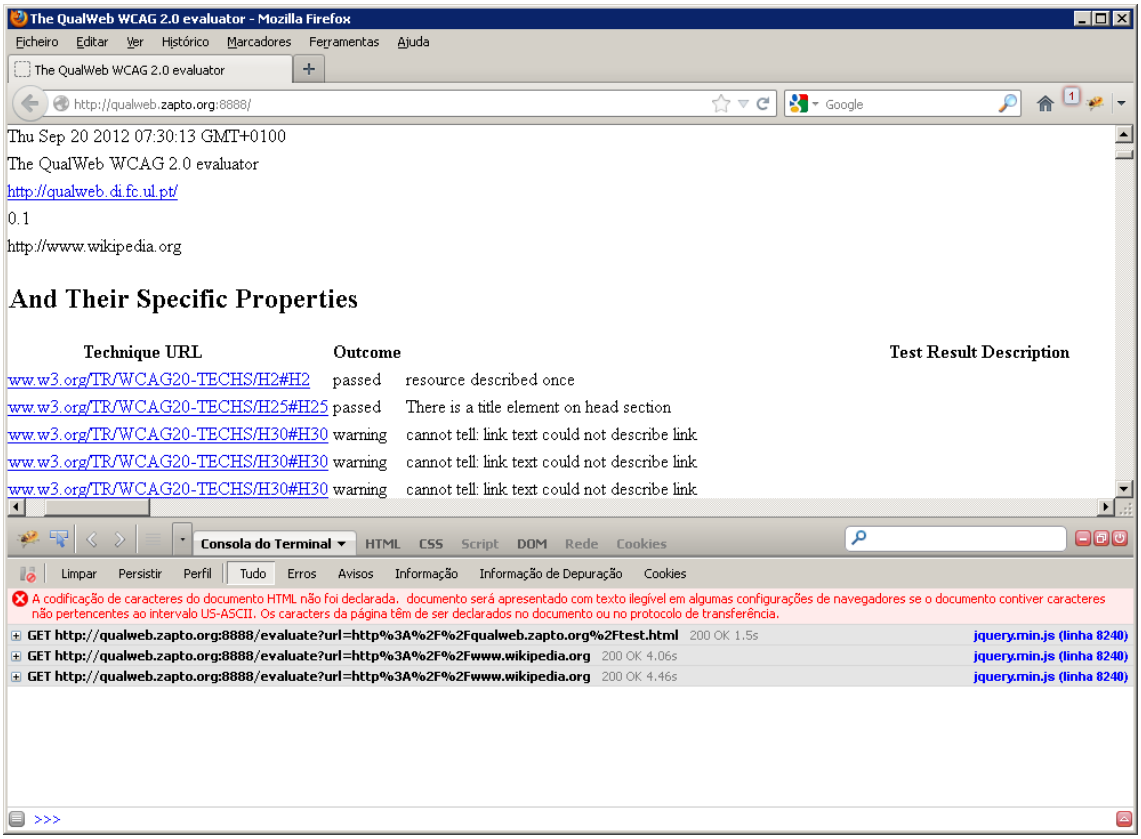

Figura 5.2: wikipedia.org - interface numa iframe

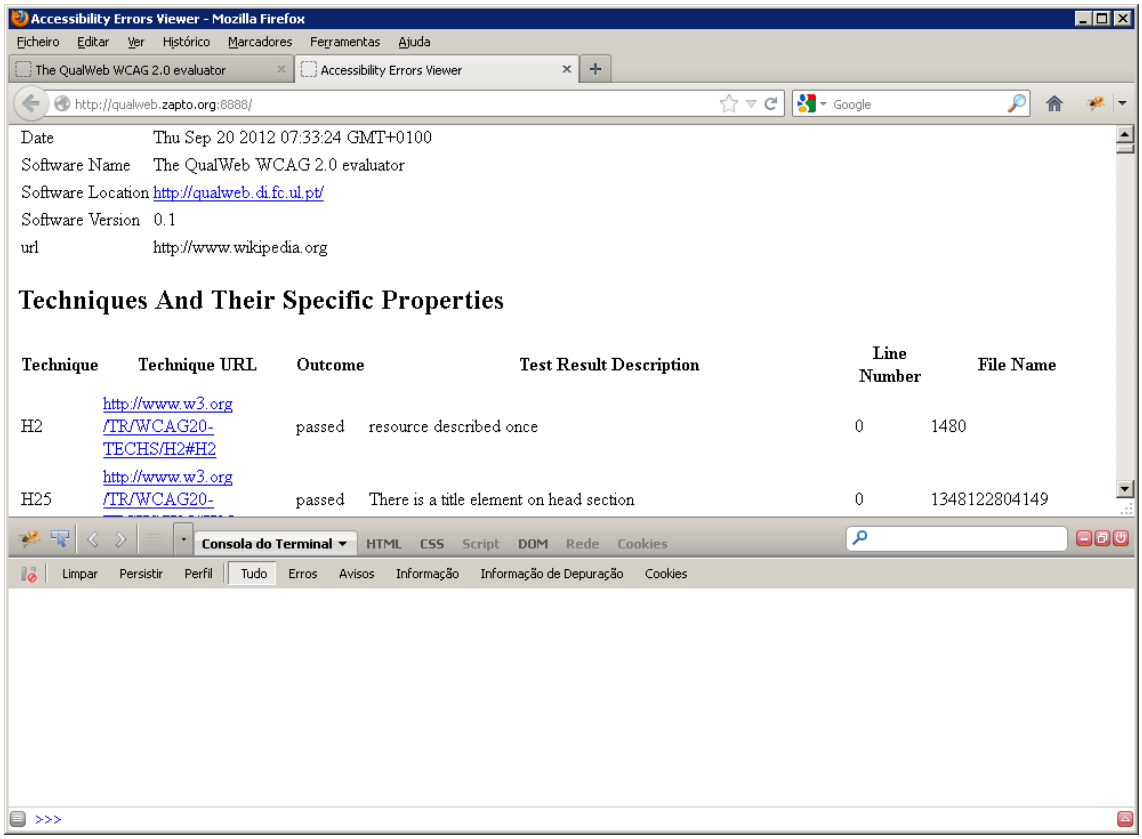

Figura 5.3: wikipedia.org - interface num outro separador

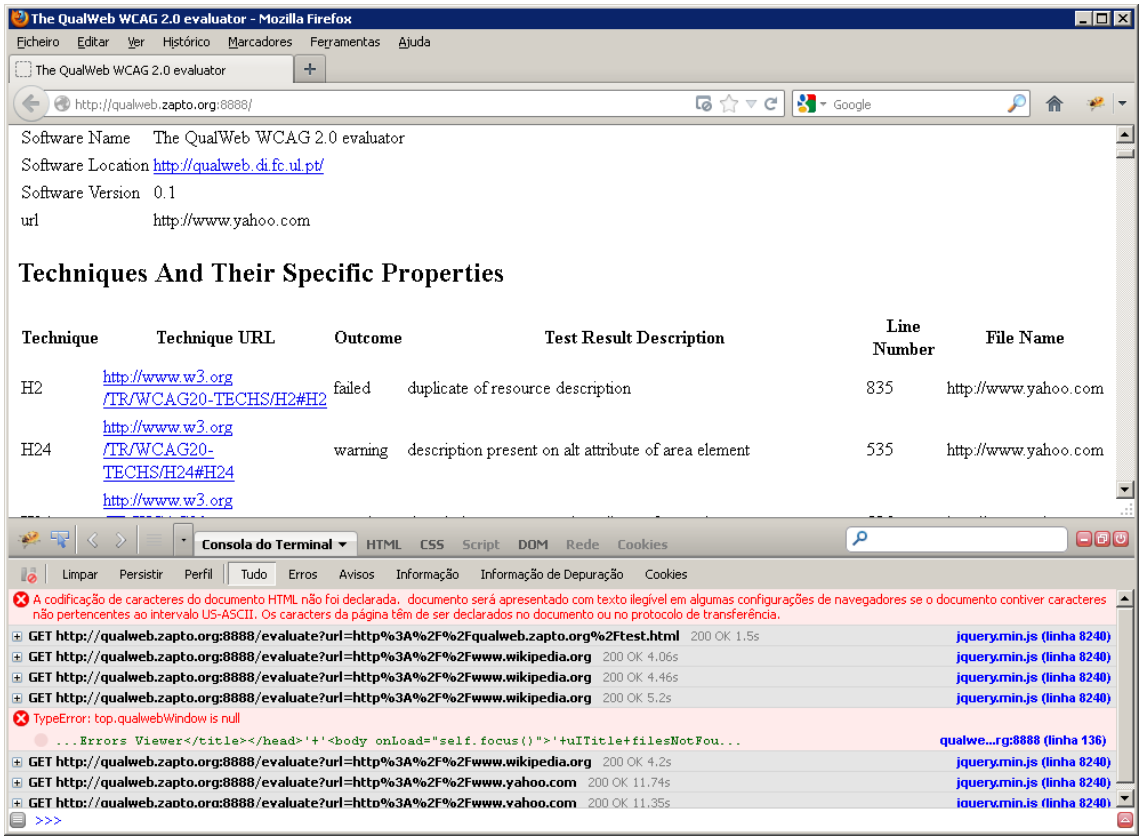

Figura 5.4: yahoo.com - interface na mesma página

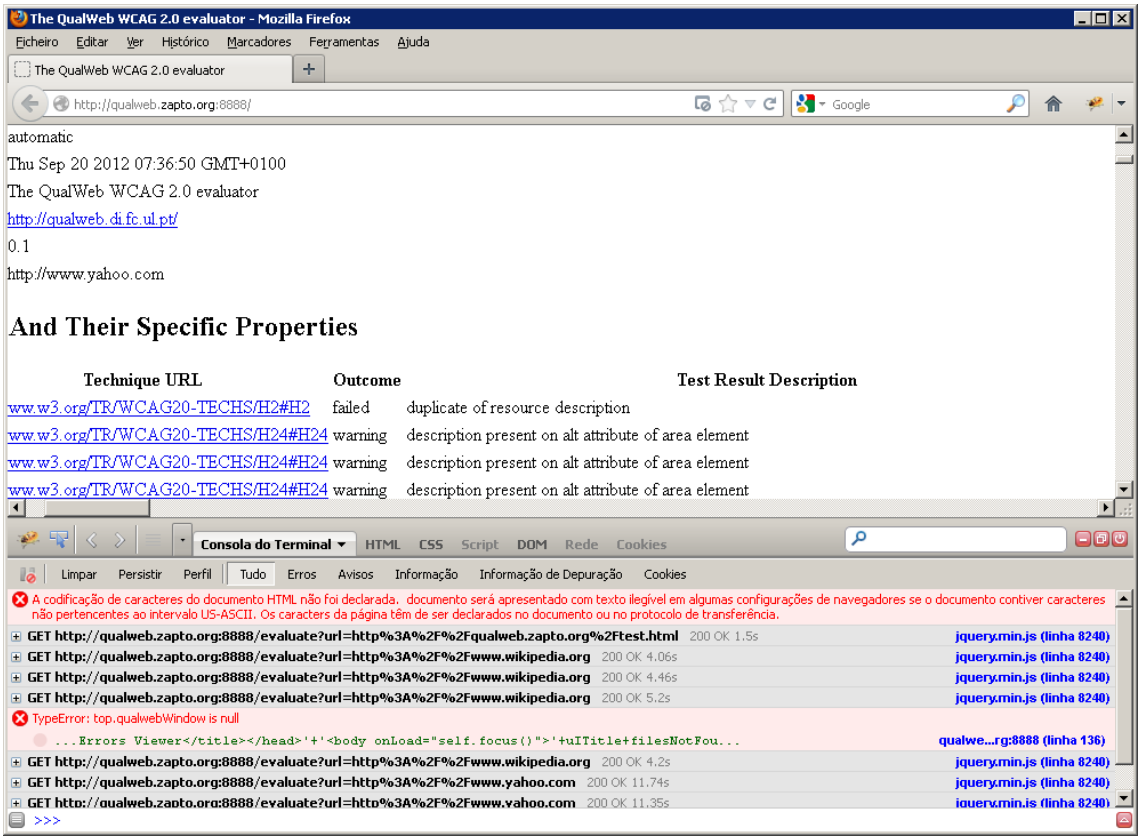

Figura 5.5: yahoo.com - interface numa iframe

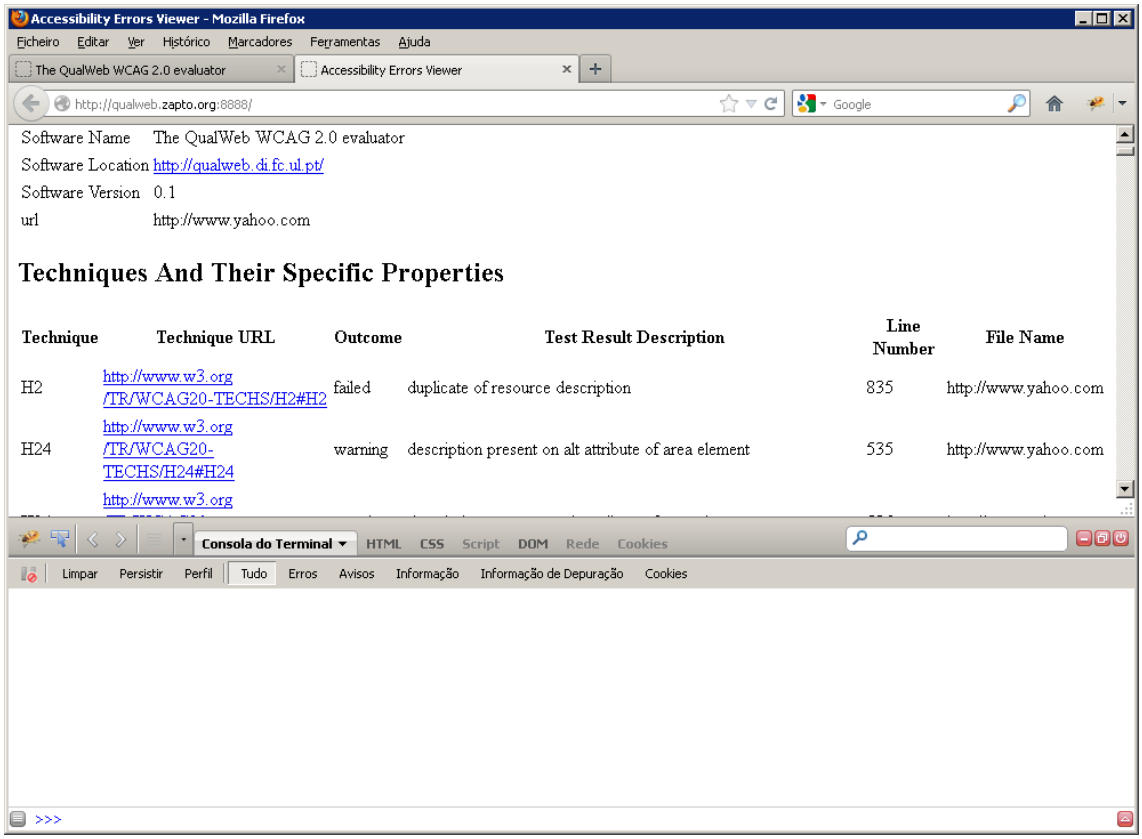

Figura 5.6: yahoo.com - interface num outro separador

5.2 Correcção do algoritmo de pré-processamento

Com o objectivo de verificar a correcção do algoritmo de pré-processamento foi implementado um pequeno teste com a criação de dois ficheiros:

- Ficheiro test.html: contém um elemento <script> com inclusão do ficheiro dir/file.js, alguns elementos HTML e código javascript que cria outros elementos HTML

- Ficheiro dir/file.js: é incluído pelo ficheiro test.html e insere alguns elementos no DOM.

Foi executado o avaliador sobre os ficheiros originais e em seguida foram recolhidos os ficheiros temporários modificados após o algoritmo de pré-processamento.

Segue-se o conteúdo dos ficheiros. Devido aos limites e quebras de linha impostos pelo editor, cada linha foi anotada com o seu número. Cada linha que não temm número é a continuação da linha anterior.

```
1. <html>
2. <head>
3. <title>page</title>
4. </head>
5. <body>
6. olá!
7. <img src="./gato.jpg" alt = "" />
8. <div id = "mydiv">
9. </div>
10. <script type="text/javascript">
11. var el = document.getElementById("mydiv");
12. el.innerHTML="<p id = \"par\">primeiro paragrafo</p>";
13. var link1 = document.createElement("a");
14. link1.setAttribute("href","http://www.google.com");
15. link1.appendChild(document.createTextNode("google"));
16. var prim_par = document.getElementById("par");
17. prim_par.appendChild(link1);
18.
19. var link2 = document.createElement("a");
20. link2.setAttribute("href","http://www.sapo.pt");
21. link2.appendChild(document.createTextNode("sapo"));
22. link2.setAttribute("class","ab cd");
23.
24. document.body.insertBefore(link2,el);
25. </script>
26. <script type="text/javascript" src="dir/file.js"></script>
27. </body>
28. </html>
```

Figura 5.7: Ficheiro test.html original

```
1. var paragrafo = document.getElementById("par");
2. paragrafo.innerHTML += "<span><img src=\"./gato3.jpg\" alt = \"um gato\" /></span>";
```

Figura 5.8: Ficheiro dir/file.js original

```
1. <html>
2. <head>
3. <title data-qualweb = "line_3 file_http://qualweb.zapto.org/test.html">page</title>
4. </head>
5. <body data-qualweb = "line_5 file_http://qualweb.zapto.org/test.html">
6. ola!
7. <img src="./gato.jpg" alt = "" data-qualweb = "line_7 file_http://qualweb.zapto.org/test.html" />
8. <div id = "mydiv" data-qualweb = "line_8 file_http://qualweb.zapto.org/test.html">
9. </div>
10. <script type="text/javascript" data-qualweb = "line_10 file_http://qualweb.zapto.org/test.html">
11. var el = document.getElementById("mydiv");
12. el.innerHTML="<p id = \"par\" data-qualweb = \"line_12 file_http://qualweb.zapto.org/test.html\">primeiro paragrafo</p>";
13. var link1 = document.createElement("a");
14. link1.setAttribute("href","http://www.google.com");
15. link1.appendChild(document.createTextNode("google"));
16. var prim_par = document.getElementById("par");
17. prim_par.appendChild(link1);link1.setAttribute("data-qualweb","line_17 file_http://qualweb.zapto.org/test.html");
18.
19. var link2 = document.createElement("a");
```

```
20. link2.setAttribute("href","http://www.sapo.pt");
21. link2.appendChild(document.createTextNode("sapo"));
22. link2.setAttribute("class","ab cd");
23.
24. document.body.insertBefore(link2,el);link2.setAttribute("data-qualweb","line_24 file_http://qualweb.zapto.org/test.html");
25. </script>
26. <script type="text/javascript" src = "file_1.js" data-qualweb = "line_26 file_http://qualweb.zapto.org/test.html"></script>
27. </body>
28. </html>
```

Figura 5.9: Ficheiro test.html modificado (nome temporário file_0.html)

```
1. var paragrafo = document.getElementById("par");
2. paragrafo.innerHTML += "<span data-qualweb = \"line_2 file_http://qualweb.zapto.org/dir/file.js\"><img src=\"./gato3.jpg\" alt = \"um gato\" data-qualweb = \"line_2 file_http://qualweb.zapto.org/dir/file.js\" /></span>";
```

Figura 5.10: Ficheiro dir/functions.js modificado (nome temporário file_1.js)

Pela análise do conteúdo dos ficheiros, pode constatar-se que o algoritmo de pré-processamento funciona correctamente, quer na funcionalidade de inserção de números de linha e de nomes de ficheiro, quer na funcionalidade de alteração de URLs.

Para servir como reforço à validação da correcção deste algoritmo, encontram-se no anexo A excertos de código de ficheiros retirados das páginas Web National Federation of the Blind (www.nfb.org) e World Wide Web Consortium (www.w3.org). Foram escolhidas estas páginas por apresentarem uma complexidade moderada no que respeita à programação javascript. Deste modo, consegue-se provar que, a menos das limitações já indicadas no capítulo anterior, o algoritmo de pré-processamento funciona correctamente.

5.3 Resumo

Este capítulo divisou apresentar as interfaces de visualização de resultados de avaliação de acessibilidade sobre dois dos sites mais populares (escolhidos através do site Alexa top 500) e mostrar que o algoritmo de pré-processamento produz resultados fiáveis. O próximo capítulo encerra este relatório, relevando os aspectos menos positivos do projecto e mencionando os pontos fundamentais a desenvolver num trabalho futuro.

Capítulo 6

Conclusão

Os desafios sentidos ao longo deste projecto foram maiores (e consequentemente mais fascinantes) na fase do desenho das funcionalidades do que na implementação propriamente dita. Em particular, aquele que se mostrou mais difícil foi a definição do modo como se deveria implementar o algoritmo de inclusão de informação extra, em que foi preciso algum tempo de debate.

Os requisitos definidos para este trabalho foram todos cumpridos, apesar de ainda haver espaço para melhoramentos.

Foram implementadas sete técnicas HTML e sete técnicas CSS. Algumas técnicas HTML mais complexas foram reduzidas a versões simplificadas. Paralelamente, foi estendida a bateria de testes das técnicas com a criação de ficheiros de teste para as técnicas implementadas.

Procedeu-se ao desenvolvimento de três interfaces para visualização dos resultados da avaliação, facilitando a tarefa dos utilizadores finais, organismos públicos e programadores. Para além disso, as interfaces foram enriquecidas com informação extra de localização dos problemas de acessibilidade, tendo sido envidados esforços para que a informação fornecida seja correcta, útil e tão próxima quanto possível do local onde os problemas ocorrem.

A utilização da aplicação phantomjs na execução do avaliador permitiu que as avaliações fossem efectuadas independentemente do navegador do utilizador.

6.1 Trabalho Futuro

Apesar dos esforços desenvolvidos para que a extensão do avaliador QualWeb ficasse perfeita, isso não foi possível, pelo que se apresentam vários aspectos que poderão ser melhorados no futuro:

- Melhoramento do algoritmo de pré-processamento, para que:

i) seja adicionada informação aos elementos criados a partir da concatenação de literais com variáveis ou de variáveis com variáveis;

II) Seja implementado de forma mais genérica a fim de considerar elementos criados por Frameworks javascript.

- Melhoramento do algoritmo de modificação de URLs para considerar URLs que provenham de concatenações;

- Introdução de informação de localização de problemas relacionada com folhas de estilo;

- Extensão do algoritmo de pré-processamento para ficheiros do lado do servidor (php, asp, etc.);

- Desenvolvimento de mais técnicas WCAG 2.0, tanto em categorias já exploradas como em categorias ainda por iniciar (scripting e Accessible Rich Internet Applications);

- Desenvolvimento de trabalho nas Rich Internet Applications, no sentido de aferir se estas implementam a especificação Accessible Rich Internet Applications [21] correctamente, nomeadamente na acessibilidade que conferem às regiões dinâmicas das páginas Web e aos controlos e modos de navegação que estas aplicações introduzem.

- Correcção automática de erros de acessibilidade.

Anexo A – Excertos de código

Deste anexo consta o código dos ficheiros html e javascript das páginas National Federation of the Blind (www.nfb.org) e World Wide Web Consortium (www.w3.org).

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
2.  "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
3.  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" version="XHTML+RDFa 1.0" dir="ltr">
4.
5. <head profile="http://www.w3.org/1999/xhtml/vocab">
6.  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
7.  <link rel="alternate" type="application/rss+xml" title="NFB RSS" href="http://www.nfb.org/rss.xml" />
8. <meta name="Generator" content="Drupal 7 (http://drupal.org)" />
9.  <title>NFB</title>
10.  <style type="text/css" media="all">@import url("http://www.nfb.org/modules/system/system.base.css?ma6tf1");
...
33.  <script type="text/javascript" src="http://www.nfb.org/misc/drupal.js?ma6tf1"></script>
...
48. </head>
49. <body class="html front not-logged-in one-sidebar sidebar-first page-node" >
50.  <div id="skip-link">
...
```

Figura A.1: site nfb.org: excerto do ficheiro /index.html original

```
...
282.  Drupal.unfreezeHeight();
283.  $('<div id="freeze-height"></div>').css({
284.    position: 'absolute',
...
358.    // Make the responseText more readable by stripping HTML tags and newlines.
359.  responseText = responseText.replace(/<("[^"]*"|'[^']*'|[^'">])*>/gi,"");
360.  responseText = responseText.replace(/[\n]+\s+/g,"\n");
...
382.  if (jQuery.support.positionFixed === undefined) {
383.                var el = $('<div style="position:fixed; top:10px" />').appendTo(document.body);
384.    jQuery.support.positionFixed = el[0].offsetTop === 10;
...
407.  placeholder: function (str) {
408.    return '<em class="placeholder">' + Drupal.checkPlain(str) + '</em>';
409.  }
...
```

Figura A.2: site nfb.org: excerto do ficheiro /misc/drupal.js?ma6tf1 original

1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"

2. "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">

3. <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" version="XHTML+RDFa 1.0" dir="ltr">

4.

5. <head profile="http://www.w3.org/1999/xhtml/vocab">

6. <meta http-equiv="Content-Type" content="text/html; charset=utf-8" data-qualweb = "line_6 file_http://www.nfb.org/" />

7. <link rel="alternate" type="application/rss+xml" title="NFB RSS" href="http://www.nfb.org/rss.xml" data-qualweb = "line_7 file_http://www.nfb.org/" />

8. <meta name="Generator" content="Drupal 7 (http://drupal.org)" data-qualweb = "line_8 file_http://www.nfb.org/" />

9. <title data-qualweb = "line_9 file_http://www.nfb.org/">NFB</title>

10. <style type="text/css" media="all" data-qualweb = "line_10 file_http://www.nfb.org/">@import url("http://www.nfb.org/modules/system/system.base.css?ma6tf1");

...

33. <script type="text/javascript" src = "file_4.js" data-qualweb = "line_33 file_http://www.nfb.org"></script>

...

48. </head>

49. <body class="html front not-logged-in one-sidebar sidebar-first page-node" data-qualweb = "line_49 file_http://www.nfb.org/" >

50.

...

Figura A.3: site nfb.org: excerto do ficheiro /index.html modificado (nome temporário file_0.html)

...

282. Drupal.unfreezeHeight();

283. $('<div id="freeze-height" data-qualweb = \"line_283 file_http://www.nfb.org/misc/drupal.js?ma6tf1\"> ').css({

284. position: 'absolute',

...

358. // Make the responseText more readable by stripping HTML tags and newlines.

359. responseText = responseText.replace(/<("[^"]*"|'[^']*'|[^'" data-qualweb = \"line_359 file_http://www.nfb.org/misc/drupal.js?ma6tf1\">])*>/gi,"");

360. responseText = responseText.replace(/[\n]+\s+/g,"\n");

...

382. if (jQuery.support.positionFixed === undefined) {

383. var el = $('<div style="position:fixed; top:10px" data-qualweb = \"line_383 file_http://www.nfb.org/misc/drupal.js?ma6tf1\" />').appendTo(document.body);

384. jQuery.support.positionFixed = el[0].offsetTop === 10;

...

407. placeholder: function (str) {

408. return '<em class="placeholder" data-qualweb = \"line_408 file_http://www.nfb.org/misc/drupal.js?ma6tf1\">' + Drupal.checkPlain(str) + '';

409. }

...

Figura A.4: site nfb.org: excerto do ficheiro /misc/drupal.js?ma6tf1 modificado (nome temporário file_4.js)

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
2. <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3. <!-- Generated from data/head-home.php, ../../smarty/{head.tpl} -->
4. <head>
5. <title>World Wide Web Consortium (W3C)</title>
6. <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
7. <link rel="Help" href="/Help/" />
8. <link rel="stylesheet" href="/2008/site/css/minimum" type="text/css" media="handheld, all" />
9. <style type="text/css" media="print, screen and (min-width: 481px)">
...
21. <body id="www-w3-org" class="w3c_public w3c_home">
...
464. <!-- Generated from data/scripts.php, ../../smarty/{scripts.tpl} -->
465. <div id="w3c_scripts"><script type="text/javascript" src="/2008/site/js/main">
466. //<![CDATA[
467. <!-- -->
468. //]]>
469. </script></div>
470. </body>
471. </html>
```

Figura A.5: site w3.org: excerto do ficheiro /index.html original

```
...
222. }else{BH.text=BI
223. }BJ.insertBefore(BH,BJ.firstChild);
224. BJ.removeChild(BH)
...
311. W.style.display="none";
312. W.innerHTML=" <link/><table></table><a href='/a' style='color:red;float:left;opacity:.55;'>a</a><input type='checkbox'/>";
313. var U=W.getElementsByTagName("*"),T=W.getElementsByTagName("a")[0],P=o.createElement("select"),S=P.appendChild(o.createElement("option"));
...
```

Figura A.6: site w3.org: excerto do ficheiro /2008/site/js/main original

1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" data-qualweb = "line_1 file_http://www.w3.org">

2. <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

3. <!-- Generated from data/head-home.php, ../../smarty/{head.tpl} -->

4. <head>

5. <title data-qualweb = "line_5 file_http://www.w3.org">World Wide Web Consortium (W3C)</title>

6. <meta http-equiv="Content-Type" content="text/html; charset=utf-8" data-qualweb = "line_6 file_http://www.w3.org" />

7. <link rel="Help" href="/Help/" data-qualweb = "line_7 file_http://www.w3.org" />

8. <link rel="stylesheet" href="/2008/site/css/minimum" type="text/css" media="handheld, all" data-qualweb = "line_8 file_http://www.w3.org" />

9. <style type="text/css" media="print, screen and (min-width: 481px)" data-qualweb = "line_9 file_http://www.w3.org">….

21. <body id="www-w3-org" class="w3c_public w3c_home" data-qualweb = "line_21 file_http://www.w3.org">

...

464. <!-- Generated from data/scripts.php, ../../smarty/{scripts.tpl} -->

465. <script type="text/javascript" src = "file_1.js" data-qualweb = "line_465 file_http://www.w3.org">

466. //<![CDATA[

468. //]]>

469. </script>

470. </body>

471. </html>

Figura A.7: site w3.org: excerto do ficheiro /index.html modificado (nome temporário file_0.html)

...

222. }else{BH.text=BI

223. }BJ.insertBefore(BH,BJ.firstChild);BH.setAttribute("data-qualweb","line_223 file_http://www.w3.org/2008/site/js/main");

224. BJ.removeChild(BH)

...

311. W.style.display="none";

312. W.innerHTML=" <link data-qualweb = \"line_312 file_http://www.w3.org/2008/site/js/main\"/><table data-qualweb = \"line_312 file_http://www.w3.org/2008/site/js/main\"></table>a<input type='checkbox'/>";

313. var U=W.getElementsByTagName("*"),T=W.getElementsByTagName("a")[0],P=o.createElement("select"),S=P.appendChild(o.createElement("option"));

...

Figura A.8: site w3.org: excerto do ficheiro /2008/site/js/main modificado (nome temporário file_1.js)

Bibliografia

[1] Abou-Zahra, S. Complete List of Web Accessibility Evaluation Tools, march

2006, http://www.w3.org/WAI/ER/tools/complete.

[2] Wendy Chisholm, Gregg Vanderheiden, and Ian Jacobs. Web Content Accessibility Guidelines 1.0. W3C Recommendation, World Wide Web Consortium (W3C), May 1999. http://www.w3.org/TR/WCAG10/.

[3] Caldwell, B., Cooper, M., Chisholm, W., Reid, L. G. and Vanderheiden, G., Web Content Accessibility Guidelines 2.0. , 2008. , W3C Recommendation, World Wide Web Consortium (W3C) http://www.w3.org/TR/WCAG20/

[4] Abou-Zahra, S. Evaluation and report language (EARL) 1.0 schema. Last call WD, W3C, May 2011. http://www.w3.org/TR/2011/WD-EARL10-Schema-20110510/

[5] Cooper, M., Reid, L. G., Vanderheiden, G., and Caldwell, B.

Techniques for WCAG 2.0 - Techniques and Failures for Web Content Accessibility Guidelines 2.0. W3C Note, World Wide Web Consortium (W3C), October

2010. http://www.w3.org/TR/WCAG-TECHS/.

[6] Lopes, R., and Carriço, L.

Macroscopic characterisations of Web Accessibility. New Review of Hypermedia and Multimedia, Vol. 16, No. 3, December 2010, 221!243

[7] Law, C., Jacko, J., and Edwards, P. Programmer-focused website accessibility evaluations. In Assets ’05 Proceedings of the 7th international ACM

SIGACCESS conference on Computers and accessibility, New York, NY, USA, 2005. ACM.

[8] Fuertes, J. L., González, R., Gutiérrez, E., and Martínez. L.

Developing hera-ffx for wcag 2.0. In W4A ’11: Proceedings of the 2011 International

Cross-Disciplinary Conference on Web Accessibililty (W4A), New York, NY, USA, 2011. ACM.

[9] WAVE - Web Accessibility Evaluation Tool. 2011. http://wave.webaim.org/toolbar/

[10] Foxability - Accessibility Analyzing Extension for Firefox, 2008. http://foxability.sourceforge.net

[11] Firefox Accessibility Extension. Illinois Center for Information Technology and Web Accessibility, University of Illinois. 2011. http://firefox.cita.illinois.edu/

[12] Functional Accessibility Evaluator. Illinois Center for Information Technology and Web Accessibility, University of Illinois. 2011. http://fae.cita.uiuc.edu/

[13] HTML Best Practices. Illinois Center for Information Technology and Web Accessibility, University of Illinois. http://html.cita.illinois.edu/

[14] Resolução do Conselho de Ministros nº 155/2007. http://dre.pt/pdf1sdip/2007/10/19000/0705807058.PDF

[15] Section 508 of Reabilitation Act. http://www.**section508**.gov

[16] Fernandes, N., Lopes, R., and Carriço, L.

On Web Accessibility Evaluation Environments. W4A2011. 2011.

[17] Christopher Bailey and Elaine Pearson. 2012. Evaluation of the effectiveness of a tool to support novice auditors. In Proceedings of the International Cross-Disciplinary Conference on Web Accessibility (W4A '12). ACM, New York, NY, USA, , Article 33 , 10 pages. DOI=10.1145/2207016.2207044.

[18] Nádia Fernandes, Daniel Costa, Sergio Neves, Carlos Duarte, and Luís Carriço. 2012. Evaluating the accessibility of rich internet applications. In Proceedings of the International Cross-Disciplinary Conference on Web Accessibility (W4A '12). ACM, New York, NY, USA, , Article 13 , 4 pages. DOI=10.1145/2207016.2207019

[19] A-Checker. http://achecker.ca/checker/index.php

[20] TAW Standalone. 2012. http://www.tawdis.net/

[21] James Craig and Michael Cooper. Accessible rich internet applications (wai-aria)

1.0. W3C Candidate Recommendation, W3C, January 2011.

http://www.w3.org/TR/wai-aria/