CIC – mecanismo para identificação automática de *widgets* Calendários em *Rich Internet Applications*

Leonardo Sensiate[1], **Humberto Lidio Antonelli**[1], **Renata Pontin de Mattos Fortes**[1]

[1]Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo (USP)
Caixa Postal 13.566-590 – São Carlos – SP – Brazil

{leonardo.sensiate, humbertoantonelli}@usp.br, renata@icmc.usp.br

Abstract. *The Web has provided users with interactivity through Rich Internet Applications (RIAs). However, due to the dynamic nature of RIAs, most applications still do not implement accessible solutions in their components for a variety of reasons, such as the lack of specific knowledge by developers. In addition, it can be observed the lack of tools capable of performing an automatic evaluation for accessibility in RIAs. To evaluate accessibility in RIAs the automated tools mainly require the identification of the dynamic elements. In this paper, we propose an alternative solution,* ***CIC*** *– Calendar Identification Component, that aims to automatically identify Calendar widgets in RIAs.*

Resumo. *A Web proporciona cada vez mais interatividade aos usuários por meio de Rich Internet Applications (RIAs). Entretanto, devido a natureza dinâmica de RIAs, a maioria das aplicações ainda não implementam soluções acessíveis em seus componentes por diversos motivos, como a falta de conhecimento dos desenvolvedores. O primeiro passo para construção de ferramentas automáticas para avaliação de acessibilidade em RIAs é a identificação dos elementos dinâmicos que compõem uma página Web. Este artigo apresenta um mecanismo capaz de identificar automaticamente os componentes dinâmicos presentes em RIAs. Em especial, foi desenvolvido um componente, o* ***CIC*** *– Calendar Identification Component, como instância do mecanismo proposto, demonstrando sua viabilidade.*

1. Introdução

A Web continua em constante processo evolutivo e dentre os avanços, destaca-se o aumento de funcionalidades e da complexidade das tarefas oferecidas nas aplicações, que fornecem interações cada vez mais sofisticadas aos usuários. À medida que as aplicações Web passaram a apresentar cada vez mais interatividade, os usuários passaram a influenciar diretamente na forma como o conteúdo lhes é apresentado durante a navegação [Cooper 2007].

As interfaces implementadas nas páginas Web também tornam-se cada vez mais "ricas" e interativas [Gibson 2007]. Essas interfaces referem-se, em sua maioria, às chamadas *Rich Internet Applications* (RIAs). RIAs são caracterizadas como aplicações Web que visam fornecer recursos e funcionalidades, cuja experiência de uso seja similar à das aplicações *Desktops* tradicionais [Casteleyn et al. 2014].

Atualmente, parte significativa dos *websites* são RIAs ou apresentam componentes (*widgets*) que implementam características de uma RIA. As RIAs estendem os aplicativos

da Web tradicionais fornecendo componentes de interface do lado do cliente, como os *widgets*. Ao utilizarem recursos tecnológicos especializados, desenvolvidos para a Web, RIAs conferem às aplicações Web meios para que o conteúdo seja apresentado de maneira dinâmica ao usuário, frequentemente sem a necessidade de comunicação com o servidor [Almeida and Baranauskas 2012, Fortes et al. 2019]. Sendo assim, a dinamicidade das RIAs pode produzir uma estrutura complexa da interface [Mesbah et al. 2012], acarretando diversos problemas de acessibilidade, caso não tenha sido desenvolvida de maneira adequada, em particular para as pessoas com deficiência [Connor 2012, Fogli et al. 2014].

Observa-se, no entanto, que apesar da ampla utilização de RIAs para disponibilizar conteúdo Web, muitas aplicações são implementadas com sérias barreiras de acessibilidade devido à natureza dinâmica de seus componentes [Antonelli et al. 2018]. Assim, essas aplicações não estão acessíveis aos usuários com deficiências ou mobilidade reduzida [Watanabe et al. 2017].

Do ponto de vista tecnológico, a acessibilidade visa viabilizar a utilização da Web por pessoas com a mais ampla variedade de capacidades, a partir de dispositivos cada vez mais diversificados em diferentes contextos. Do ponto de vista legal e social, acessibilidade possibilita a utilização da Web e de serviços, especialmente governamentais e do Setor Público, a cidadãos com algum tipo de deficiência ou que estejam condicionados a limitações temporárias (chamados Pessoas com Deficiência – PcD). Assim como ocorre nas outras aplicações, a garantia de RIAs usáveis e acessíveis é um aspecto valioso e fundamental para todos [Fortes et al. 2019].

Dentre as razões para a falta de acessibilidade, tem-se: (a) o desconhecimento por parte dos desenvolvedores e (b) a inerente dificuldade em realizar avaliações manuais que auxiliem a identificação das barreiras de acessibilidade presentes em RIAs [Crabb et al. 2019]. Além disso, a falta de ferramentas automáticas que apoiem o processo de avaliação de acessibilidade contribui com a disponibilização de RIAs com graves problemas de acessibilidade [Silva et al. 2018].

Ferramentas automáticas para avaliar acessibilidade em RIAs devem primeiramente identificar os elementos dinâmicos que compõem a página Web. Este trabalho apresenta o desenvolvimento de um mecanismo capaz de identificar automaticamente elementos dinâmicos presentes em RIAs.

Neste artigo, na Seção 2 são discutidos trabalhos significativos na literatura, que se dedicaram a propor avaliar RIAs, considerando seus requisitos específicos. Nas Seções 3 e 4, são apresentadas a concepção e instanciação de um componente identificador de elementos de RIAs, o "Componente Identificador". Na Seção 5, é apresentada uma validação do componente proposto, o CIC (*Calendar Identification Component*), atuando na identificação de um *widget* específico, visando classificá-lo para auxiliar na avaliação automática dos componentes do tipo calendário de uma página Web qualquer. Ao final, na Seção 6, tecemos as considerações finais sobre a proposta apresentada.

2. Avaliação de Acessibilidade nas aplicações Web dinâmicas

O objetivo principal da atividade de avaliação no processo de desenvolvimento de um produto de software é encontrar problemas antes que ele seja disponibilizado aos interessados [ISO/IEC 25066 2016]. No caso de aplicações Web, que geralmente estão em constante manutenção, esse objetivo, portanto, já consiste um desafio. No caso da avaliação

de acessibilidade na Web, a expectativa é verificar se as aplicações podem ser acessadas por pessoas com diferentes deficiências, indicando os problemas existentes, ou seja, os inconvenientes que essas pessoas enfrentam durante o acesso. Esses problemas, que dificultam ou até mesmo impossibilitam o acesso, se tornam "barreiras" ao acesso. Assim, a avaliação de acessibilidade deve ainda apontar essas barreiras para que possam ser analisadas e adequadamente solucionadas [Fortes et al. 2016].

Para lidar com a complexidade das interfaces Web advindas das RIAs, o *World Wide Web Consortium* (W3C) disponibilizou a especificação *Accessible Rich Internet Applications* (WAI-ARIA) [Diggs et al. 2017]. WAI-ARIA é parte da especificação do HTML5, a qual tem por finalidade permitir que usuários, especialmente os que utilizam recursos de Tecnologia Assistiva, consigam interagir com RIAs [Watanabe et al. 2012].

Apesar da especificação WAI-ARIA proporcionar orientações gerais para o desenvolvimento de aplicações dinâmicas acessíveis, ela não apresenta formas para se garantir o cumprimento de suas especificações técnicas. Além disso, o conhecimento sobre este tema ainda é fragmentado [Antonelli et al. 2018], o que geralmente resulta na falta de padronização no desenvolvimento de RIAs. Assim, especialmente os usuários com deficiência ainda enfrentam diversos problemas ao interagir com RIAs [Buzzi et al. 2010], [Crabb et al. 2019].

De acordo com Abou-Zahra (2008), as técnicas de avaliação de acessibilidade podem ser classificadas em três tipos básicos: (a) inspeções manuais, (b) testes automáticos e (c) testes com usuários. Esses tipos de avaliação podem ser combinados em um processo, dependendo do propósito e contexto da aplicação Web em desenvolvimento.

Nas inspeções manuais, os avaliadores analisam a estrutura do código da página para verificar se a implementação dos componentes atende às especificações de acessibilidade, tendo como principal referência as diretrizes propostas pelo W3C. Um dos conjuntos de diretrizes de acessibilidade mais relevantes é o *Web Content Accessibility Guidelines* (WCAG) elaborado pelo W3C [W3C 2018]. Contudo, os custos para a realização de inspeções manuais ou testes com usuários são relativamente altos [W3C 2008, Zhang et al. 2017], quando comparados aos do uso de ferramentas para avaliações automáticas. Nesse sentido, torna-se necessário uma investigação para provimento de ferramentas de avaliação de acessibilidade, capazes de automatizar esse processo.

Segundo Doush *et al.* (2013), as ferramentas de avaliação de acessibilidade convencionais são incapazes de atender aos requisitos inerentes de RIAs, pois apresentam capacidades limitadas de análise da estrutura HTML do conteúdo e não consideram as mudanças e atualizações que são geradas em tempo de execução em RIAs. Na literatura, diversas estratégias se propõem a realizar avaliação de RIAs de maneira automática. A seguir, são descritos os trabalhos propostosque apresentam os principais esforços para atender os requisitos inerentes de RIAs.

No trabalho de Fernandes *et al.* (2011), são destacadas as diferenças ao conduzir avaliações de acessibilidade em aplicações Web, considerando o conteúdo HTML estático e a estrutura dinâmica da página. Com os resultados obtidos, os pesquisadores concluíram que analisar apenas o código estático de uma página Web pode não revelar todos possíveis problemas de acessibilidade, porque o conteúdo avaliado não condiz com o conteúdo que

os usuários finais de fato interagem. Assim, Fernandes *et al.* (2013) incluíram o processo de simulação das interações que os usuários podem realizar nas páginas Web. Os resultados dessa nova abordagem evidenciam que a quantidade de problemas de acessibilidade em RIAs aumenta consideravelmente ao se considerar todas as possíveis alterações advindas das interações dos usuários com o conteúdo disponível. Portanto, fica evidente que conduzir avaliações de RIAs, sem considerar seus componentes dinâmicos, pode fornecer uma percepção equivocada de sua acessibilidade.

Diante deste cenário, Doush *et al.* (2013) apresentaram um método para avaliação automatizada de RIAs, que considera o processo de simulação das interações dos usuários, bem como a identificação e classificação automática de elementos dinâmicos em páginas Web. Esta proposta é promissora, porém é apresentada apenas em nível conceitual, ou seja, não são descritos/ discutidos os meios para viabilizar o projeto e desenvolvimento dos componentes desse método. Ohara *et al.* (2016) e Watanabe *et al.* (2017) também propuseram um processo de identificação dos elementos dinâmicos. Nessas últimas abordagens, o processo de identificação depende de características específicas presentes no código-fonte da aplicação Web, o que dificulta a generalização do processo de identificação para qualquer RIA disponível na Web.

Schiavone e Paternò (2015) desenvolveram uma ferramenta com a finalidade de avaliar a acessibilidade de páginas Web com conteúdo dinâmico, denominada MAUVE – *Multiguideline Accessibility and Usability Validaton Environment*. Essa ferramenta inclui, além dos meios tradicionais de avaliação automática para página estáticas, o uso de um *plugin* acoplado ao navegador capaz de extrair a DOM da página a ser avaliada. O objetivo deste *plugin* é capturar o conteúdo gerado dinamicamente em tempo de execução, o qual corresponde às interações dos usuários. Os resultados obtidos indicam uma boa capacidade de análise e precisão da ferramenta MAUVE em verificar a conformidade com as especificações de acessibilidade. Entretanto, a abordagem ainda apresenta imperfeições no processo de verificação de acessibilidade, uma vez que o conteúdo dinâmico foi avaliado sem considerar todas as possíveis interações dos usuários e como essas interações podem influenciar na estrutura da página.

Watanabe *et al.* (2017) desenvolveram uma ferramenta, denominada **aria-check**, para apoiar especificamente a avaliação de acessibilidade em RIAs. Tal ferramenta delimita o domínio de avaliação para *widgets*, além de incluir um conjunto de casos de teste de aceitação que representam um modelo de interação genérico. Contudo, é importante ressaltar que a abordagem depende de atributos incluídos na estrutura do código-fonte, de modo que seja possível identificar o *widget* na aplicação Web. Assim, caso tais atributos não estejam claramente declarados, a ferramenta aria-check será incapaz de avaliar o modelo de interação acessível do *widget* por meio dos casos de teste de aceitação.

Apesar dos esforços para desenvolver novas estratégias e ferramentas de avaliação automática de acessibilidade, a identificação automática dos elementos dinâmicos ainda é uma das principais limitações evidenciadas nas abordagens que têm sido apresentadas na literatura. Tal limitação deve-se ao fato de não existir um modelo de código padrão que possa ser comparado diretamente. Nesse sentido, este trabalho propõe uma abordagem dedicada a essa identificação, de modo a viabilizar um mecanismo que forneça às ferramentas de avaliação a capacidade de analisar não somente o código-fonte da aplicação, mas também os componentes dinâmicos nela presentes.

3. Concepção de componente para identificação automática de RIAs

Trabalhos reconhecidos na literatura da área consideram o teste com usuários em laboratório como um "padrão ouro" (*gold standard*) dentre os métodos de avaliação. Para analisar o nível de acessibilidade de páginas Web, a utilização de ferramentas automáticas é uma das formas mais simples de avaliar se o conteúdo disponível é acessível, especialmente para usuários com deficiência [Abou-Zahra 2008]. Entretanto, esta visão da existência de um único e melhor método de avaliação é contestada. Apesar da rapidez e falta de exigência de conhecimentos específicos por parte dos avaliadores, esse tipo de avaliação "automatizada" não é completa e deve ser sempre complementada por avaliações com especialistas e por testes com usuários [Petrie and Bevan 2009].

Em avaliações automatizadas de acessibilidade, as ferramentas têm como função verificar a estrutura do código das aplicações Web para conferir o cumprimento das especificações propostas pelo W3C. Como esse é um processo a ser executado de maneira automática, devido ao grande volume de ações e informações manipuladas, qualquer ferramenta que visa a avaliação de acessibilidade em RIA deve ser capaz de gerar e executar sequências de eventos diferentes, de modo que possam ser analisados todos os possíveis estados dinâmicos da aplicação Web.

Para atender às demandas relativas ao tratamento da dinâmica de RIAs, um mecanismo de avaliação automática precisa iniciar identificando quais elementos da estrutura de uma página Web implementam ações que, em resposta a eventos que podem ser disparados pelo usuário, mudem dinamicamente o que é apresentado no navegador Web. Neste trabalho foi proposto um mecanismo para identificação automática dos elementos dinâmicos de uma aplicação Web. Este mecanismo foi idealizado na forma de um "Componente Identificador", como ilustrado na Figura 1.

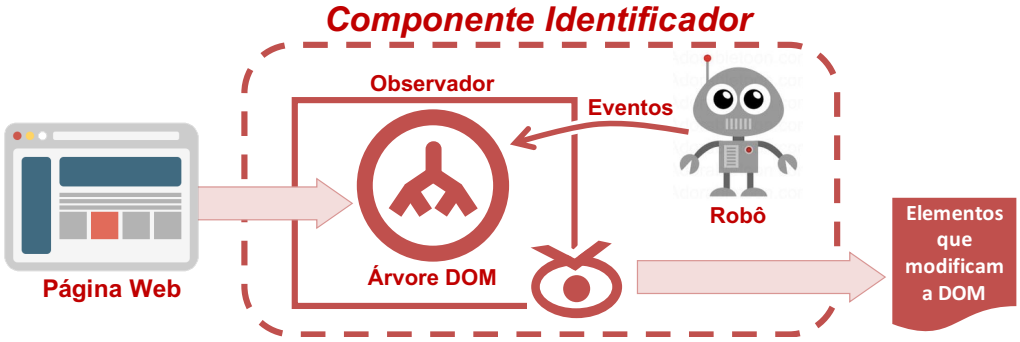

Figura 1. Visão geral do mecanismo de identificação

A concepção do "Componente Identificador" foi realizada, considerando-se o processamento de duas principais funcionalidades:

1. a automatização de disparos de eventos, que imitem as interações de usuários na página Web, e;
2. a obtenção dos estados que a página Web pode assumir, de modo a analisar se houve mudanças mediante os disparo desses eventos.

Fica a cargo de um componente **Robô Web** (representado na Figura 1) a primeira funcionalidade. Esse **Robô** deve ser capaz de simular interações de um usuário real (au-

tomaticamente disparando eventos) com *widgets* disponíveis na interface, de modo que possam ser analisados todos os possíveis estados dinâmicos da aplicação Web.

A segunda funcionalidade é desempenhada pelo componente **Observador** de eventos de RIA, também representado na Figura 1. Esse **Observador** deve ser capaz de detectar as mudanças ocorridas na estrutura de uma página Web, em tempo de execução, além de ser de capaz de identificar qual dos elementos da página foi responsável por provocar tais mudanças, como reação/resposta ao evento disparado.

4. Instanciação de um Componente para Identificação automática de RIAs

A fim de instanciar o "Componente Identificador" proposto (Seção 3), foi realizada uma busca para encontrar uma ferramenta capaz de atuar como **Robô**. O processo de instanciação visa tornar um esquema menos abstrato e aplicável a um caso específico [Darden 2002]. Portanto, o desafio foi viabilizar, por meio de alguma ferramenta já existente, um componente capaz de emular eventos/interações de usuários em aplicações Web dinâmicas, e assim atuar como o componente **Robô**.

Fernandes *et al.* 2013 e Watanabe *et al.* 2012 reportaram o uso de uma abordagem baseada na biblioteca de teste de aplicações Web, denominada *Selenium*[1], que é capaz de gerar eventos reais na interface durante a execução de testes. A partir de uma análise sobre prós e contras em relação aos possíveis mecanismos que pudessem ser empregados na implementação desse **Robô** Web, considerou-se apropriada a adoção do *Selenium*.

Além do *Selenium*, dentre as possíveis ferramentas que poderiam ser empregadas na implementação do **Robô**, existe também o *Crawljax*[2]. O *Crawljax* é uma ferramenta desenvolvida por Mesbah *et al.* (2012), adequada para "rastrear" uma página Web simulando as interações de usuários, e assim procurando os elementos dinâmicos da página, percebendo as alterações geradas por cada um desses elementos. Desse modo, o *Crawljax* funciona como um "robô" rastreador e indexador das aplicações Web com conteúdo dinâmico, especialmente aquelas que fazem uso de AJAX.

Um recurso que merece destaque no *Crawljax* é a capacidade de identificar as mudanças ocorridas na estrutura da página em resposta a um dado evento, criando um diagrama com as relações entre as mudanças de estado plausíveis de ocorrer na aplicação. Nesse diagrama tem-se: o tipo de evento ocorrido, o elemento a partir do qual o evento foi disparado, a estrutura da página correspondente e os possíveis caminhos de navegação presentes na aplicação. A vantagem do *Crawljax* é a possibilidade de tratar aplicações dinâmicas de maneira nativa na ferramenta, o que é fundamental para utilização em ferramentas de avaliação automática de RIAs.

Embora o *Crawljax* possa atuar tanto como **Robô** quanto como **Observador**, há certas limitações que devem ser consideradas. A principal delas é a impossibilidade de disparar outros tipos de eventos além do simples clique do *mouse*. Tal fato é um limitante importante para o escopo do objetivo proposto, uma vez que impede a simulação de diversas sequências de eventos diferentes que podem ser disparados pelo usuário e, por conseguinte, impede a identificação dos demais estados dinâmicos que a aplicação Web pode assumir. Dessa forma, o processo de identificação fica prejudicada, uma vez que

[1] `https://github.com/seleniumhq/selenium`
[2] `https://github.com/crawljax`

componentes dinâmicos podem ser ignorados.

Por outro lado, um dos principais recursos do *Crawljax* é a utilização das bibliotecas e APIs do *Selenium*, a qual permite o disparo de eventos de maneira automática sobre os elementos da aplicação Web. Como o *Selenium* é um conjunto de ferramentas para automatizar a execução de testes, este fornece funcionalidades para interação direta entre navegadores e aplicações Web, sem a necessidade de intervenção do usuário no processo. *Selenium* tem sido usado para gerar eventos reais na interface durante a execução de testes em trabalhos como em Watanabe *et al.* (2012), Fernandes *et al.* (2013) e Ohara *et al.* (2016).

Dentre as ferramentas que compõem o *Selenium*, *WebDriver* é a mais robusta, sendo capaz de automatizar o teste de aplicativos Web e, em particular, verificar se eles funcionam conforme o esperado. Para tanto, possui uma interface de programação na qual podem ser desenvolvidos *scripts* avançados, utilizando-se linguagens de programação tais como C, Python e Java. O *Selenium Webdriver* mostrou-se uma ferramenta adequada para simular as interações de um usuário real, uma vez que é possível programar *scripts* para disparar eventos sobre quaisquer elementos presentes na estrutura de uma página Web.

Nesse sentido, foram acrescentados ao *Selenium* diversos *scripts* desenvolvidos, a fim de verificar a viabilidade de seu uso como componente **Robô**. Esses *scripts* foram executados em aplicações Web reais, de modo a analisar a resposta das RIAs em relação ao disparo de eventos (cliques de *mouse*, envio de caracteres, envio de fomulários e troca de foco entre os elementos da página, etc.) na interface, simulando a interação dos usuários. Finalmente, a ferramenta *Selenium* mostrou-se viável para ser utilizada como componente **Robô**, atuando como simulador das possíveis interações do usuário.

Apesar do *Selenium* apresentar a capacidade de emular quase todos os eventos possíveis, observa-se que a ferramenta não possui recursos para identificar mudanças na interface ou rastrear componentes dinâmicos. Assim, para suprir essa necessidade, foi investigado a adoção do *MutationObserver*[3] em conjunto com o *Selenium*.

O *MutationObserver* é uma API acessada a partir da estrutura DOM, por meio da linguagem *Javascript*, e é capaz detectar alterações na estrutura das aplicações Web. Ao utilizar essa API, é possível observar e monitorar os componentes que são adicionados, modificados ou removidos de qualquer página Web. O *MutationObserver* foi selecionado para atuar como **Observador**, uma vez que realiza a verificação de mudanças na estrutura da página, a medida que o *Selenium* (**Robô**) realiza o disparo dos diversos eventos. Portanto, na atual implementação do "Componente Identificador", o *Selenium* e o *MutationObserver* compõem os dois processos distintos, a serem executados no navegador. A Figura 2 ilustra os componentes instanciados.

A exemplo do que foi feito com o *Crawljax*, essas duas ferramentas em conjunto têm sido testadas em um ambiente controlado. Para tanto, desenvolveu-se uma página HTML que possui diversos elementos de RIAs, os quais podem ser comumente encontrados em aplicações Web reais. Tais testes têm apresentado resultados satisfatórios, uma vez que essa proposta permite emular as interações de usuários de maneira mais completa, além de permitir a identificação e recuperação dos componentes dinâmicos presentes na página de maneira mais simples e intuitiva.

[3] `https://dom.spec.whatwg.org/#mutation-observers`

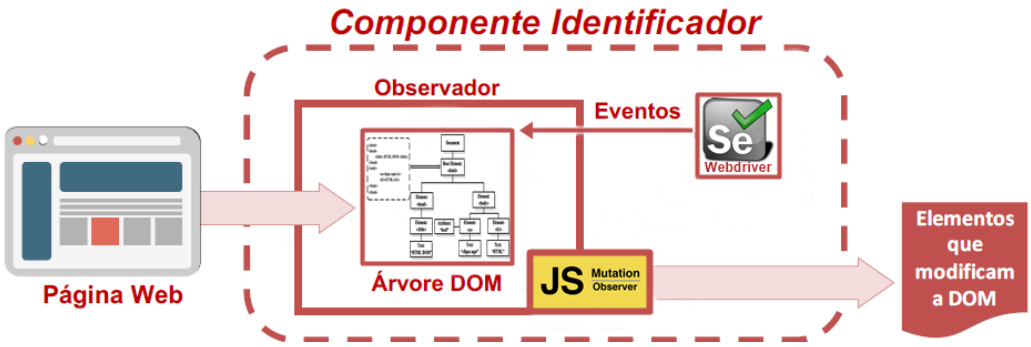

Figura 2. O mecanismo de identificação instanciado.

A Figura 3 ilustra o resultado da execução do mecanismo proposto com o *MutationObserver* atuando simultaneamente com o *Selenium Webdriver*, no qual uma das alternativas é mostrar no próprio navegador o elemento e a mudança dinâmica que resultou da interação da página com o evento disparado. Na Figura 3, estão identificados (destacados em vermelho) os principais recursos de interesse, a partir dos quais se encontra o elemento desejado (destacado em azul).

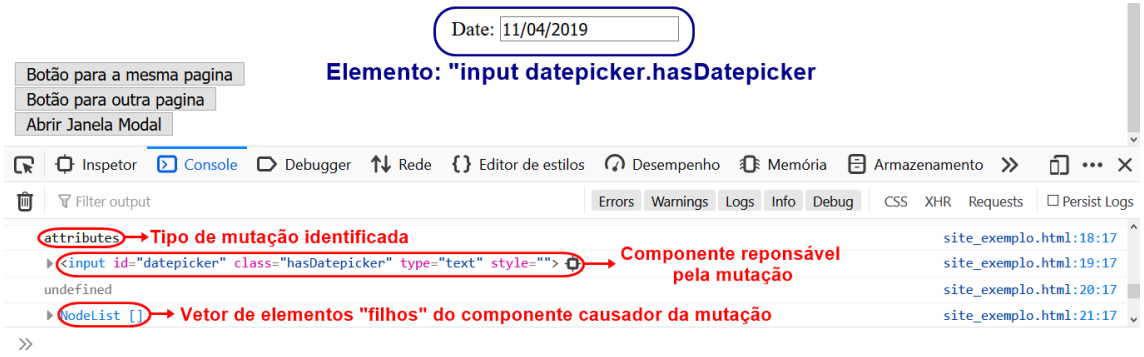

Figura 3. Exemplo de captura das mutações feitas pelo *MutationObserver*.

Como pode ser visto na Figura 3, após a execução do mecanismo é possível recuperar (na saída do código ou no próprio *console* do navegador) o tipo da mutação identificada, o código HTML referente ao elemento e também um vetor com os nós "filhos" do componente, os quais foram gerados durante a mutação dinâmica. Com base nessas informações, é possível extrair as características e propriedades do elemento que disparou as mudanças dinâmicas na página.

Desse modo, a integração das ferramentas que instanciaram o "Componente Identificador" permite identificar as alterações que ocorrem dinamicamente nas páginas Web, bem como os componentes responsáveis por disparar essas alterações. A partir dos componentes identificados, é possível definir quais atributos referentes aos *widgets* podem ser utilizados para sua classificação em quaisquer aplicações. Deste modo, foi disponibilizado um suporte para que ferramentas de avaliação automática de acessibilidade Web funcionem nos diferentes tipos de ambiente.

A seguir, é descrito como foi dado foco à validação da proposta, viabilizando um mecanismo capaz de identificar e recuperar características específicas de componentes

dinâmicos, em particular *widgets* do tipo calendários, neste trabalho.

5. *Calendar Identification Component* (CIC) validando a proposta

Com base no "Componente Identificador" instanciado, realizou-se uma validação e estudo do seu potencial para, com base num processo de estudo sobre componentes dinâmicos específicos, extrair as características gerais que pudessem representar genericamente qualquer elemento. Desse modo, foram obtidas informações específicas, de acordo com cada tipo de elemento identificado, bem como as alterações dinâmicas que executam na página.

Para realizar a validação, decidiu-se trabalhar com diversas implementações de um *widget* de RIA em especial, amplamente utilizado em aplicações Web, o *Datepicker* ou Calendário. Esse *widget* apresenta-se usualmente como um componente de interface gráfica que permite ao usuário selecionar uma data de um calendário. Portanto, o mecanismo instanciado foi especializado, focando nas características específicas desse tipo de componente. Por tal motivo, o mecanismo foi batizado como "CIC" – *Calendar Identification Component*. O CIC pode ser entendido como um "Componente Identificador" adaptado para identificar características específicas de *widgets* de Calendários.

Inicialmente, desenvolveu-se uma página HTML de testes, representando um ambiente controlado, contendo as principais implementações dos *widgets* do tipo calendário de páginas atuais, como ilustra Figura 4.

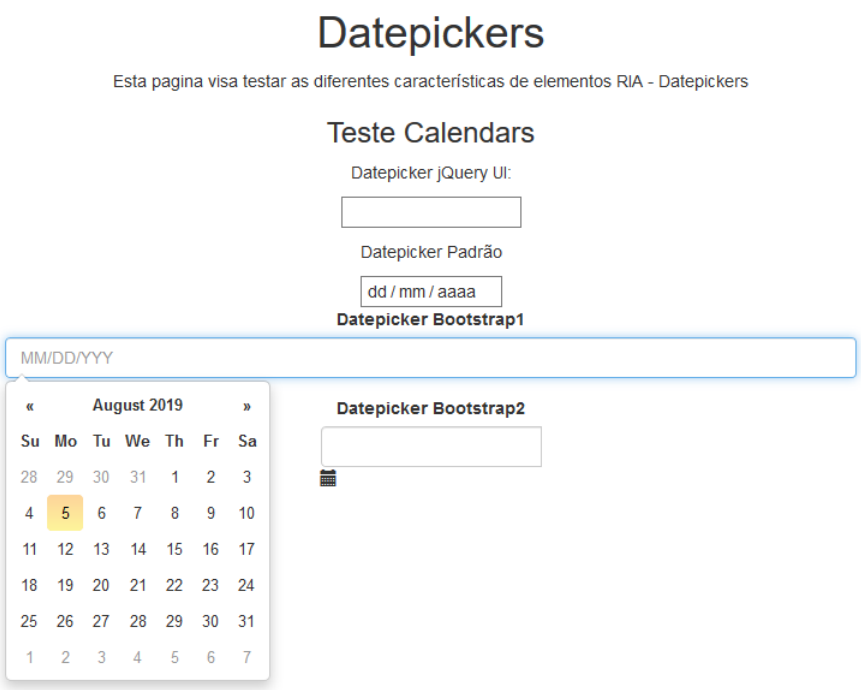

Figura 4. Página HTML contendo diferentes *Datepickers*, elaboradas para estudo e generalização das propriedades destes componentes dinâmicos.

Durante sua execução, o CIC atua na página com ambos os processos do *Selenium* e *MutationObserver*, rodando de maneira simultânea, com o *Selenium* "controlando" o navegador e realizando o disparo de eventos (cliques, submissões de caracteres e formulários, entre outros). A Figura 5 ilustra a atuação do CIC em uma página Web, na qual

é possível observar que o navegador está sendo controlado remotamente pelo **Robô**, além de ser possível observar as interações sendo realizadas com a página.

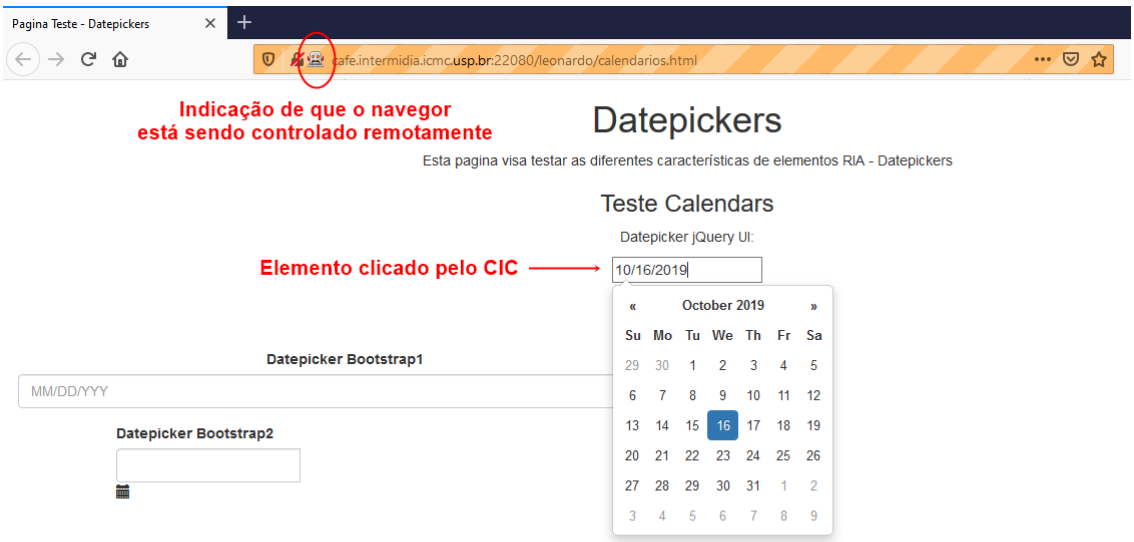

Figura 5. Tela de execução do CIC - disparando os eventos via *Selenium*.

Ao final da execução do CIC, é gerado um arquivo contendo os elementos que compõem a página, bem como suas propriedades e seus nós "filhos" (Figura 6). A obtenção dessas propriedades possibilitou uma análise das diferentes implementações de *widgets* de Calendários presentes na página HTML de testes (Figura 4), viabilizando uma generalização desse tipo de *widget*. Essa generalização foi analisada e inserida como conjunto "conhecido/testado" de verificações no CIC. Finalmente, o CIC analisa as propriedades dos componentes dinâmicos identificados durante sua execução, classificando-os como um *widget* Calendário caso contenha propriedades semelhantes às extraídas dos elementos "testados".

No ambiente controlado, criado para os testes do CIC, foram utilizadas quatro implementações diferentes de Calendários. Duas dessas implementações são derivadas do *framework jQuery UI* [4], outra do *framework Bootstrap* [5] e a última é implementada diretamente via HTML, sem uso de bibliotecas específicas. Ao testar o CIC nesse ambiente controlado, a acurácia foi de 75%, uma vez que o mecanismo identificou 3 dos 4 calendários presentes na aplicação de teste.

Ao analisar os resultados obtidos, foi possível notar que o CIC identificou os Calendários implementados dos *frameworks jQuery UI* e *Bootstrap*, mas não reconheceu o componente implementado puramente com a linguagem HTML. Analisando o comportamento do algoritmo que implementa as verificações no CIC, justifica-se pelo fato do calendário implementado somente com a linguagem HTML não promover uma alteração na estrutura da página (não adiciona e nem remove nós da página) e, portanto, o CIC não o reconhece como um componente dinâmico.

[4]https://jqueryui.com/
[5]https://getbootstrap.com/

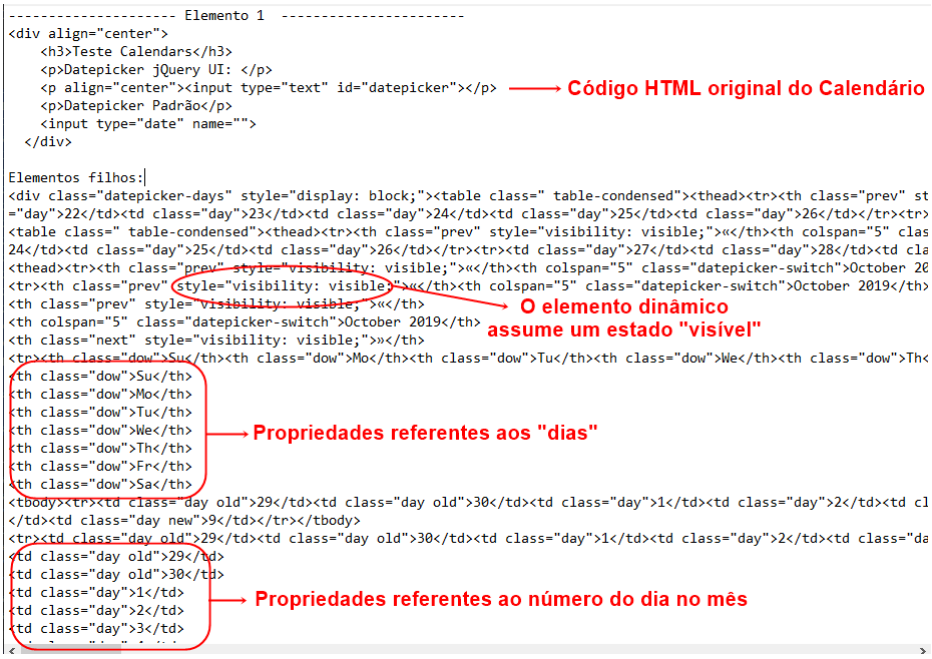

Figura 6. Estrutura dos elementos dinâmicos, recuperada após a execução do CIC.

Para comprovar de maneira mais eficiente o comportamento do CIC e obter maior confiabilidade em seus resultados, o mecanismo foi testado em algumas páginas Web reais, que contêm algum tipo de implementação de calendários. Foram avaliadas as próprias páginas de exemplo do *Bootstrap* e *jQuery UI*. Porém, ao executar o CIC em páginas Web reais, devido à grande quantidade de elementos que promovem o redirecionamento para outras páginas, o mecanismo acabou não funcionando conforme o esperado, uma vez que a alteração do endereço no qual o CIC está sendo executado acarreta perda de referências aos elementos, impedindo a identificação correta dos mesmos.

Para contornar esse comportamento inesperado, foram efetuadas mudanças no mecanismo, de forma a direcionar os disparos de evento via *Selenium* aos principais elementos dinâmicos da página (incluindo os Calendários). Assim, foi possível identificar e recuperar as propriedades dos elementos para então classificá-los como sendo ou não um *widget* Calendário, com igual acurácia atingida nos testes em ambiente controlado.

Portanto, com esse resultado, o mecanismo proposto e projetado demonstrou-se satisfatório por ser capaz de identificar os componentes dinâmicos específicos (calendário) de uma página e comparar suas propriedades, a fim de classificá-los. Posteriormente, esse processo de classificação dos componentes de uma página pode ser adicionado a uma ferramenta de avaliação automática de acessibilidade genérica, fornecendo informações confiáveis e que auxiliem positivamente no processo de avaliação.

Finalmente, utilizando como base o "Componente Identificador" e o CIC, viabilizamos e generalizamos um processo para automatizar a identificação de tipos específicos de RIAs. Ao realizar refinamentos e adequação para os diferentes identificadores, o mecanismo poderá ser executado nos diversos ambientes, além de atuar na comparação e recuperação de quaisquer propriedades que sejam relevantes para a identificação automática dos componentes RIAs desejados.

6. Considerações Finais

Atualmente, a Internet dispõe de grande quantidade de aplicações Web dinâmicas (RIAs - *Rich Internet Applications*), estimulando que as pessoas em geral vivenciem experiências agradáveis, com agilidade e satisfação, ao acessar o conteúdo disponibilizado. Assim, a acessibilidade dessas aplicações torna-se cada vez mais essencial. A observância dos critérios necessários para garantir acessibilidade é vital para o desenvolvimento da Web tanto do ponto de vista tecnológico, quanto do ponto de vista social.

Neste trabalho, foram descritas soluções que auxiliam no processo de avaliação de acessibilidade de RIAs, especificamente abordando a identificação de elementos dinâmicos por meio de um mecanismo genérico, para poder ser utilizado em ferramentas automáticas de avaliação.

A combinação de ferramentas, como *Selenium*, *Crawljax* e *MutationObserver*, possibilitou a obtenção de bons resultados para alcançar o objetivo de integrar em um mecanismo, o processamento adequado para identificação das alterações causadas na estrutura da página Web dinâmica e de seus elementos causadores. Contudo, ainda é necessário realizar ajustes para que o mecanismo final possua todas as funções almejadas prontas e seja capaz de dar o devido suporte a ferramentas de avaliação automáticas.

A partir do mecanismo capaz de identificar componentes dinâmicos, almeja-se extrair mais características gerais que descrevam genericamente os componentes /elementos dinâmicos e as alterações que eles geram, para que os componentes de RIAs presentes na Web possam ser generalizados e classificados. Espera-se que, a partir da generalização de identificação dos tipos de componentes de RIAs, seja possível iniciar a construção e disponibilização de uma Base de Dados com as informações e características de diversos elementos de RIAs. Essa Base deverá ser utilizada como guia no processo de avaliação de acessibilidade desses elementos, conforme orienta a especificação WAI-ARIA.

Como trabalhos futuros, pretende-se continuar investigações sobre o potencial de uso do mecanismo proposto, visando atender demandas diferentes de avaliações automatizadas. Uma outra perspectiva seria a de construir um quadro geral de critérios para análises de inclusão de outras ferramentas ao mecanismo proposto, proporcionando sua evolução contínua e criteriosa.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, bem como pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) sob os números de processo #2015/24525-0 e #2018/06322-3.

Referências

Abou-Zahra, S. (2008). Web accessibility evaluation. In *Web Accessibility: A Foundation for Research*, pages 79–106. Springer London, London.

Almeida, L. D. A. and Baranauskas, M. C. C. (2012). Accessibility in Rich Internet Applications: People and Research. In *Proc. of the 11th Brazilian Symposium on Human Factors in Computing Systems*, IHC '12, pages 3–12, Porto Alegre RS, Brazil. SBC.

Antonelli, H. L., Rodrigues, S. S., Watanabe, W. M., and Fortes, R. P. M. (2018). A Survey on Accessibility Awareness of Brazilian Web Developers. In *Proc. of the 8th Intl. Conf. on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion*, DSAI 2018, pages 71–79, New York, NY, USA. ACM.

Buzzi, M. C., Buzzi, M., Leporini, B., Mori, G., and Penichet, V. M. R. (2010). Accessing Google Docs via Screen Reader. In *Computers Helping People with Special Needs: 12th International Conference, ICCHP 2010, Vienna, Austria, July 14-16, 2010. Proceedings*, pages 92–99. Springer Berlin Heidelberg, Berlin, Heidelberg.

Casteleyn, S., Garrigós, I., and Mazón, J.-N. (2014). Ten Years of Rich Internet Applications: A Systematic Mapping Study, and Beyond. *ACM Trans. Web*, 8(3):18:1–18:46.

Connor, J. O. (2012). *Pro HTML5 Accessibility*. Apress, New York, NY, 1st. edition.

Cooper, M. (2007). Accessibility of emerging rich web technologies: Web 2.0 and the semantic web. In *Proc. of the 2007 Intl Cross-disciplinary Conf. on Web Accessibility (W4A)*, W4A '07, pages 93–98, New York, NY, USA. ACM.

Crabb, M., Heron, M., Jones, R., Armstrong, M., Reid, H., and Wilson, A. (2019). Developing accessible services: Understanding current knowledge and areas for future support. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 216:1–216:12, New York, NY, USA. ACM.

Darden, L. (2002). Strategies for discovering mechanisms: Schema instantiation, modular subassembly, forward/backward chaining. *Philosophy of Science*, 69(S3):S354–S365.

Diggs, J., McCarron, S., Cooper, M., Schwerdtfeger, R., and Craig, J. (2017). *Accessible Rich Internet Applications (WAI-ARIA) 1.1*. World Wide Web Consortium (W3C).

Doush, I. A., Alkhateeb, F., Maghayreh, E. A., and Al-Betar, M. A. (2013). The design of RIA accessibility evaluation tool. *Advances in Engineering Software*, 57:1–7.

Fernandes, N., Batista, A. S., Costa, D., Duarte, C., and Carriço, L. (2013). Three web accessibility evaluation perspectives for RIA. In *10th International Cross-Disciplinary Conference on Web Accessibility*, W4A '13, pages 12:1–12:9, New York, NY, USA. ACM.

Fernandes, N., Lopes, R., and Carriço, L. (2011). On web accessibility evaluation environments. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, W4A '11, pages 4:1–4:10, New York, NY, USA. ACM.

Fogli, D., Provenza, L. P., and Bernareggi, C. (2014). A universal design resource for Rich Internet Applications based on design patterns. *Universal Access in the Information Society*, 13(2):205–226.

Fortes, R. P. M., Antonelli, H. L., and Salgado, A. L. (2016). Avaliação de Acessibilidade e Usabilidade em RIA. In de Jesus Lima Gomes et al, F., editor, *Anais do XXII Simpósio Brasileiro de Sistemas Multimídia e Web (Vol. 3): Minicursos*, chapter 2, pages 37–65. Teresina, Piauí.

Fortes, R. P. M., Antonelli, H. L., and Watanabe, W. M. (2019). Dynamic Web Content. In Yesilada, Y. and Harper, S., editors, *Web Accessibility: A Foundation for Research*, pages 373–395. Springer London, London.

Gibson, B. (2007). Enabling an accessible web 2.0. In *Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, volume 1 of *W4A '07*, pages 1–6, New York, NY, USA. ACM.

ISO/IEC 25066 (2016). Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Common Industry Format (CIF) for Usability — Evaluation Reports. `https://www.iso.org/obp/ui/#iso:std:iso-iec:25066:ed-1:v1:en`.

Mesbah, A., van Deursen, A., and Roest, D. (2012). Invariant-based automatic testing of modern web applications. *IEEE Transactions on Software Engineering*, 38(1):35–53.

Ohara, T., Iwata, H., Shirogane, J., and Fukazawa, Y. (2016). Support to apply accessibility guidelines to web applications. *International Journal of Computer and Communication Engineering*, 5(2):99–109.

Petrie, H. and Bevan, N. (2009). The Evaluation of Accessibility, Usability, and User Experience. In *The Universal Access Handbook*, Human Factors and Ergonomics, pages 1–16. CRC Press.

Schiavone, A. G. and Paternò, F. (2015). An extensible environment for guideline-based accessibility evaluation of dynamic web applications. *Universal Access in the Information Society*, 14(1):111–132.

Silva, C., Eler, M. M., and Fraser, G. (2018). A Survey on the Tool Support for the Automatic Evaluation of Mobile Accessibility. In *Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion*, DSAI 2018, pages 286–293, New York, NY, USA. ACM.

W3C (2008). Conformance Evaluation of Web Sites for Accessibility. `http://www.w3.org/WAI/eval/conformance.html`.

W3C (2018). WCAG 2.1 - Web Content Accessibility Guidelines. `http://www.w3.org/TR/WCAG21/`.

Watanabe, W. M., Fortes, R. P. M., and Dias, A. L. (2012). Using Acceptance Tests to Validate Accessibility Requirements in RIA. In *Proc. of the International Cross-Disciplinary Conference on Web Accessibility*, W4A '12, pages 15:1–15:10, New York, NY, USA. ACM.

Watanabe, W. M., Fortes, R. P. M., and Dias, A. L. (2017). Acceptance tests for validating ARIA requirements in widgets. *Universal Access in the Information Society*, 16(1):3–27.

Zhang, M., Wang, C., Yu, Z., Shen, C., and Bu, J. (2017). Active learning for web accessibility evaluation. In *Proc. of the 14th Web for All Conference on The Future of Accessible Work*, W4A '17, pages 16:1–16:9, New York, NY, USA. ACM.