UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de São José do Rio Preto

Livia Cristina Gabos Martins

ModelUI$_{VIZ}$ - Uma proposta para o entendimento da interface do usuário utilizando técnicas de Visualização de Informação

São José do Rio Preto
2017

Livia Cristina Gabos Martins

ModelUI$_{VIZ}$ - Uma proposta para o entendimento da interface do usuário utilizando técnicas de Visualização de Informação

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Orientador: Prof. Dr. Rogério Eduardo Garcia

São José do Rio Preto
2017

Martins, Livia Cristina Gabos.
ModelUI$_{VIZ}$ - Uma proposta para o entendimento da interface do usuário utilizando técnicas de Visualização de Informação / Livia Cristina Gabos Martins. -- São José do Rio Preto, 2017
125 f. : il., tabs.

Orientador: Rogério Eduardo Garcia
Dissertação (mestrado) – Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas

1. Computação. 2. Interfaces (Computação) 3. Interface de programação de aplicativos (software de computador). 4. Engenharia de Software . 5. Visualização da informação. 6. Programação para internet. I. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. II. Título.

CDU – 518.72

Livia Cristina Gabos Martins

ModelUI$_{VIZ}$ - Uma proposta para o entendimento da interface do usuário utilizando técnicas de Visualização de Informação

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Comissão Examinadora

Prof. Dr. Rogério Eduardo Garcia
UNESP – Presidente Prudente
Orientador

Profª. Drª. Débora Maria Barroso Paiva
UFMS – Campo Grande

Prof. Dr. Milton Hirokazu Shimabukuro
UNESP – Presidente Prudente

Profª. Drª. Maria Istela Cagnin
UFMS – Campo Grande

Prof. Dr. Ronaldo Celso Messias Correia
UNESP – Presidente Prudente

São José do Rio Preto
2017

"Coloque-se sempre de frente para a vida, porque, se não o fizer, estará dando as costas à realidade."

Carlos Bernardo González Pecotche, *Bases para sua conduta*

Agradecimentos

Gosto de pensar que algumas situações são necessárias para que se obtenha os aprendizados da vida. O mestrado me deu a oportunidade de aprender muito mais sobre a vida. Sair de Bauru e começar do zero em Presidente Prudente, para mim, foi uma grande oportunidade de vida.

Eu só tenho que agradecer à minha mãe e meu marido pelo apoio e amor incondicional ao longo do mestrado. Sem vocês ao meu lado, em todos os momentos, NADA disso seria possível. Obrigada por acreditarem em mim.

Agradeço ao Rogério Eduardo Garcia, meu orientador, pela grande paciência durante esse período. Você acolheu uma desconhecida e eu só tenho a agradecer pelo tanto que aprendi com você.

Agradeço minha família de Presidente Prudente, Yara Oliveira Florêncio da Hora e Karla Gagg pelo acolhimento e apoio que vocês me proporcionaram nessa cidade, inicialmente desconhecida para mim. Muito obrigada pela paciência, carinho e comidas gostosas.

Muito obrigada aos meus amigos de Bauru pela compreensão nos momentos de ausência e oferecimento de ajuda, sempre que possível. Principalmente ao Bruno Fernandes Casella pelos empréstimos de tempo para a colaboração nas escritas em inglês. Aos outros amigos, me desculpem por não listar todos, mas você sabe que estou falando de vocês.

Agradeço aos meus companheiros de laboratório, pelas conversas e momentos de descontração. Especialmente ao Lenon Fachiano Silva e à Ingrid Marçal, muito obrigada pelo amparo nos momentos de desespero que passamos juntos. Vocês foram essenciais durante todo o processo.

Aos alunos que tive durante todo esse período, obrigada por me ajudarem a descobrir e reforçar algo que gosto tanto: compartilhar conhecimento.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Contexto: Os modelos de interface são utilizados para representar suas características sob diferentes aspectos e, assim, facilitar a compreensão das informações da interface do usuário. Na literatura são utilizadas representações da Engenharia de Software, como a UML, e suas extensões para os modelos de interface. **Motivação e Justificativa:** As informações são dispersas em múltiplos modelos e níveis de abstração, o que motiva pesquisas para criar mecanismos que facilitem o entendimento da interface. Ter informação sobre a implementação da interface atualizada contribui para seu entendimento. **Problema:** A dispersão e os vários níveis de abstrações dos modelos são problemas para a atualização constante desses modelos, de modo a manter consistência entre eles. E consequentemente, gera o problema aqui tratado: a dificuldade da manutenção é maior por não existir artefatos confiáveis que a apoiem. Quanto às interfaces *web* o problema se agrava, não tendo sido observado trabalhos na literatura que permitam uma visão geral da implementação da interface. **Objetivo:** O objetivo deste trabalho é apoiar o entendimento da implementação de interfaces *web* usando técnicas de Visualização de Informação, facilitando a compreensão da sua estrutura e do processo de manutenção da interface. **Metodologia:** Para isso, é proposta a ModelUI$_{VIZ}$, que consiste em modelos visuais para a representação das informações de uma interface *web*. A ModelUI$_{VIZ}$ é organizada em *WebModelUI Data* – um *plug-in* para extração dos dados da interface – e a ferramenta *WebModelUI Tool* – que realiza a leitura dos dados gerados pelo *plug-in*, apresentando-os visualmente. Para avaliar a compreensão da representação visual criada foram realizados três experimentos controlados. Entre as avaliações, foram criadas três versões com melhorias e novas funcionalidades para a representação visual. **Resultado:** Como resultado, a última versão da ModelUI$_{VIZ}$ foi compreendida e aceita pelos participantes do experimento, e importantes lições aprendidas foram alcançadas.

Palavras-chave: Compreensão da interface do usuário. Modelos de Interface. Engenharia Reversa. Visualização de Informação.

Abstract

Context: User Interface (UI) models are used to represent their characteristics under different aspects and, thus, facilitate the understanding of UI. In the literature, are used representations of Software Engineering, such as UML, and the extensions to UI models. **Motivation and Justification:** The information is scattered in multiple models and levels of abstraction, which motivates this research to create mechanisms that facilitate the understanding of user interface. Having updated information about UI implementation has contributed to understanding. **Problem:** The scattered information and the multiple levels of abstractions are problems to update models, in order to keep consistency among them. And consequently, it generates the problem treated here: the difficulty of maintenance is greater because there are no reliable artifacts to support it. Regarding to web interfaces the problem is aggravated, there is no observed works in literature that allow an overview of user interface implementation. **Goal:** The goal of this work is to support the understanding of web interface, using Information Visualization techniques, in order to facilitate the understanding of its structure and, consequently, the maintenance task. **Methodology:** We proposed the ModelUI$_{VIZ}$, which consists of visual models to represent information from user interface. The ModelUI$_{VIZ}$ is organized into *WebModelUI Data* – a plug-in to extract the data from UI – and *WebModelUI Tool* – which reads the data generated by the plug-in, presenting them visually. To evaluate the comprehension of visual representation were conducted three controlled experiments. Among the evaluations, we were created three versions with improvements and new functionalities for visual representation. **Result:** The latest version of ModelUI$_{VIZ}$ were understood and accepted by participants of experiment, and important lessons learned were reached.

Keywords: Understanding User Interfaces. User interface models. Reverse Engineering. Information Visualization.

CAPÍTULO 1

Introdução

1.1 Contextualização

O *Object Management Group (2014)* define modelo como uma descrição abstrata de um sistema e seu ambiente, feita por meio textual, gráfico ou uma combinação gráfico-textual. Modelos são muito utilizados para facilitar a verificação da conformidade com os requisitos, colaborar na compreensão do software no processo de desenvolvimento, facilitando a comunicação entre pessoas com diferentes níveis de conhecimento por meio de um vocabulário comum (Raneburger et al., 2012).

De acordo com Puerta (1997), modelos de interface podem ser definidos como uma representação declarativa de aspectos relevantes de uma interface de usuário, como componentes e leiaute. Os componentes da interface (por exemplo, botões) são incorporados aos modelos com diferentes níveis de abstração. Tais componentes podem ser relacionados às tarefas, aos elementos de domínio, aos diálogos entre o sistema e o usuário, e às opções relacionadas ao leiaute (Puerta, 1997). Na literatura, modelos de interface são utilizados em diversas etapas do processo de desenvolvimento do software, desde a criação de interface até os testes baseados em modelos.

Em metodologias de desenvolvimento dirigidas a modelos (do inglês *Model- Driven Development* - MDD), os modelos de interface podem ser usados para gerar múltiplas versões para diferentes ambientes (*web, mobile*, etc) de uma mesma interface. Além disso, nas extensões específicas para interface, como *Model-driven UI Generation* (Raneburger et al., 2012) e na *Model-Based Design of User Interfaces* (MBUI) (Motti et al., 2013), os modelos de interface podem ser utilizados como oráculos das características da interface.

Independente da metodologia, modelos de interface podem ser expressos utilizando diferentes notações ou linguagens com diferentes níveis de abstração. Na lite-

ratura são utilizadas notações usuais da Engenharia de Software, como a UML (Markopoulos e Marijnissen, 2000, Almendros-Jimenez e Iribarne, 2009, Silva e Silveira, 2010) e suas extensões para a interface – como a UMLi (Silva e Paton, 2003b). Existem, também, notações criadas especificamente para o contexto da interface do usuário, como a CTT (*ConcurTaskTree*) (Paternò et al., 1997). A CTT é uma das notações utilizadas na literatura para a modelagem das tarefas do usuário (Ahmed e Ashraf, 2007, Nunes, 2003, Almendros-Jiménez e Iribarne, 2008, Pleuss et al., 2013).

1.2 Formulação do problema

Na literatura, diferentes modelos em diferentes níveis de abstração e complementares entre si são utilizados para representar todas as informações que compõem uma interface (Almendros-Jimenez e Iribarne, 2009). Em geral, as informações de um único componente da interface são representadas em vários tipos de modelos (comportamento, navegação, tarefa, etc.), apresentando apenas o aspecto relacionado àquele modelo. Além disso, os diferentes tipos de modelos de interface são subdivididos em níveis de abstração, para apresentar todas as informações de uma interface (Almendros-Jimenez e Iribarne, 2009).

As propostas de diferentes representações para os modelos de interface na literatura, como UML, UMLi e CTT, não colaboram para diminuir a dispersão das informações e/ou têm o objetivo de diminuir os níveis de abstração das informações dos modelos. Ao tentar relacionar todas as informações de um mesmo componente em diversos modelos, essa dispersão e os vários níveis de abstração dificultam encontrar e relacionar essas informações. Além disso, de acordo com a representação utilizada existe a fase de adaptação dos desenvolvedores a compreenderem à representação.

Estudos realizados em contextos diferentes de modelos de interface mostram que a utilização de representações conhecidas e com ferramentas de modelagem disponíveis, não evita que existam problemas relacionados à inconsistência entre os modelos (Lange et al., 2006, Petre, 2013).

Além disso, a interface do usuário é uma das partes do projeto de software que mais sofrem alterações durante o processo de desenvolvimento. Por essas constantes mudanças, os artefatos relacionados à interface são comumente dispensados de uso pela falta de atualização, sendo utilizada a comunicação verbal e esboços informais no processo de manutenção (Theunissen e van Heesch, 2016). No ambiente *web* esses problemas se agravam, pela dificuldade em representar a implementação da interface, em sua totalidade, de maneira clara e compreensiva. Na literatura, a maioria dos trabalhos utilizam a UML como representação da interface, sem o objetivo de sanar os problemas destacados (Antoniol et al., 2004, Amalfitano et al., 2010b, Marchetto et al., 2012), ou representam apenas uma pequena parte das informações da interface (Cloutier et al., 2016, Zaidman et al., 2013).

Portanto, o problema tratado neste trabalho é a dificuldade de entendimento do es-

tado atual da interface do usuário, causada pela falta de artefatos formais atualizados.

1.3 Motivação e Justificativa

A representação dispersa das informações dos modelos de interface aumenta a probabilidade de má interpretação dos modelos e da falta de padronização no detalhamento dos modelos (Lange et al., 2006, Petre, 2013).

Na literatura, a maioria das representações encontradas não permitem a compreensão da interface como um todo, apresentando apenas parte das informações e sem a possibilidade relacionar informações de acordo com a interação do usuário quando necessário.

Uma representação com maior interação e um detalhamento pontual da estrutura da interface, quando requerido, auxilia na exploração da interface e na compreensão da estrutura da interface. Por consequência, colabora no processo de manutenção da interface.

1.4 Objetivos

O objetivo deste trabalho é apoiar o entendimento da interface do usuário durante o processo de manutenção da interface com a criação de artefatos de qualidade e fácil compreensão. Para isso, é proposta uma representação visual das informações de uma interface *web* utilizando técnicas de Visualização de Informação, com foco de aplicação nos modelos navegacional, comportamental e de diálogo da interface do usuário. Assim, tem-se como objetivos específicos:

- Especificar a representação visual (ModelUI$_{Viz}$) da estrutura da interface do usuário;
- Desenvolvimento da ferramenta de extração das informações da interface do usuário (*WebModelUI Data*);
- Desenvolvimento da ferramenta de apresentação da ModelUI$_{Viz}$ (*WebModelUI Tool*), baseado nas informações extraídas da interface;
- Avaliação da compreensão do modelo visual utilizando um experimento controlado.

1.5 Organização do trabalho

Este trabalho está organizado como segue:

- No Capítulo 2 são apresentados os modelos navegacional, comportamental e de diálogo de uma interface. Mostra-se, também, as abordagens disponíveis na literatura sobre como as informações dos modelos são extraídas, e um referencial teórico sobre validação de modelos.

- No Capítulo 3 são apresentados os conceitos básicos de Visualização de Informação; técnicas de visualização aplicadas à Visualização de Software; técnicas de interação e coordenação relevantes a esta proposta; e um referencial bibliográfico sobre avaliação da aplicação de técnicas de visualização.
- No Capítulo 4 são apresentados o plug-in desenvolvido para a extração das informações da interface; a ferramenta e a representação visual desenvolvida utilizando as técnicas de Visualização de Informação; bem como as versões da ferramenta e da representação visual.
- No Capítulo 5 são apresentados os experimentos realizados para avaliar a ferramenta, a representação visual criada e seus resultados.
- No Capítulo 6 são apresentados as conclusões; limitações do trabalho; propostas futuras e as produções bibliográficas.

CAPÍTULO 2

Modelos de interface

Os modelos de interface são utilizados neste projeto, e por isso, neste capítulo são descritos os tipos de modelos (Seção 2.1) existentes na literatura. Dentre esses tipos, os modelos de navegação, de diálogo e de comportamento foram selecionados como objetos de estudo, por apresentarem as informações relevantes para o processo de criação de uma interface do usuário e por serem os modelos mais utilizados na literatura. Para cada um desses modelos são mostradas as informações que possuem e como são representadas na literatura. Conhecer as características dos modelos selecionados é um dos primeiros passos para definir quais técnicas de Visualização de Informação podem ser utilizadas e qual o modo mais relevante de apresentar essas informações.

Existem diversos métodos de criação e utilização dos modelos de interface na literatura para as abordagens da engenharia avante e engenharia reversa. Por este trabalho ter o foco no processo de manutenção de interfaces, são utilizadas interfaces finalizadas como objetos de estudos e técnicas de engenharia reversa para extração de informações da interfaces (Seção 2.2). O ambiente *web* foi escolhido para aplicação das técnicas de extração pela complexidade das informações existentes em uma interface *web* (Theunissen e van Heesch, 2016) e pela facilidade na separação da camada de interface do restante do software. Nas Seções 2.3.1 e Seção 2.3.2 são apresentadas, respectivamente, as técnicas de sessões com usuários e a técnica de *crawling*, da abordagem dinâmica, utilizadas para extrair dados relacionados aos modelos de interface. Na Seção 2.4 foram relacionadas algumas das ferramentas encontradas na literatura utilizadas para a compreensão da interface do usuário e na Seção 2.5 é mostrada a análise sobre a validação dos modelos de interface.

2.1 Tipos de modelos

Dentre os modelos de interface existentes na literatura, tem-se o modelo de tarefa, de domínio, de usuário, de diálogo, de apresentação, comportamental e navegacional. O **modelo de tarefa** (*task model*) descreve as atividades que o usuário deve realizar para alcançar determinado objetivo (Paganelli e Paternò, 2003). Esse modelo é muito utilizado na literatura com a representação *ConcurTaskTrees* (CTT) (Paternò et al., 1997) para auxiliar no desenvolvimento da interface e para colaborar nos testes de usabilidade (Paternò, 2002).

O **modelo de domínio** (*domain model*) descreve as relações entre objetos de um domínio específico (tarefa) (Almendros-Jiménez e Iribarne, 2008) e características como (Schlungbaum, 1996):

- A hierarquia de classe de objetos que existem na aplicação
- As propriedades dos objetos
- Ações que podem ser realizadas sobre os objetos
- As unidades de informações (parâmetros) requeridas pelas ações
- As pré e pós-condições para as ações

O **modelo do usuário** (*user model*) possui os requisitos para a personalização da interface pelo usuário, de acordo com suas preferências, contexto e dispositivos de uso (Ahmed e Ashraf, 2007, Almendros-Jiménez e Iribarne, 2008). De acordo com Schlungbaum (1996), o modelo do usuário colabora para identificar as características usadas para a tomada de decisões no *design* da interface do usuário.

O **modelo de apresentação** (*presentation model*) descreve a maneira que a interface é apresentada ao usuário, com seus componentes e informações de leiaute (Ahmed e Ashraf, 2007, Almendros-Jiménez e Iribarne, 2008). Esse modelo pode apresentar as informações em um modelo abstrato ou um modelo concreto. No modelo abstrato os componentes são declarados de modo genérico, sem fazer relação a uma tecnologia específica. No modelo concreto os objetos são descritos para cada uma das plataformas utilizadas (Almendros-Jiménez e Iribarne, 2008).

Dentre os modelos existentes, os modelos navegacional, comportamental e de diálogo foram selecionados como objeto de estudo. Esses modelos são os mais utilizados na literatura Amalfitano et al. (2010b,a), Marchetto et al. (2012) por agregarem mais valor para a documentação inicial e por serem mais efetivos para a compreensão da estrutura da interface do usuário (pelo tipo de informação que apresentam). Esses modelos são descritos com mais detalhes a seguir.

2.1.1 Modelo navegacional

O modelo navegacional possui informações detalhadas da estrutura da interface, apresentando os possíveis caminhos que o usuário pode utilizar (Jeschke et al., 2008, Silva, 2010, Amalfitano et al., 2011), sem apresentar a resposta do sistema. De acordo com Wolff e Forbrig (2009), o modelo de navegação pode ser derivado da abstração do modelo de tarefas. Almendros-Jiménez e Iribarne (2008) afirmam que em interfaces *web* o modelo de diálogo pode ser substituído pelo modelo de navegação, por ser possível a conexão entre páginas pelos *links*. No entanto, em interfaces com maior complexidade, com muitos componentes e diferentes caminhos de navegação, ambos modelos (diálogo e navegação) se tornam necessários, sendo complementares entre si.

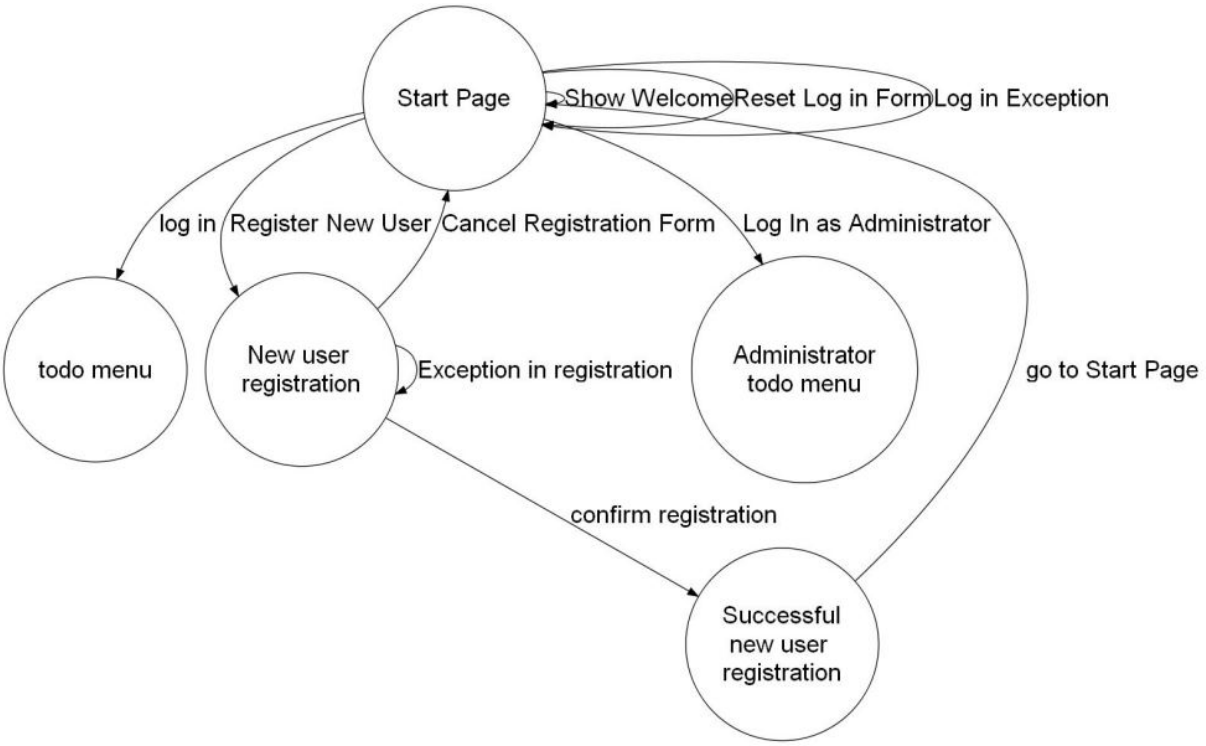

Figura 2.1: Exemplo de modelo de navegação correspondente aos casos de uso Login e registro de um aplicativo *web*, criado pela ferramenta *CReRIA* – Adaptado de Amalfitano et al. (2011)

Para criar um modelo navegacional é necessário observar todos os comandos de direcionamento realizados pela linguagem da interface. Na sintaxe do HTML, por exemplo, existem *tags* que direcionam o usuário para outras páginas quando acionadas, como:

- <a>
- <button type = “submit”>
- <input type = “submit”>

Na literatura o modelo navegacional é representado utilizando grafos (Silva, 2010, Amalfitano et al., 2011) ou *statecharts* (Leung et al., 2000, Winckler e Palanque, 2003) para representar as ligações entre os objetos disponíveis para navegação. Um exemplo de um modelo de navegação é mostrado na Figura 2.1, que apresenta a navegação de um aplicativo *web*.

2.1.2 Modelo comportamental

Em trabalhos como de Puerta (1997), Ahmed e Ashraf (2007), Almendros-Jiménez e Iribarne (2008), as informações sobre estados e transições da interface pertencem ao modelo de diálogo. No entanto, no trabalho de Silva e Paton (2003a) o modelo comportamental é apresentado de maneira separada dos outros modelos, colaborando na diminuição de informações apresentadas pelo modelo. Por isso, neste trabalho o modelo de comportamento é apresentado separado do modelo de diálogo.

O modelo comportamental apresenta os estados que a interface e seus componentes podem assumir durante sua execução (Silva e Paton, 2003a). Na literatura são utilizados diagramas de estados (Almendros-Jimenez e Iribarne, 2009) ou máquinas de estados (Mesbah et al., 2012) para representar o modelo comportamental da interface. Um exemplo de alteração de estado é mostrado na Figura 2.2.

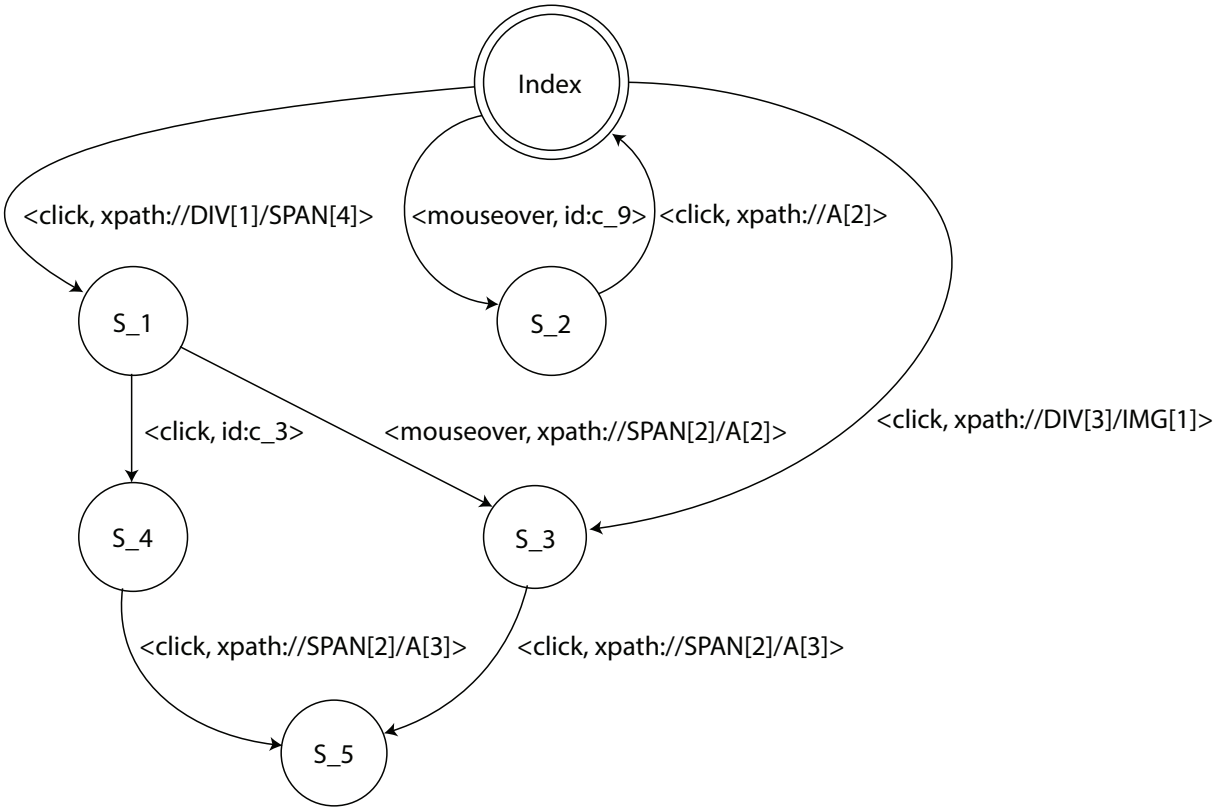

Figura 2.2: Grafo de fluxo de uma interface criado pela ferramenta *Crawljax* - (Mesbah et al., 2012, p. 6)

Mesbah et al. (2012) utilizam grafos para representar o comportamento da inter-

face. Para todos os elementos DOM (componentes HTML que possuem atributos como eventos, estados, etc.) da interface existem eventos associados que observam a ocorrência ou não de mudanças nos atributos (*event listeners*). Para registrar as mudanças nos estados foi utilizada a seguinte definição:

Definição 1. Um grafo de estados G para um *site* em Ajax **A** é nomeado, direcionado e denotado por 4 tuplas < r,*V*,*E*, L> sendo:

1. r é o nó inicial que representa o estado inicial depois que **A** for carregado no navegador.

2. *V* é um conjunto de vértices que representam os estados. Cada $v \in V$ representam um estado de um elemento DOM em **A**.

3. *E* é um conjunto de arestas entre os vértices. Cada (v_1, v_2) $\in$ *E* representa uma ligação entre dois estados de um elemento c clicável se, e somente se, o estado v_2 é alcançado pela execução do elemento c no estado v_1.

4. L é uma função que rotula um conjunto de tipos de eventos e as propriedades de um elemento DOM para cada uma das arestas.

5. G pode ter múltiplas arestas e ser cíclico.

Todos os estados são criados a partir do estado inicial e adicionados aos grafos de acordo com o rastreamento dos estados.

2.1.3 Modelo de diálogo

De acordo com Puerta (1996), o modelo de diálogo representa as ações da interface de acordo com a interação do usuário na interface. Basicamente, esse modelo descreve como o usuário insere os dados, como ele interage com a interface e como a interface apresenta os dados (Almendros-Jiménez e Iribarne, 2008). Essas ações podem ser decompostas em outras com detalhes, de acordo com as respostas do sistema. Para representar as informações do modelo de diálogo podem ser utilizadas redes de Petri (Elkoutbi e Keller, 2000), *statecharts* (Horrocks, 1999, Silva e Paton, 2003b) ou diagramas de atividades (Nunes, 2003, Almendros-Jiménez e Iribarne, 2008).

Um exemplo do uso de diagramas de estados para os modelos de interface é mostrado nas figuras 2.3 e 2.4, que exibem representações da interação do usuário em uma interface de compras, usando um diagrama de estados. Na Figura 2.3 são apresentados no diagrama *UI_Purchase* os estados de uma interface de um sistema de compra. Na Figura 2.4 são mostrados detalhes do estado *Manager Shopping Cart* da Figura 2.3. No diagrama *UI_QueryCatalogue* é detalhado o estado *Query Catalogue* e no *UI_ShoppingCart* é detalhado o estado *Shopping Cart* do diagrama *UI_Manage*

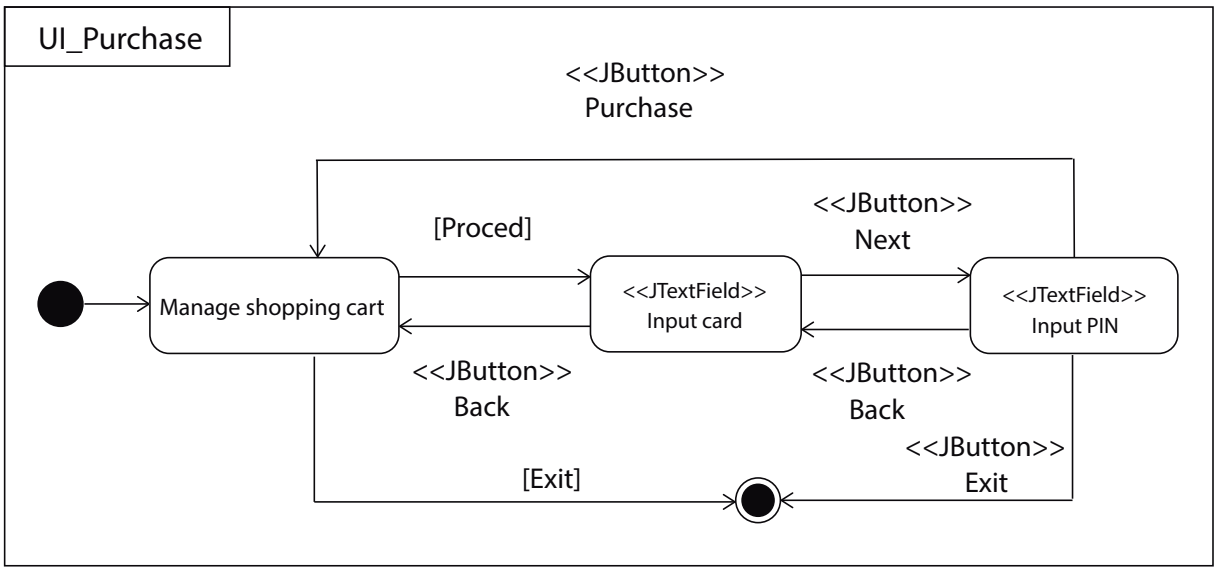

Figura 2.3: Diagrama principal da interação do usuário com uma interface de um sistema de compras – Adaptado de Almendros-Jiménez e Iribarne (2008).

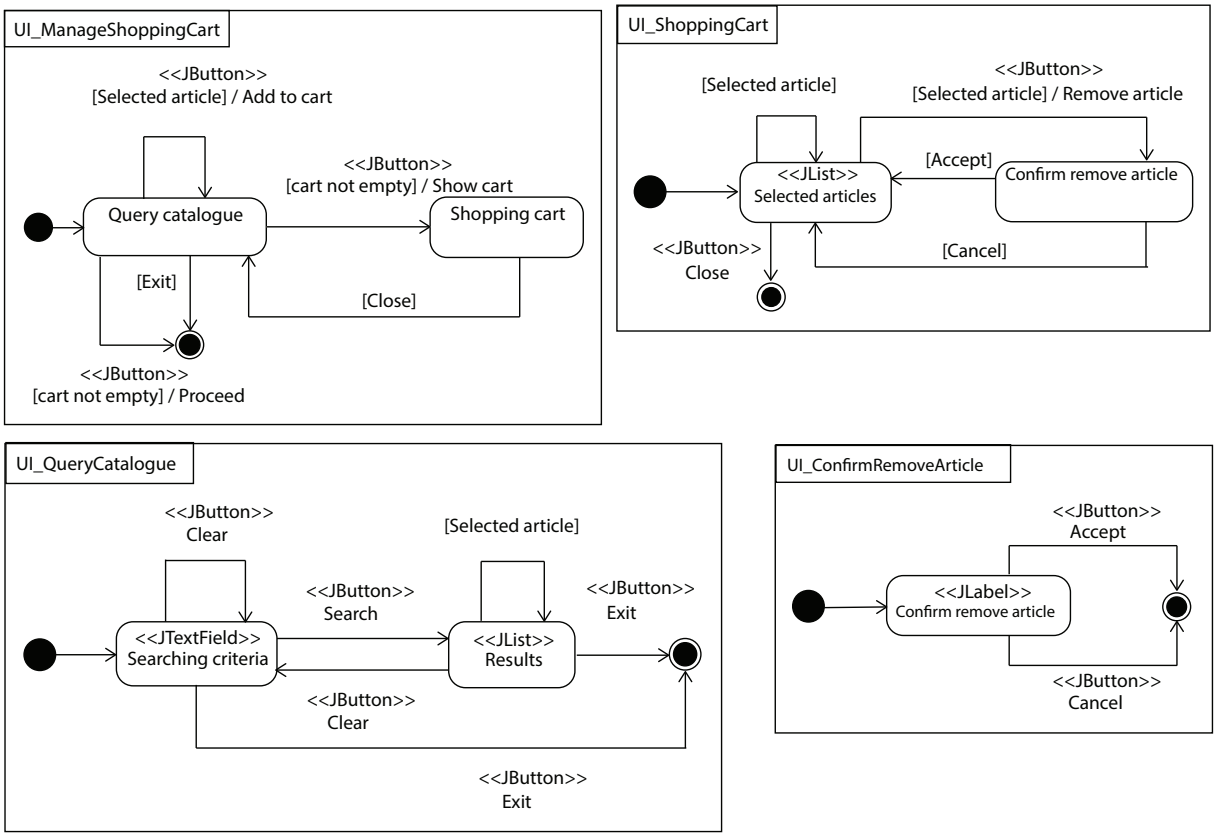

Figura 2.4: Detalhamento do estado *Manage shopping cart* apresentado na Figura 2.3 do sistema de compra – Adaptado de Almendros-Jiménez e Iribarne (2008).

ShoppingCart. No diagrama *UI_ConfirmRemoveArticle* é detalhado o estado *Confirm Remove Article* do diagrama *UI_ShoppingCart*.

Com a descrição dos modelos navegacional, comportamental e de diálogo é possível observar como as informações da interface estão dispersas, sendo repetidas informações de uma interface e de seus componentes em múltiplas representações.

O modo como as informações são apresentadas colabora para o aumento da dificuldade em compreender os modelos de interface. Assim, os modelos navegacionais, comportamentais e de diálogo foram escolhidos para a aplicação do estudo por sua representatividade e pela quantidade de informações que os modelos disponibilizam.

2.2 Criação de modelos

Na Engenharia de Software, os modelos de interface podem ser criados antes do processo de desenvolvimento iniciar (engenharia avante) ou durante e/ou depois a criação da interface (engenharia reversa). Chikofsky et al. (1990) definem a engenharia avante (*forward engineering*) como o processo de desenvolvimento tradicional, que se inicia com um alto nível de abstração de informações até o nível mais baixo de abstração, dando apoio à criação em fases como a especificação e o projeto, por exemplo. A engenharia reversa (*reverse engineering*) é definida por um processo de análise que se inicia nas informações de baixo nível de abstração, criando a representação dessas informações em um nível alto de abstração (Chikofsky et al., 1990, Patel et al., 2007).

Como exemplo de engenharia avante, o Desenvolvimento Dirigido a Modelos (*Model Driven Development*) utiliza ferramentas para a geração de interface de maneira automática. Ou seja, os modelos são utilizados como artefatos para a implementação (geração) da interface (Raneburger et al., 2012, Motti et al., 2013), restrito ao que se descreve nos modelos. Neste trabalho foi utilizada a engenharia reversa e suas técnicas para a criação dos modelos de interface pela facilidade de obtenção dos modelos de interface de um projeto real. Com essa abordagem é possível obter os modelos em qualquer momento do processo de desenvolvimento e não apenas ao final, com a interface finalizada.

Na literatura, a engenharia reversa tem sido utilizada para extrair as informações da interface (Memon, 2007, Amalfitano et al., 2008, Grilo et al., 2010, Aho et al., 2014). Para extrair as informações existem as abordagens estática, dinâmica e híbrida (Demeyer et al., 1999, Systä e Tamperensis, 2000, Salihu e Ibrahim, 2016). A abordagem estática analisa e extrai informações a partir do códigos-fonte ou do código executável. Essa abordagem permite extrair informações da estrutura do software de maneira completa e automática (Systä e Tamperensis, 2000, Grilo et al., 2010). A abordagem dinâmica extrai informações durante a execução do software utilizando a sua interface, sem acesso aos códigos-fontes. Essa abordagem também permite obter informações da estrutura do software e de seu comportamento. De acordo com (Grilo et al., 2010), a criação dos modelos de maneira automática utilizando a abordagem dinâmica é mais difícil, pois na maioria dos casos é necessária uma interação com a interface para obter as informações para os modelos. Neste caso, a abordagem híbrida utiliza tanto a abordagem estática quanto a dinâmica para extrair as informações, não existindo uma ordem de execução definida entre as abordagens (Demeyer et al., 1999, Systä e

Tamperensis, 2000, Salihu e Ibrahim, 2016). Na Figura 2.5 é mostrado um esquema sobre as abordagens mencionadas a partir de um software em análise.

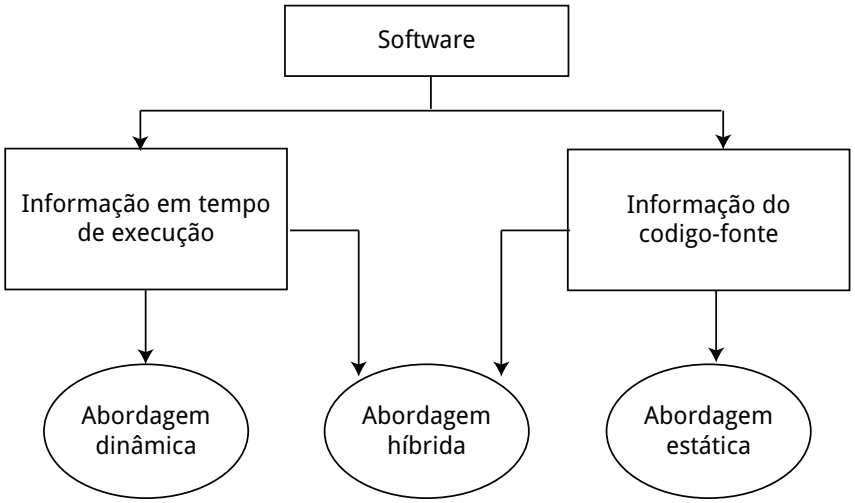

Figura 2.5: Representação da extração das informações utilizando a engenharia reversa, baseado em Systä e Tamperensis (2000)

2.2.1 Técnica de extração estática

Técnicas de extração estática são utilizadas para extrair informações a partir do código-fonte ou do código executável, sobre a estrutura e o comportamento da aplicação em análise (Systä e Tamperensis, 2000, Grilo et al., 2010). Algumas das técnicas utilizadas são:

- **Separação de código** (*code slicing*) (Sasirekha et al., 2011) - trechos extraídos do código-fonte são examinados e analisados para a recuperação de informações do item analisado.
- **Interpretação abstrata** (*abstract interpretation*) (Bouillon e Vanderdonckt, 2002) - o código inteiro é analisado e é construída uma representação abstrata.
- **Programação estratégica** (*strategic programming*) (Lämmel et al., 2002, Lämmel e Visser, 2002) - é uma proposta de linguagem de programação genérica, desenvolvida para aplicações com termos transversais e de sintaxe abstrata.
- **Técnica de agrupamento** (*clustering techniques*) (Michail, 2002) - explora a equivalência de critérios para reconhecer comportamentos equivalentes e, assim, classificá-los.

A técnica de separação do código é uma das técnicas mais utilizadas na literatura, mas todas podem ser usadas em conjunto (Salihu e Ibrahim, 2016). Na literatura, as ferramentas *phpModeler* (Maras et al., 2009) e *GUISurfer* (Silva et al., 2010) utilizam a técnica de separação de códigos. A ferramenta *phpModeler* extrai as informações do software em linguagem PHP, criando diagramas UML de acordo com a

estrutura do software. A ferramenta *GUISurfer*, criada em linguagem Java, analisa o código utilizando uma árvore de sintaxe abstrata (*Abstract Syntax Tree*) para separar o código-fonte relacionado à interface do software.

2.2.2 Técnica de extração dinâmica

De acordo com Morgado et al. (2012), existem dois tipos de técnicas para extrair as informações dinamicamente: abordagem com e sem instrumentação. Instrumentação é uma técnica utilizada para observar e extrair informações em execução após alterações no código da aplicação (Alalfi et al., 2009). Podem ser inseridas no código-fonte informações como, trechos de código, variáveis e chamadas de métodos na aplicação (Kazman et al., 2002, Alalfi et al., 2009). No entanto, as modificações realizadas no código não podem alterar as funcionalidades principais do software e os resultados gerados antes e depois da instrumentação não devem ser diferentes (Alalfi et al., 2009). Neste trabalho, não é realizada a instrumentação nas aplicações *web*, pois o objetivo é apenas extrair as informações da interface sem alterar o código-fonte da interface. Além disso, não havia acesso ao código-fonte dos *sites* escolhidos como artefatos nos experimentos realizados (Capítulo 5).

Existem várias ferramentas que utilizam a técnica de extração dinâmica para a criação de modelos, dentre elas destacam-se a ferramenta *ReGUI* (Morgado et al., 2012) e a *GUIRipper* (Memon et al., 2003). A ferramenta *GUIRipper* extrai informações de comportamento para sistemas em Java, sem utilizar a instrumentação dinâmica. A ferramenta *ReGUI* analisa tanto aplicações *web* e *desktop*, extraindo informações estruturais e de comportamento da interface sem utilizar a instrumentação dinâmica. Inicialmente, a interface que será rastreada deve ser indicada para a ferramenta. A ferramenta realiza a análise do estado inicial da interface selecionada e depois são registradas as interações e as mudanças de estados gerados da interface. Para registrar as informações e a estrutura da interface é utilizado o *framework UI Automation*, criado pela Microsoft, que fornece a estrutura de acessibilidade para a plataforma Windows (Microsoft, 2016).

2.2.3 Abordagem híbrida

Em alguns casos, apenas a utilização da abordagem estática não é suficiente para extração de todas as informações necessárias. Em tecnologias *web*, por exemplo, as conexões entre os eventos e os objetos da interface, de um modelo comportamental, podem ocorrer apenas durante a execução do software (Silva e Campos, 2013). Ou seja, utilizando a abordagem estática é possível determinar todos os eventos que podem ser acionados na interface, mas não quais foram utilizados durante a execução da interface. Por conta disso, são utilizadas abordagens dinâmicas ou híbridas para tecnologias *web*.

A abordagem híbrida utiliza as técnicas de extração estática e dinâmica para extrair o máximo de informações do software em análise. No trabalho de Silva e Campos (2013) o processo para extrair as informações para o modelo é iniciado pela abordagem dinâmica, observando os eventos encontrados durante a execução. Após a abordagem dinâmica, é realizada a abordagem estática, observando as variáveis e as entradas de dados. O modelo criado pela abordagem dinâmica pode ser atualizado após a abordagem estática, se necessário.

Das ferramentas que utilizam a abordagem híbrida para extração de informações são destacadas a *Ware* (Lucca et al., 2004) e a *PHP2XMI* (Alalfi et al., 2009). A ferramenta *Ware* utiliza primeiro a técnica de separação de código para extrair as informações relacionadas tanto ao código-fonte quanto ao código HTML. Depois, baseados no resultado da análise estática, a análise dinâmica sem instrumentação é realizada. Por meio da ferramenta *Ware* são criados diagramas de classe, de sequência, de colaboração e caso de uso.

A ferramenta *PHP2XMI* extrai as informações do comportamento de sistemas *web*, que utilizam a linguagem de programação PHP. Primeiro são extraídas informações usando a técnica de separação de código, posteriormente é feita a análise dinâmica com instrumentação para extrair informações sobre as permissões e comportamentos de uso dos usuários.

2.3 Técnicas de extração para interfaces *web*

De acordo com Amalfitano et al. (2010d), podem ser utilizadas duas técnicas para extrair as informações de uma interface: por sessões com o usuário e por uma técnica de *crawling*. A seguir são apresentadas definições e ferramentas selecionadas para ambas.

2.3.1 Sessões com usuários

A utilização de sessões com usuários produz rastros reais da execução da interface. Dentre as ferramentas que realizam o rastreio das informações do usuário destacadam-se as ferramentas *FireDetective* (Zaidman et al., 2013), *CReRIA* (Amalfitano et al., 2010c) e *ReAJAX* (Marchetto et al., 2012).

A ferramenta *FireDetective* é uma extensão do navegador *Mozilla Firefox*, que extrai as informações dos eventos da interface. A utilização da interface é registrada analisando o código *JavaScript* e as interações do lado do cliente e do lado do servidor da aplicação. A ferramenta grava os nomes das requisições de funções e métodos feitos pela interação do usuário e reconstrói posteriormente uma árvore com a ordem dessas requisições feitas ao sistema. A ferramenta funciona para aplicações Ajax, com servidor *Java EE Web Server*, mas Zaidman et al. (2013) afirmam que é possível expandir a aplicação para outras linguagens de programação *web* como o PHP.

A ferramenta *CReRIA* utiliza uma máquina de estados finitos para representar o comportamento da interface. A ferramenta oferece um navegador *web* integrado, criado com o código-fonte do navegador *Mozilla Firefox*, que permite ao usuário navegar pela interface. As informações da interface como o tipo de evento disparado (*click, mouseover, mousedown*, etc.) e os dados colocados na interface, são registradas em uma base de dados. A ferramenta utiliza heurísticas de clusterização para avaliar se as interfaces navegadas são iguais e agrupá-las no modelo criado.

A ferramenta *ReAJAX* extrai as informações dos eventos e estados da interface utilizando a máquina de estados finitos como representação visual. Ela é uma extensão do navegador *Mozilla Firefox*, que adiciona um fragmento de *script* de código em tempo de execução para cada página *web* visitada para rastrear as informações da interface. Com a extensão são rastreados os eventos ativados, o conteúdo da página e a estrutura DOM (estados da interface). Em seu trabalho Marchetto et al. (2012) realizam uma comparação entre a ferramenta *ReAJAX* e a *CReRIA*. Das diferenças entre as ferramentas destaca-se a customização do modelo gerado. A *ReAJAX* possui uma configuração padrão da criação da representação, mas é possível customizar as informações para documentação, testes, entre outros. Para utilizar a ferramenta *CReRIA* é necessário analisar a saída a ser produzida e os critérios para criação dos *clusters* obrigatoriamente.

2.3.2 Técnicas de *crawling*

Crawling é um processo de exploração de um sistema *web* para descobrir as páginas existentes (Choudhary et al., 2012). De acordo com Choudhary et al. (2012), a utilização dessa técnica possui duas motivações principais, que são o rastreamento para indexação do conteúdo *web* e o rastreamento para testes automatizados em busca de vulnerabilidades de segurança, problemas de acessibilidade e outros. A base do rastreamento é a localização de todos os endereços (URLs) disponíveis na aplicação *web* a partir de um endereço inicial. Após descobrir uma nova página, ela pode ser analisada de acordo com o objetivo do rastreamento.

Choudhary et al. (2012) definem como estratégia de rastreamento a decisão do algoritmo sobre o caminho que deve ser seguido durante o rastreamento. Choudhary et al. (2012) descrevem duas estratégias de rastreamento (Dincturk et al., 2014):

1. Em largura (*Breadth-First*): explora o estado descoberto menos recentemente pela primeira vez.

2. Em profundidade (*Depth-First*): explora todo estado descoberto mais recentemente.

Na Figura 2.6 é mostrada uma demonstração das estratégias de rastreamento, sendo a estratégia em profundidade é mostrada do lado esquerdo (a) e a estratégia em largura é mostrada do lado direito (b) da figura.

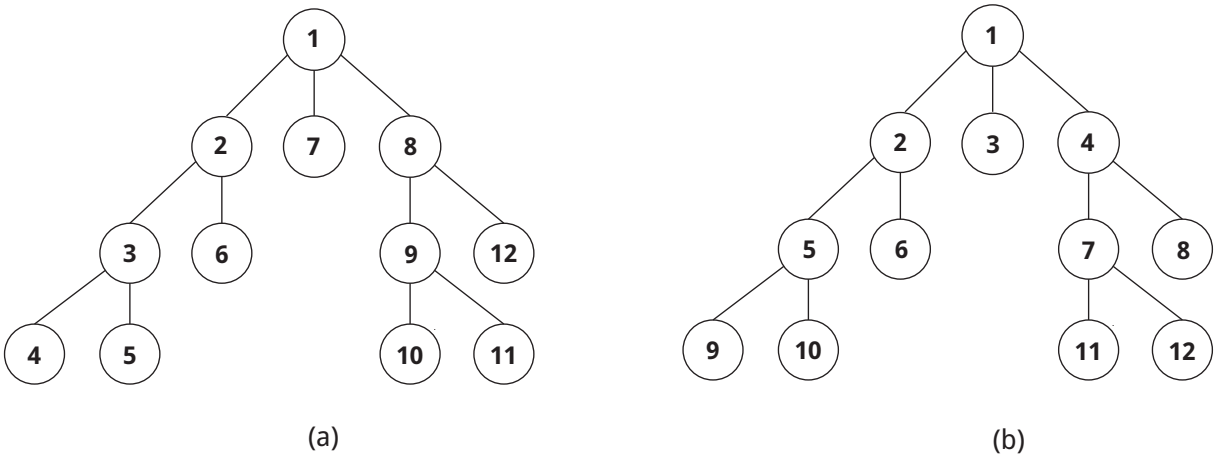

Figura 2.6: Representação das estratégias de rastreamento em profundidade (lado esquerdo - (a)) e em largura (lado direito - (b))

Algumas das ferramentas que aplicam a técnica de *crawling* são a *Crawljax* (Mesbah et al., 2012) e a *CrawlRIA* (Amalfitano et al., 2010d). A ferramenta *Crawljax* utiliza a técnica em profundidade para buscar os estados da interface, representados por um grafo de fluxo. De acordo com Dincturk et al. (2014), utilizando essa estratégia apenas os eventos associados a um elemento DOM, que são diferentes do estado anterior, serão explorados. Esta estratégia pode não achar todos os estados, já que a execução de um evento fixo de diferentes estados pode levar a diferentes estados. Para considerar os estados iguais ou diferentes dos já explorados é utilizada uma distância de edição (método Levenshtein (Levenshtein, 1966)) para analisar se o estado atual é igual a outro já explorado. A ferramenta permite a customização das informações geradas para fins de documentação, aplicação de testes ou compreensão da interface.

A ferramenta *CrawlRIA* também utiliza a estratégia em profundidade para buscar os estados da interface. A abordagem adotada por essa ferramenta é similar à da *Crawljax*, com a diferença de que o rastreamento dos eventos da interface gera rastros de execução. Ou seja, a partir do estado inicial, os eventos são rastreados até o evento anterior ao objeto visitado atualmente. A partir desse ponto são salvos os rastros de execução do usuário em um banco de dados. Essa ferramenta utiliza a mesma técnica de criação de rastros de execução e a mesma representação visual utilizada por *CReRIA*.

2.4 Ferramentas relacionadas à compreensão da interface do usuário

As ferramentas apresentas neste capítulo são dividi-las em duas categorias: relacionadas aos testes de interface, relacionadas à compreensão da interface e pertencentes a ambas categorias. Dentro dessas categorias, as ferramentas se distinguem pelas técnicas aplicadas para a extração das informações da interface (Seção 2.2) e pelo ambiente na qual elas realizam a análise da interface (*desktop* ou *web*).

As ferramentas relacionadas aos testes de interface utilizam os modelos de interface como oráculos para criação de testes. No entanto, existe uma lacuna com relação a validação dos modelos de interfaces na literatura, como descrita na seção a seguir (Seção 2.5).

Na literatura existem várias ferramentas relacionadas à compreensão da estrutura da interface, como *WANDA* (Antoniol et al., 2004), *ReAJAX* (Marchetto et al., 2012), *FireDetective* (Zaidman et al., 2013) e *WAVI* (Cloutier et al., 2016). A ferramenta *DynaRIa* (Amalfitano et al., 2010b) é classificada como pertencente a ambas categorias de ferramentas. A maioria dessas ferramentas utilizam os diagramas da UML para representar as classes existentes na interface (Antoniol et al., 2004), as sequências de interações e o comportamento da interface (Amalfitano et al., 2010b). No entanto, essas representações possuem uma alta dispersão das informações e vários níveis de abstrações, dificultando a compreensão dos modelos criados.

Algumas dessas ferramentas utilizam representações customizadas para apresentar as informações da interface, como no trabalho de Marchetto et al. (2012). A ferramenta *ReAJAX* utiliza uma máquina de estados customizada para apresentar as informações dos estados com mais detalhes, como apresentada na Figura 2.7. Neste caso, não é possível afirmar que a incorporação de mais informações a um mesmo modelo colabora para seu entendimento e para a compreensão da estrutura da interface. Além disso, nos trabalhos destacados não foram apresentados experimentos controlados para avaliar o nível de compreensão oferecido pelas representações utilizadas. Foram utilizados apenas estudos de casos.

As ferramentas *FireDetective* (Zaidman et al., 2013, Matthijssen et al., 2010) e *WAVI* Cloutier et al. (2016) apresentam informações relacionadas à arquitetura da interface. A ferramenta *FireDetective* utiliza uma abordagem dinâmica e a *WAVI* utiliza uma abordagem estática para identificação das chamadas entre os métodos e das funções *JavaScript*, sem relacionar com os componentes existentes na interface do usuário. Desses trabalhos, apenas o da ferramenta *FireDetective* (Matthijssen et al., 2010) apresenta os resultados da utilização da ferramenta utilizando um experimento controlado. Como apresentado nesse trabalho, ferramentas como a *FireDetective* colaboram para a compreensão e, por consequência, para o processo de manutenção da interface, apresentando o estado atual da interface. Porém, essas ferramentas utilizam uma pequena parcela das informações existentes em uma interface para serem representadas.

2.5 Validação de modelos

Akpan e Brooks (2014) afirmam que a validação de modelo é um processo para avaliar se o modelo é uma representação precisa do sistema real. O processo de validação deve ser feito antes da utilização do modelo, para garantir que as informações

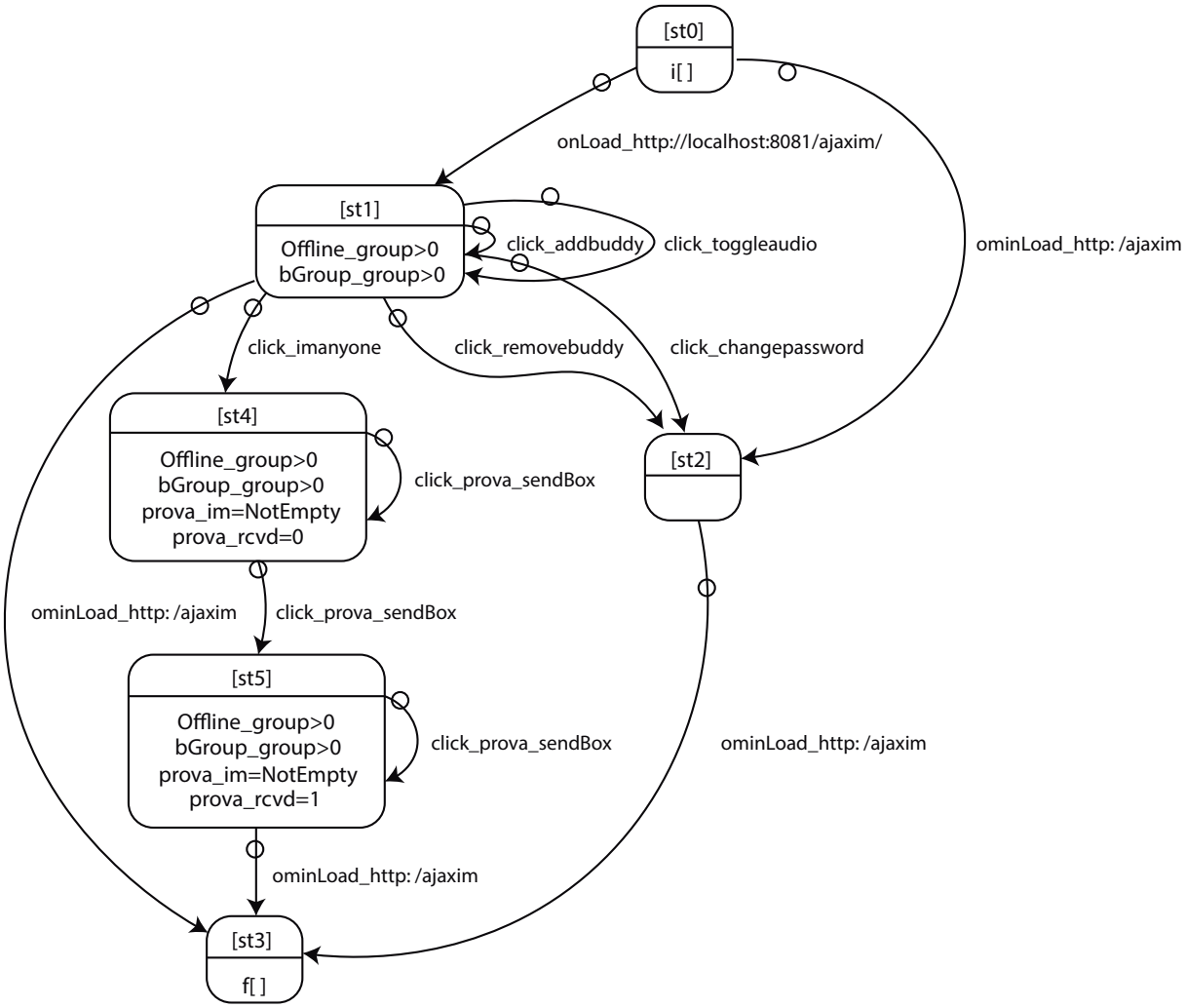

Figura 2.7: Representação customizada da máquina de estados da interface de *Tudu*, apresentada por Marchetto et al. (2012), que mostra as propriedades de cada estado de maneira detalhada.

fornecidas por ele estão corretas (Arcaini et al., 2014).

Hurtado et al. (2015) afirmam que existem 3 passos a serem realizados para a validação de modelos:

1. Verificação do modelo: verifica se o modelo não contém erros relacionados à consistência da sua dimensão, sintaxe e semântica.

2. Validação da estrutura: essa validação tem como objetivo verificar a estrutura do modelo, comparando com a estrutura atual do sistema.

3. Validação do comportamento: mede qual a precisão do modelo para reproduzir os padrões de comportamento do sistema.

Leopold et al. (2014) mencionam em seu trabalho abordagens para realizar a validação de modelos:

- Prototipação: utiliza a implementação do modelo para obter avaliações o mais breve possível.

- Abstração e filtragem: reduz a informação a ser passada para o usuário, utilizando abstrações do modelo. Cenários também podem ser criados para filtrar as informações.
- Visualização da especificação: utiliza a visualização de cenários com modelos gráficos ou até animação dos cenários.
- Checagem de propriedades: o objetivo é comparar o modelo com a especificação formal das suas propriedades.
- Geração de linguagens naturais: a partir de modelos a linguagem é gerada para facilitar a comunicação entre especialistas e analistas do sistema.

Na maioria das abordagens citadas por Leopold et al. (2014) e em outros artigos sobre a validação de modelos (Kof et al., 2010, Egyed, 2011, Lalioti e Loucopoulos, 1993) os modelos criados a partir de artefatos da engenharia avante são comparados com os modelos criados a partir do sistema desenvolvido, identificando inconsistências. Porém, em trabalhos que utilizam a engenharia reversa para a criação dos modelos (Grilo et al., 2010, Chuang et al., 2011, Aho et al., 2011) não existem artefatos para a comparação com os modelos gerados, ou esses artefatos não são citados nesses trabalhos.

No trabalho de Bellettini et al. (2007), a validação de modelos utiliza a engenharia reversa como método de extração de informações. Nesse trabalho são analisados modelos UML criados a partir de sistemas *web*, utilizando a abordagem híbrida para criação dos modelos. Primeiro é feita a abordagem dinâmica para definir os caminhos que a abordagem estática percorre baseada na navegação do usuário. Depois é feita a abordagem estática com a técnica de separação de código, nos caminhos selecionados. A abordagem dinâmica utiliza a análise de mutação e são feitas outras verificações para que nenhum problema de navegação interfira na criação do modelo, por exemplo verificar se existem *links* quebrados.

Prévio a este trabalho foi realizado um estudo sobre a validação de modelos de interface. No trabalho de Martins e Garcia (2015) foi apresentado que a maioria dos trabalhos selecionados não realizam a validação dos seus modelos de interface. Fora do contexto de interface, quando realizada a validação do modelo é feita de maneira manual, utilizando uma representação gráfica conhecida pelo validador (Guglielmo et al., 2011) ou são fornecidos treinamentos para familiarização com o modelo (Chiappini et al., 2010).

2.6 Considerações finais

Neste capítulo foram apresentados os tipos de modelos de interface existentes na literatura (Seção 2.1). Os modelos de navegação, de diálogo e de comportamento foram mostrados com mais detalhes por possuírem as informações mais relevantes da interface e identificarem as informações a serem extraídas da interface do usuário.

Na Seção 2.2 foram descritas a engenharia avante e a reversa, como as possíveis abordagens utilizadas para a criação de modelos. Neste trabalho foi utilizada a engenharia reversa como abordagem para a descrição dos modelos de interface. Ela foi definida por disponibilizar dados suficientes para o estudo, permitindo que sejam extraídos as informações da interface desenvolvida.

Dentre as técnicas dinâmicas existentes para o ambiente *web* foram destacadas as técnicas de *crawling* (Seção 2.3.2) e de sessões com usuários (Sessão 2.3.1) por serem referenciadas na literatura para a extração dinâmica de informações de interfaces de aplicações *web*. Ambas técnicas foram utilizadas neste trabalho, sendo descrita sua aplicação no Capítulo 4. No entanto, nenhuma das ferramentas citadas foram utilizadas para extração dos dados da interface, sendo necessária a criação de uma ferramenta própria descrita no Capítulo 4. O desenvolvimento dessa ferramenta foi necessário pela maioria das ferramentas não funcionarem para versões atuais dos navegadores ou não exportam esses dados para a utilização por outras ferramentas.

A análise das informações referentes a cada um dos modelos de interface, utilizados neste trabalho, permite identificar as melhores técnicas de visualização para cada modelo. No capítulo seguinte são descritas as técnicas de visualização selecionadas para esse trabalho.

CAPÍTULO 3

Visualização de Informação

De acordo com Mazza (2009), a Visualização de Informação (VisInfo) provê métodos, técnicas e ferramentas para organização e representação visuais de dados, com a finalidade de evidenciar informações neles presentes. De acordo com Card et al. (1999), a VisInfo utiliza representações visuais de dados abstratos para ampliar a cognição, expressando semanticamente conexões entre dados. São utilizadas técnicas que definem as regras de organização dos dados para a criação dessas representações.

As representações visuais possibilitam estruturar, observar e criar ou confirmar as relações entre dados, sem necessariamente realizar algum tipo de cálculo. No entanto, os dados precisam passar por um mapeamento visual para serem transformados nas representações visuais. Na Figura 3.1 são apresentados os processos de mapeamento visual das representações apresentado por Card et al. (1999). As setas dos *Dados brutos* até o usuário indicam as transformações realizadas sobre um conjunto de dados. As setas voltadas para as próprias atividades significam adaptações necessárias nas transformações. Na *Transformação dos dados* os *Dados brutos* são mapeados transformados em *Tabelas de dados*. No *Mapeamento visual* as *Tabelas de dados* são transformados em *Estruturas visuais*, relacionando os dados e suas características em como eles devem ser apresentados visualmente. No último passo do processo, são aplicadas técnicas de visualização nas *Estruturas visuais* para apresentar as *Visões* para o usuário, que interage com algumas tarefas.

Esse processo de mapeamento visual pode ser aplicado para diversos tipos de dados em várias áreas. Uma das áreas de aplicação da VisInfo conhecida na literatura é relacionada ao processo de desenvolvimento de software, a Visualização de Software (VisSoft). Neste capítulo é apresentada a VisSoft (Seção 3.1) e algumas técnicas de VisInfo utilizadas na Visualização de Software (Seção 3.2), com um destaque para

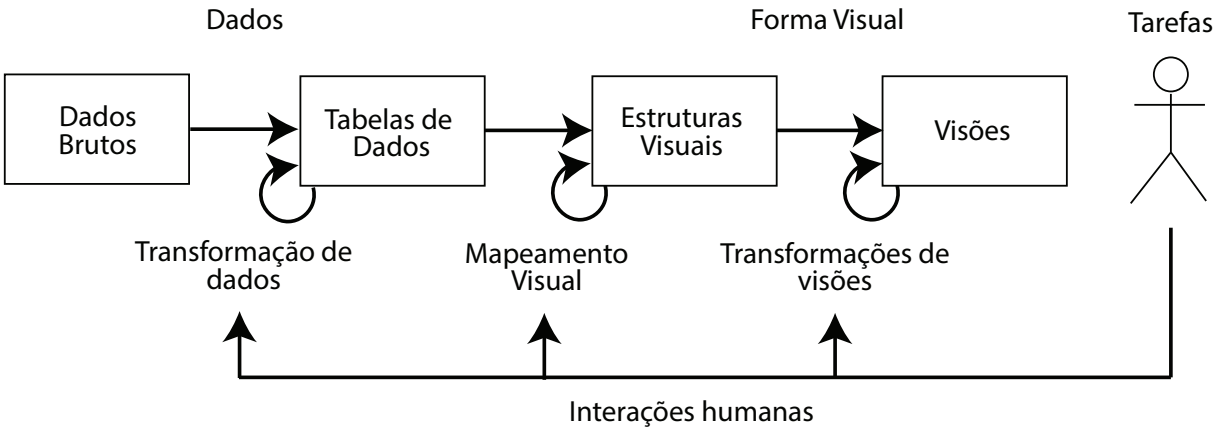

Figura 3.1: Modelo de mapeamento visual adaptado de Card et al. (1999).

as representações relacionadas à interface do usuário (Seção 3.1.1), que é o foco desse estudo. Também são apresentadas algumas técnicas de interação (Seção 3.3) e técnicas de coordenação (Seção 3.4) aplicadas na VisSoft. Na Seção 3.5 são apresentados estudos relacionados ao processo de validação das técnicas de VisInfo.

3.1 Visualização de software

A Visualização de Software é uma disciplina que faz uso de atributos visuais, como forma e cores, para facilitar a compreensão de um sistema, seus artefatos e outros dados relacionados ao processo de desenvolvimento de software (Knight e Munro, 1999, Diehl, 2007). Por outro lado, de acordo Diehl (2007), a Visualização da Informação tem por objetivo mostrar as informações de maneira abstrata, usando uma maneira natural e intuitiva para facilitar a compreensão dos dados. Por conta disso, a VisSoft pode ser considerada como uma aplicação das técnicas da Visualização da Informação (Gallagher et al., 2008), voltadas para a representação e a compreensão de dados do software.

Na VisSoft são utilizadas técnicas de VisInfo para a representação de aspectos estáticos, dinâmicos e de evolução do software (Caserta e Zendra, 2011). Os aspectos estáticos do software são relacionados às partes que podem ser analisadas sem a execução do software, como código e arquitetura. Os aspectos dinâmicos do software são relacionados às informações que podem ser analisadas durante a execução do software, como a chamada de funções ou a comunicação entre objetos (Diehl, 2007, Caserta e Zendra, 2011). A evolução do software é relacionada ao processo de desenvolvimento do software, com foco nas mudanças que acontecem no software com relação ao tempo (Diehl, 2007).

3.1.1 Representação de informações da interface do usuário

Na literatura existem muitas ferramentas relacionadas à interface do usuário para a

identificação de chamadas de funções, análise do comportamento da interface, entre outras funções. Mas a maioria desses trabalhos utilizam apenas grafos ou máquinas de estados para representar essas informações (Amalfitano et al., 2010b, Silva et al., 2010, Mesbah et al., 2012, Cloutier et al., 2016). O trabalho de Cloutier et al. (2016), por exemplo, utiliza o diagrama de força direta (grafos) para representar as chamadas entre as funções *JavaScript* da interface do usuário.

Existem trabalhos que utilizam técnicas de VisInfo para apresentar informações relacionadas à estrutura da interface e do uso da aplicação *web*. O trabalho de Santos et al. (2004) utiliza técnicas de visualização para apresentar informações estruturais e de uso de uma aplicação *web*, para apoiar a avaliação de usabilidade. O trabalho de Rooke et al. (2011) se diferencia dos demais por utilizar a visão de um mapa mundial para representar a organização dos ícones de uma ferramenta e como os ícones se relacionam, de acordo com a análise de interação do usuário. No entanto, esses trabalhos não fazem referência aos modelos de interface do usuário.

Nas próximas seções são mostradas as técnicas de VisInfo que são utilizadas para a VisSoft (Diehl, 2007, Caserta e Zendra, 2011).

3.2 Técnicas de Visualização de Informação

Na literatura, existe um grande número de técnicas de VisInfo que podem ser aplicadas para diferentes tipos de dados. De acordo com Keim e Kriegel (1996) e Keim (2002) as técnicas de visualização podem ser classificadas em geométricas, iconográficas, orientadas a *pixel*, hierárquicas e baseada em grafos. Dentre essas técnicas, as mais usadas na VisSoft são as técnicas hierárquicas e baseadas em grafos, que são apresentadas nas próximas seções.

3.2.1 Técnicas baseadas em grafos

Grafos são estruturas utilizadas para descrever a relação entre objetos, sendo os objetos representados por nós e a relação entre os objetos é representada por arestas (Diehl, 2007). Existem diferentes tipos de grafos para diferentes tipos dados visualizados. Uma árvore, por exemplo, é um dos muitos tipos de grafos existentes, sendo um grafo conectado, não ponderado e acíclico (Ward et al., 2010). Na Figura 3.2 são apresentados três tipos de grafos: força direcionada, ortogonal e hierárquico. No grafo de força direcionada são aplicadas forças entre os nós. No grafo ortogonal a quantidade de curvas são minimizadas, no máximo uma curva por arestas, exceto para bordas reflexivas. No grafo hierárquico existe uma relação de precedência nos nós com base na direção das arestas (Dogrusoz et al., 2002).

Grafos possuem algumas propriedades úteis à apresentação de característica de programas (Diehl, 2007) como representar a conexão entre diferentes objetos, sentido de algo, entre outras. Algumas delas são mostradas nas figuras a seguir. Na Figura 3.3

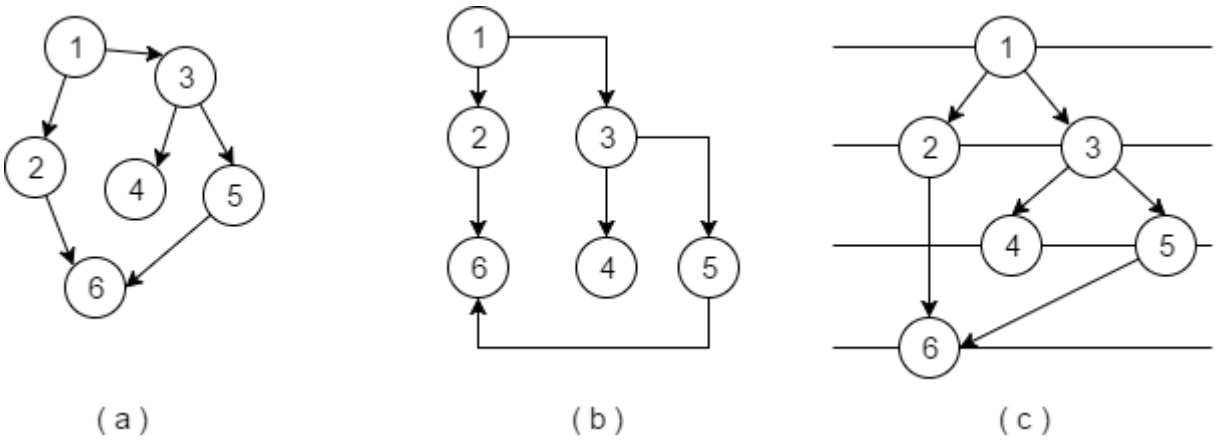

Figura 3.2: Exemplos dos tipos de grafos, em que (a) é um grafo de força direcionada, (b) é um grafo ortogonal e (c) é um grafo hierárquico.

do lado esquerdo (a) é um exemplo de um grafo acíclico e do lado direito (b) é um exemplo de um grafo cíclico. Na Figura 3.4 do lado esquerdo (a) é um exemplo de grafo não-conectado e do lado direito (b) é um exemplo de um grafo. Na Figura 3.5, lado esquerdo (a) é um exemplo de um grafo não-dirigido e do lado direito (b) é um exemplo de um grafo dirigido.

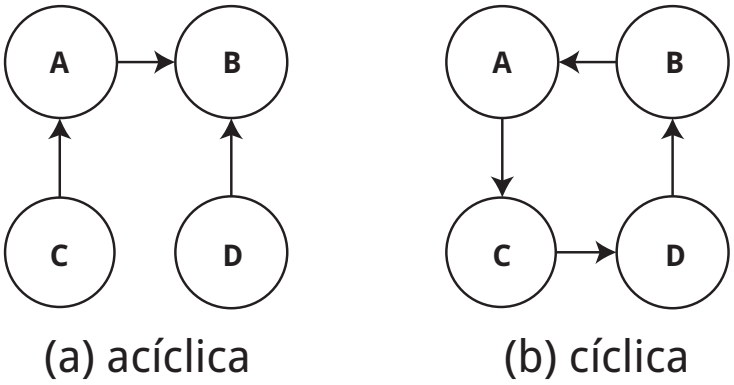

Figura 3.3: Exemplos de grafo acíclico (a) e cíclico (b)

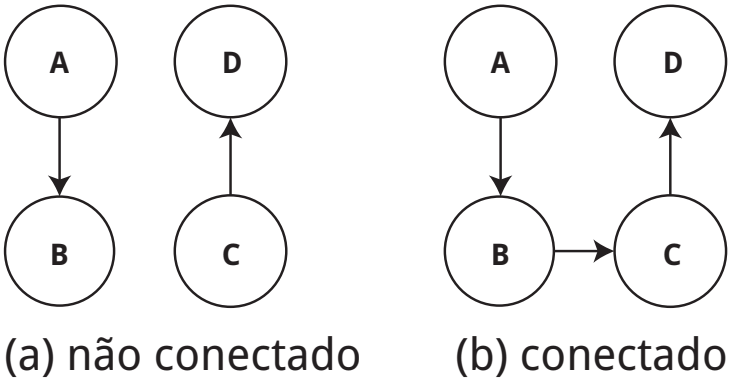

Figura 3.4: Exemplos de grafo não-conectado (a) e conectado (b)

Além dessas características, alguns autores (Beck et al., 2014, Burch et al., 2011, Diehl, 2007) classificam o grafo como estático ou dinâmico. De acordo com Beck et

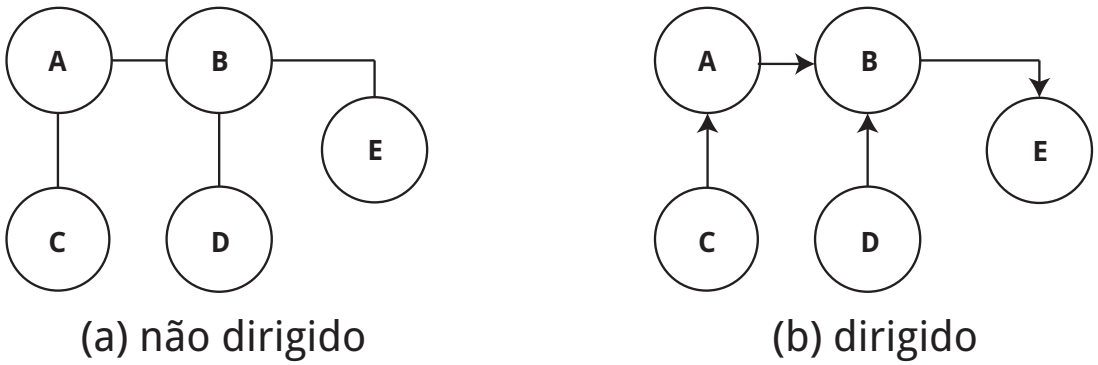

Figura 3.5: Exemplos de grafo não-dirigido (a) e dirigido (b)

al. (2014), a principal característica que diferencia o grafo estático do grafo dinâmico é que a estrutura de nós e suas conexões mudam ao longo do tempo. Para representar a evolução das informações em um grafo dinâmico, podem ser utilizadas animações com a sequência de mudanças.

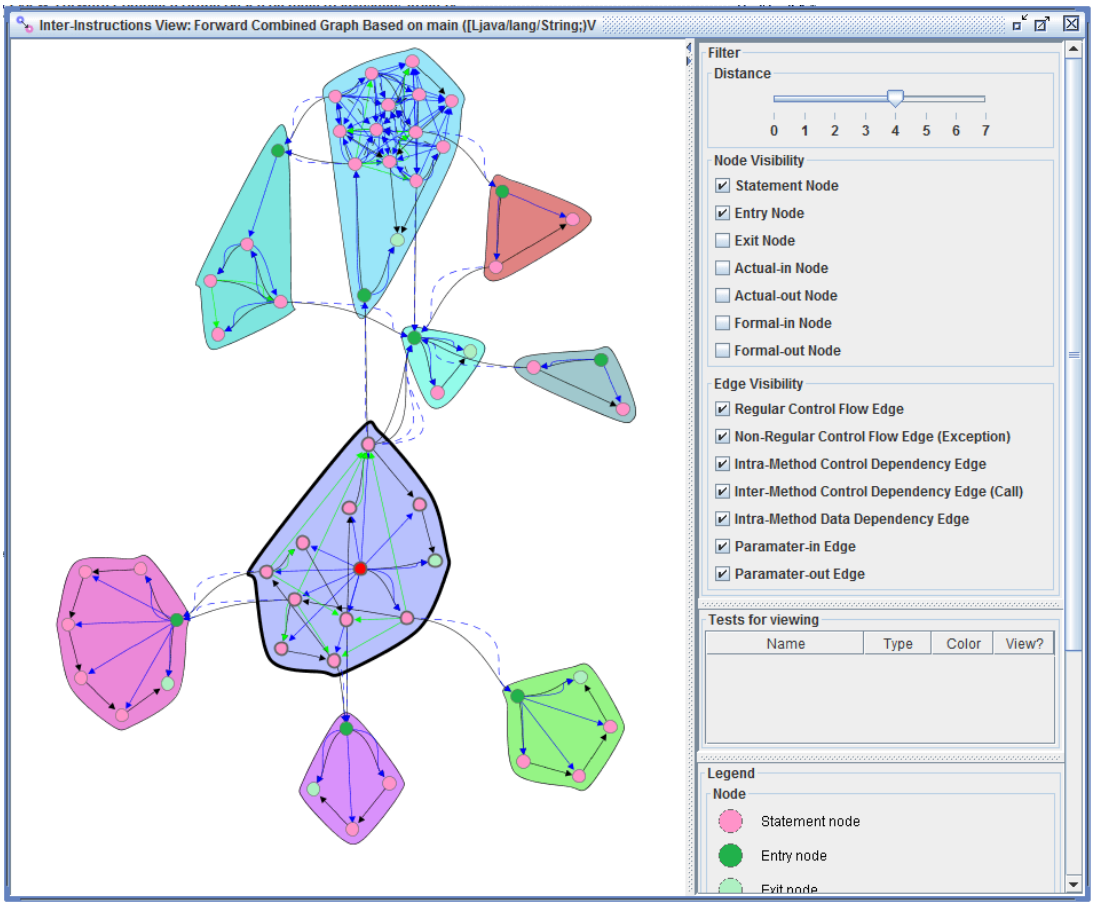

Figura 3.6: Grafo com a visão inter-métodos do método *main* do programa *Health Watcher* - (Delfim e Garcia, 2014)

Na Engenharia de Software, os grafos permitem representar as relações entre componentes de um software. Um exemplo dessa representação aplicada à análise de programas pode ser observado na Figura 3.6 e na Figura 3.7 criadas pela ferramenta $SoftVis_{4CA}$ (Delfim, 2013, Delfim e Garcia, 2014). Na Figura 3.6 é apresentada a visão

de um grafo combinado (inter-método) do método *main* do programa *Health Watcher*, desenvolvido em linguagem Java. Na Figura 3.7 é apresentada a visão intra-método do método *main* do programa *elevator*, que realiza simulação de um elevador para demostração, também desenvolvido em linguagem Java. Na Figura 3.7 as instruções de um método são representadas por nós e a área em torno dos nós representa o método – agregação de nós (área em amarelo). São representados nós físicos (em rosa) – que são instruções do programa – e nós virtuais – que representam as entradas de métodos (em verde escuro) e as saídas de métodos (em verde claro). As arestas representam fluxos de controle (linhas contínuas em preto), dependência de controle (linhas contínuas azuis) ou dependência de dados (linhas contínuas em verde) (Delfim, 2013, Delfim e Garcia, 2014). Para gerar esse exemplo foi feita uma análise do código-fonte de modo a gerar fatias (*program slices*) e, depois, gerar a representação em grafo.

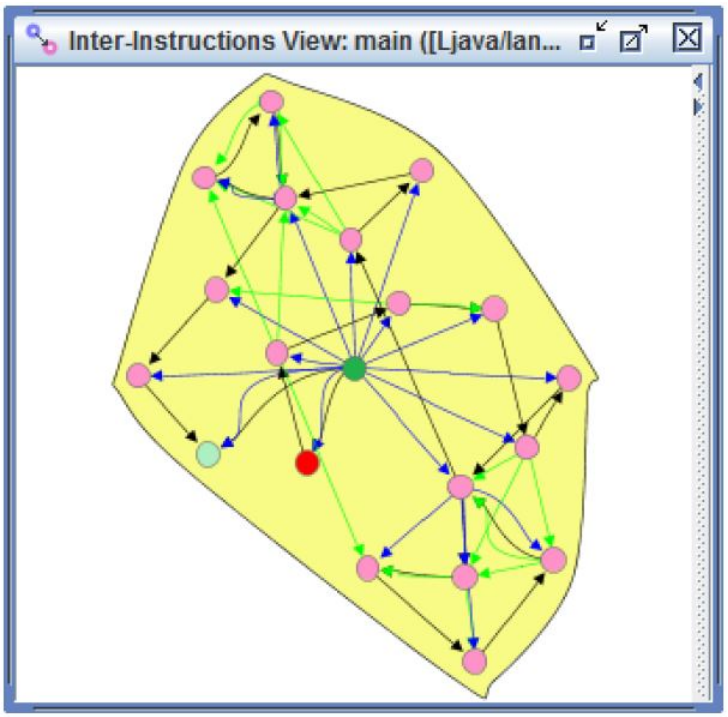

Figura 3.7: Grafo intra-método do método *main* do programa *elevator* - (Delfim, 2013)

3.2.2 Técnicas hierárquicas

De acordo com Ward et al. (2010), técnicas hierárquicas podem ser divididas em duas classes, considerando os algoritmos: relacionadas e não-relacionadas ao preenchimento de espaços (*space-filling*) e (*non-space-filling*), respectivamente.

As técnicas relacionadas ao preenchimento de espaço têm por objetivo maximizar o uso do espaço utilizado para mostrar informações. Um exemplo dessas visualizações são os *Treemaps*.

De acordo com Chen et al. (2007), *Treemaps* podem ser definidos como partições

recursivas de espaço em formato de quadriláteros. A ideia principal é dividir recursivamente um quadrilátero 2D do mesmo modo que a árvore é particionada. Um quadrilátero é utilizado como base e subdividido em quadriláteros menores de acordo com o peso dado para cada nó da árvore (Card et al., 1999). O peso de cada nó é determinado pelas seguintes propriedades (Johnson e Shneiderman, 1991):

- Se um nó, de nome No_1, é antecessor de um nó, de nome No_2, então o quadrilátero do No_1 envolve completamente, ou é igual, o quadrilátero do No_2.
- Os quadriláteros de dois nós se interseccionam se um nó é ancestral do outro.
- Nós ocupam a área do quadrilátero estritamente proporcional ao seu peso.
- O peso de um nó é maior ou igual a soma dos pesos dos seus filhos.

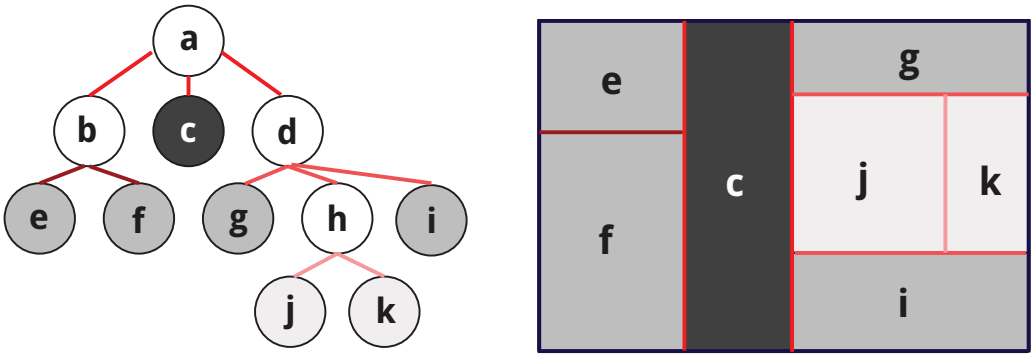

Figura 3.8: Representação de informação em árvore (à esquerda) e seu *Treemap* (à direita) – Adaptado de Caserta e Zendra (2011)

Na Figura 3.8 do lado esquerdo existe a imagem de uma árvore com sua estrutura de nós e conexões. Do lado direito existe o *Treemaps* representando os nós-folha da árvore à esquerda.

As técnicas não relacionadas ao preenchimento de espaço apresentam as informações sem o objetivo de maximização de espaço, como a *Cone Tree*. A *Cone Tree* (Robertson et al., 1991) é uma árvore 3D, onde é possível rotacionar e mover os nós para frente e para trás. Com a sombra projetada da árvore e o uso de transparências nos nós é possível observar sua estrutura e densidade dos nós (Diehl, 2007, Mazza, 2009). Na Figura 3.9 é apresentado um exemplo da estrutura da *Cone Tree*.

A representação da estrutura de árvore, também é conhecida como diagrama nó - link (*node-link diagram*), é uma das mais comummente utilizadas na literatura (Ward et al., 2010). Ward et al. (2010) explicam que o desenho da árvore é influenciado por dois fatores:

1. Largura - relacionado ao número de irmãos que um nó pai pode ter.

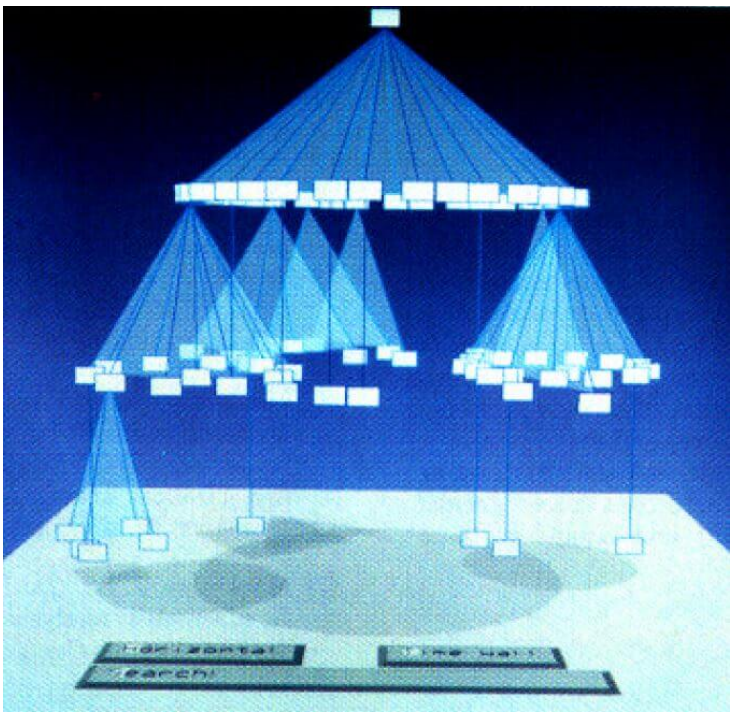

Figura 3.9: Imagem ilustrativa de uma representação da *Cone Tree* - (Robertson et al., 1991)

2. Profundidade - relacionado à distância da raiz ao último nó folha.

Existem várias propriedades visuais – como, cor (saturação e brilho), textura, forma e borda – que podem ser utilizadas para mostrar determinadas propriedades – atributos de dados – de um nó da árvore. De acordo com Johnson e Shneiderman (1991), a cor é uma das propriedades visuais mais utilizadas, por ser de fácil identificação.

Um exemplo de representação estrutural utilizando *Treemaps* é mostrado na Figura 3.10 (Delfim, 2013). Nessa figura é mostrada a visão estrutural do programa *elevator*, sendo as unidades do programa representadas por quadriláteros aninhados. O quadrilátero externo representa o programa todo (raiz) e os quadriláteros internos representam pacotes, classes, métodos e casos de testes. Os retângulos são coloridos com um gradiente linear de cinza, sendo utilizado o cinza escuro para representar o programa todo. Cada retângulo quadrilátero folha representa um método e o seu tamanho é proporcional a quantidade de instruções *bytecode* (Delfim, 2013).

3.2.2.1 Técnica hierárquica de agrupamento de linhas

Na literatura existem técnicas que complementam as técnicas hierárquicas existentes, como a técnica de agrupamento de linhas (*Hierarchical Edge Bundles*) (Holten, 2006). Essa técnica pode ser aplicável em conjunto com outras técnicas hierárquicas, para apresentar a relação entre os nós e a força dessa relação. Na representação podem ser utilizadas cores para indicar o ponto de saída, em verde, e o destino, em vermelho, com relação a um nó.

Na Figura 3.11 é mostrada a representação das dependência entre as classe de um software utilizando a técnica hierárquica de agrupamento de linhas (*Hierarchical*

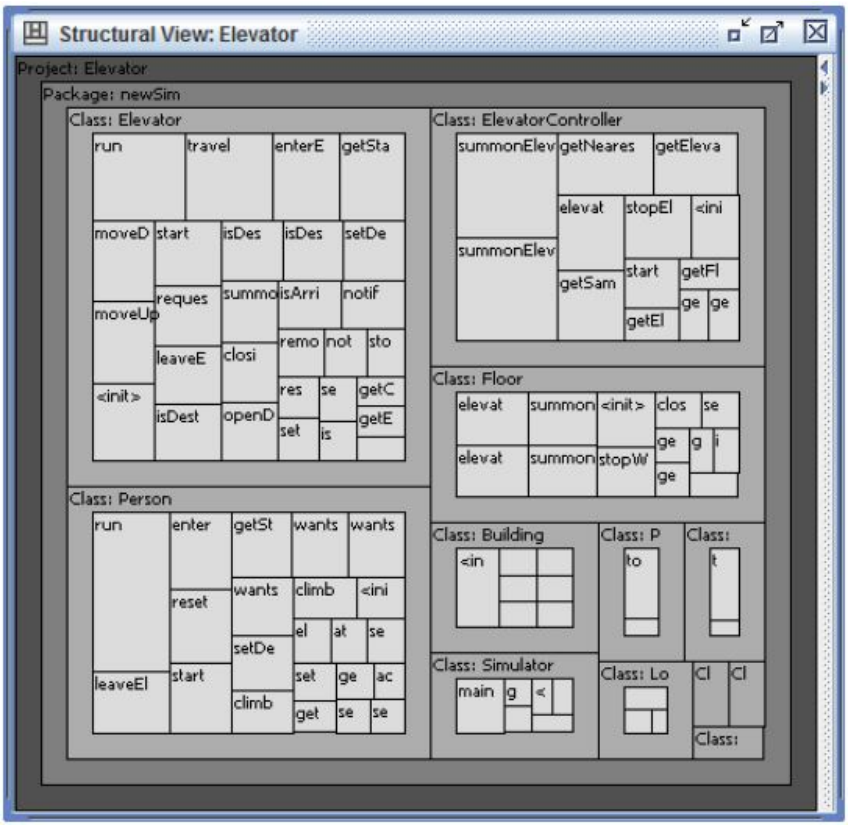

Figura 3.10: Representação da estrutura do programa *elevator* – (Delfim, 2013)

Edge Bundle) e uma árvore em formato circular e *cluster*, criado com a biblioteca D3.js (Bostock, 2016) [1]. Nessa figura, todas as classe que possuem uma chamada para a classe selecionada *DataList* estão destacadas em vermelho e todas as classe que a *DataList* chama estão destacadas em verde.

3.3 Técnicas de interação e distorção

Para a manipulação dos dados apresentados pelas técnicas de visualizações é necessário a utilização de técnicas de interação para alterar dinamicamente as visualizações. Existem várias classificações para as técnicas de interação (Shneiderman, 1996, Keim, 2002, Yi et al., 2007, Ward et al., 2010). Além das técnicas de interação, técnicas de distorção podem ser utilizadas em conjunto com as de interação. Técnicas de distorção permitem que parte dos dados sejam observados com um maior nível de detalhes e outras com menor nível de detalhe (Keim, 2002). Nas próximas seções são descritas algumas técnicas.

3.3.1 Filtragem Interativa

Com uma grande quantidade de dados, as relações e outros atributos entre os dados se tornam mais difíceis de serem observados. Para que as informações sejam

[1] Disponível em: https://bl.ocks.org/mbostock/7607999

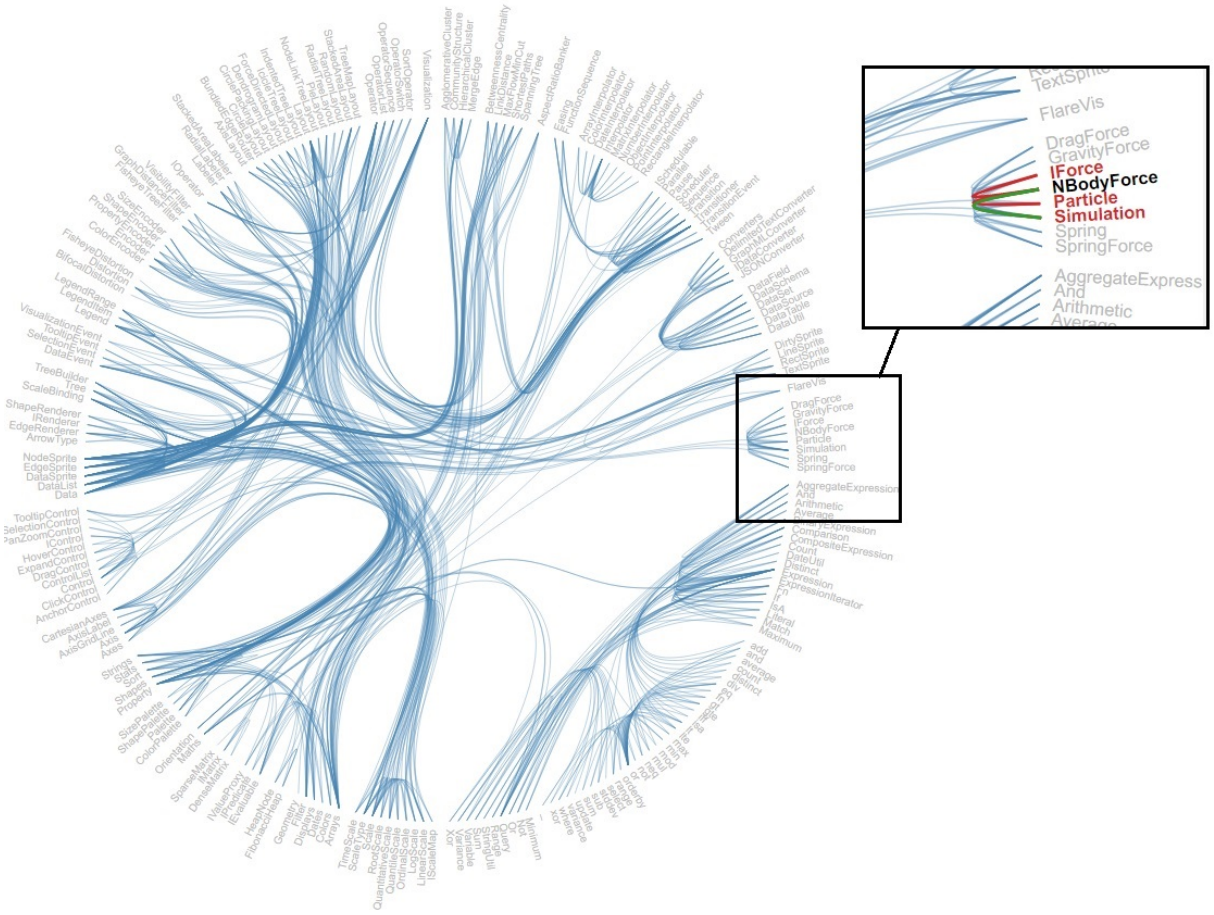

Figura 3.11: Exemplo da aplicação da técnica de agrupamento de linhas (*Hierarchical Edge Bundles*) com destaque para as chamadas de entrada ou saída para a classe *NBodyForce* de um software – (Holten, 2006)

destacadas pode ser feita uma seleção direta de um conjunto de dados (navegação – *browsing*) ou por selecionar um conjunto de dados de acordo com suas propriedades (consultando – *querying*) (Keim, 2002).

3.3.2 *Zoom* e *Pan*

De acordo com Keim et al. (2004), a técnica *Zoom* é muito utilizada quando existe uma quantidade muito grande de dados para serem mostrados de maneira reduzida, para que seja possível vê-los de maneira geral. Ao mesmo tempo, a técnica permite que sejam vistos os dados em diferentes resoluções. Essa técnica é utilizada para mostrar as informações em detalhe conforme solicitados pelo usuário e não apenas para mostrar os objetos com tamanho maior (Keim, 2002, Keim et al., 2004).

O *Pan* permite navegar pela visualização, alterando o foco, mas sem alterar a abstração das informações (Khan e Khan, 2011), por isso é muito comum utilizar as técnicas *Zoom* e *Pan* em conjunto. Na Figura 3.12 é apresentado um exemplo da técnica *Pan* em (a), a técnica *Zoom* em (b) e ambas técnicas em (c) (Elmqvist e Fekete, 2010).

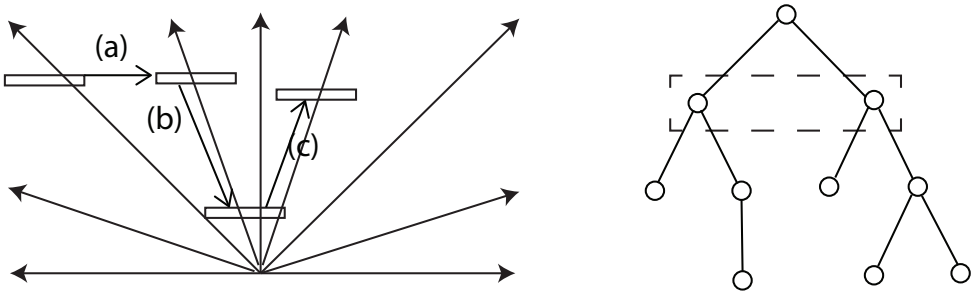

Figura 3.12: Técnica de interação *Pan* e *Zoom* (Elmqvist e Fekete, 2010). Em (a) é apresentado apenas a técnica *Pan*. Em (b) é apresentado apenas a técnica *Zoom*. Em (c) é apresentado as técnicas *Pan* e *Zoom* em conjunto.

3.3.3 *Drill-Down*

Essa técnica permite observar, com mais detalhes, dados hierárquicos (North e Shneiderman, 2000). Com essa técnica a partir de um dado inicial, os dados filhos ou agregados podem ser observados com mais detalhes em outras visões (Elmqvist e Fekete, 2010).

A técnica *roll-up* é o oposto da *drill-down*, mostrando menos detalhes da hierarquia de dados (Elmqvist e Fekete, 2010). Na Figura 3.13 são mostradas as técnicas de *drill-down* em (a) e de *roll-up* em (b) (Elmqvist e Fekete, 2010).

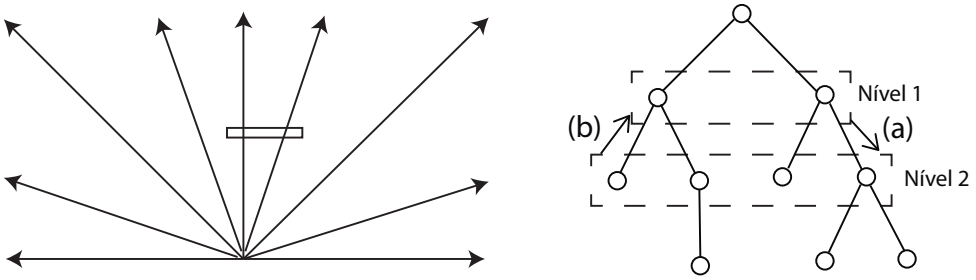

Figura 3.13: Técnicas *drill-down* e *roll-up* (Elmqvist e Fekete, 2010). Em (a) é apresentada a técnica *drill-down* mostrando do nível 1 para o nível 2. Em (b) é apresentada a técnica *roll-up* do nível 2 para o nível 1.

A técnica *drill-down* se diferencia da técnica *zoom* pelo detalhamento de informações mostradas pelas duas técnicas. A técnica *zoom* mostra os dados maiores, mas sem apresentar mais detalhes desses dados. A técnica *drill-down* apresenta os dados com um grau maior de detalhamento, mas sem apresentar eles maiores.

3.4 Múltiplas visões e técnicas de coordenação

De acordo com Roberts (2007), o termo múltiplas visões é utilizado para descrever qualquer instância onde os dados são representados em várias janelas. Isso implica que diferentes representações de um conjunto de dados são utilizadas e coordenadws entre si. É comum o uso de mais de uma técnica de visualização para aproveitar as

vantagens e superar as limitações que cada técnica apresenta (Keim et al., 2004, North e Shneiderman, 2000), explorando aspectos complementares entre as técnicas. De acordo com North e Shneiderman (2000), a coordenação de técnicas de visualizações é adequada para mostrar informações complexas, que precisam de diferentes visualizações para diferentes aspectos.

Na VisSoft a coordenação é muito utilizada por permitir a visão de diferentes níveis de abstração de informações em conjunto (Baldonado et al., 2000). A coordenação é utilizada, por exemplo, na visualização de aspectos estáticos da organização do código-fontes, juntamente com dados de execução e testes. Nesse caso, duas ou mais visualizações podem ser utilizadas para representar mais de uma característica que se pretende observar.

3.4.1 Tipos de visões e estratégias de exploração

Na literatura, sistemas com visualizações lado a lado são chamados de sistemas de visão dupla (*dual view systems*) (Convertino et al., 2003), mas existem variações desse tipo de visão, como a *Overview + detail*, visões de Foco + Contexto (*Focus + Context*) e visões de diferenças (*difference views*). A *Overview + detail* mostra em uma visão todo o conjunto de dados e em outra visão o mesmo conjunto com mais detalhes. Nas visões de Foco + Contexto apesar de comumente serem apresentados os detalhes de um conjunto de dados na mesma visão, nas visões coordenadas os detalhes são mostrados em outra visão. Nas visões de diferenças são ressaltadas as diferenças de um conjunto de dados entre duas visões. Essa visão é muito utilizada para observar diferenças entre textos no geral, como algoritmos (Roberts, 2007, Scherr, 2010).

Além desses tipos de visões, deve ser avaliado e decidido quais e como os dados nas múltiplas visões devem ser atualizados e/ou alterados. Para isso, Roberts (2000) dividiu em três categorias: substituição (*replace*), replicação (*replicate*) e sobreposição (*overlay*). Na Figura 3.14 é representado como funcionam cada uma das categorias.

Na substituição quando ocorre uma mudança de parâmetro em uma visão, nas outras visões todos os dados também são substituídos. Por conta disso, normalmente o histórico das mudanças é perdido, dificultando a comparação de pequenas mudanças. Na replicação as mudanças ocorrem em todas as janelas que possuem a visão daqueles dados. Neste caso, é mais fácil comparar as mudanças realizadas nos parâmetros dos dados. Roberts (2005) divide a replicação em duas categorias:(i) o procedimento – onde os resultados que são gerados pelas mudanças de parâmetros são exibidos em janelas separadas. (ii) o curso de ação – onde os mesmos dados podem ser apresentados por mapeamentos diferentes. Na sobreposição diferentes representações dos mesmos dados podem ser vistos na mesma exibição, como se houvessem camadas com as diferentes visões (Roberts, 2005).

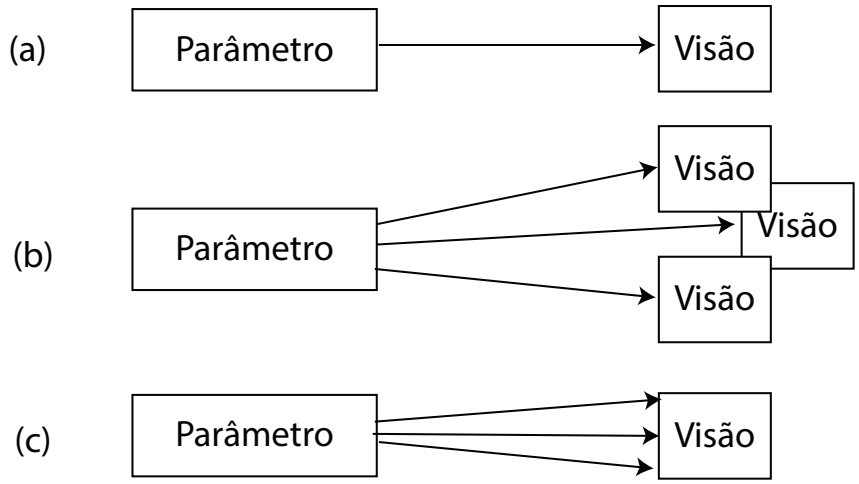

Figura 3.14: Representação das estratégias de exploração dos dados das visões coordenadas, sendo (a) substituição (*replace*), (b) replicação (*replicate*) e (c) sobreposição (*overlay*). Adaptado de Roberts (2005).

3.4.2 Coordenação entre visões

De acordo com Scherr (2010), para coordenar as visões é necessário especificar ou mapear como as mudanças em uma visão afetará as outras visões. Para isso são utilizadas técnicas de interação para coordenar as visões de acordo com a necessidade. Algumas das técnicas utilizadas na literatura e na industria de software são: *navigational slaving*, *linking-brushing* (Baldonado et al., 2000, Scherr, 2010).

A técnica *navigational slaving* os movimentos em uma visão são propagados nas outras visões disponíveis (Baldonado et al., 2000). Uma aplicação dessa técnica é o *scroll* sincronizado entre duas visões.

A técnica *linking* conecta os dados entre visões (Baldonado et al., 2000). A técnica *brushing* é uma técnica relacionada à *linking*, que permite selecionar um subconjunto de dados em uma visão, permitindo realizar ações como destacar, excluir, rotular, etc nas outras visões (Becker et al., 1987, Heinrich e Weiskopf, 2013). Na junção das duas técnicas (*brushing* - *linking*), os pontos selecionados para em uma visão são destacados em todas as outras visões disponíveis (Heinrich e Weiskopf, 2013). A união dessas técnicas facilita a identificação e relações entre as visões (Keim, 2002).

3.5 Validação de técnicas de visualização

A representação de uma visualização deve ser avaliada para garantir que ela apresente benefícios em sua utilização. Na literatura existem algumas regras para a criação de visualizações (Baldonado et al., 2000, Kienle e Muller, 2007), critérios de avaliação (Andrews, 2006, Lam et al., 2012, Winckler et al., 2004, Ellis e Dix, 2006) e *frameworks* de validação (Gallagher et al., 2008) que auxiliam para que a visualiza-

ção mostre as informações de maneira corretas e compreensiva aos seus utilizadores.

Dos trabalhos de validação pesquisados destacam-se os trabalhos de Andrews (2006), mais relacionados à avaliação e interação do utilizador final com a visualização, e de Munzner (2009), que avalia a visualização em diferentes níveis de informações.

Andrews (2006) afirma que os métodos de avaliação de uma visualização podem ser divididos em métodos de inspeção e métodos de teste. Nos métodos de inspeção um especialista em usabilidade ou Interação Humano-Computador (IHC), avalia a interface da ferramenta utilizando sua experiência nesse contexto. Nos métodos de testes são os representantes dos utilizadores finais que participam dos testes, para serem obtidas informações com relação ao uso e à compreensão ou não da visualização. Os métodos podem ser utilizados de maneira separada ou em conjunto para analisar a aplicação de uma visualização.

Alguns métodos de testes que podem ser utilizados com as visualizações são (Andrews, 2006):

- Testes formativos (*Formative testing*) – são observados problemas na utilização das visualizações e o motivo de ocorrerem. São normalmente realizados com poucos usuários.
- Testes somativos (*Summative testing*) – é realizada uma comparação de parâmetros entre duas ou mais visualizações, utilizando experimentos formais.
- Estudos de utilização (*Usage studies*) – são estudos que gravam e/ou observam a utilização da visualização por um usuário.

Em seu trabalho Munzner (2009) explica uma técnica, chamada de validação de modelos agrupados, para validar a visualização observando 4 camadas de informações. As camadas estão representadas na Figura 3.15.

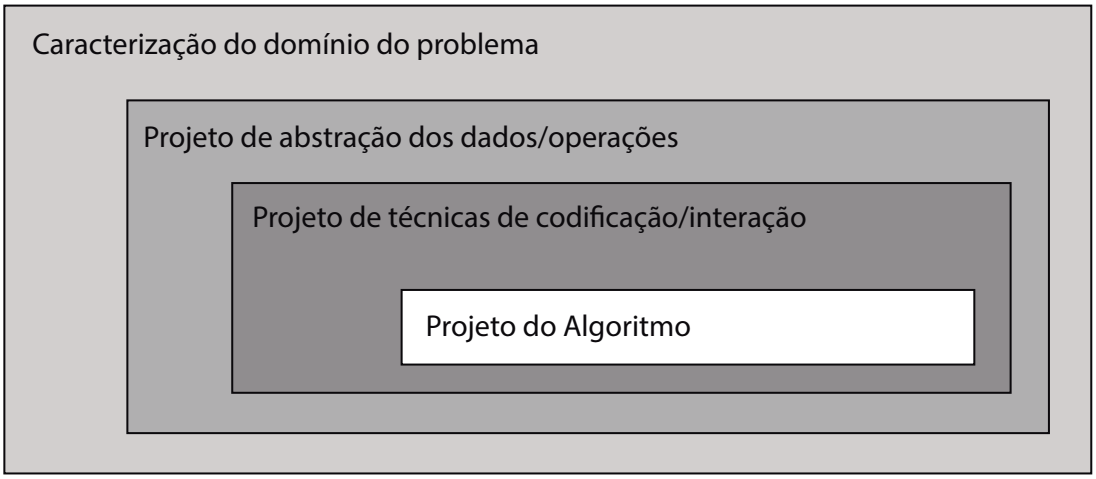

Figura 3.15: Representação do modelo de validação em 4 camadas - baseado em Munzner (2009)

No nível de *caracterização do domínio do problema*, a afirmação é que os problemas particulares do público-alvo seriam beneficiados pelo apoio à ferramenta de visualização. A ameaça principal com relação à afirmação é de que o problema está descaracterizado, ou seja, o público-alvo não possui o problema declarado. Uma das maneiras de validação da caracterização do problema é a realização de entrevistas com o público-alvo para verificar se a caracterização está correta. No nível de *projeto de abstração dos dados/operações*, a ameaça é que as informações escolhidas para serem representadas não resolvem os problemas caracterizados pelo público-alvo. O ponto principal de validação contra esta ameaça é que o sistema deve ser testado por usuários que fazem parte do público alvo. Com relação ao nível *projeto de técnicas de codificação/interação* a ameaça é que o a escolha das técnicas de visualização e interação não é eficaz em comunicar a abstração das informações desejada para a pessoa que utilizam a ferramenta. Uma das maneiras de validação é a utilização de avaliações de usabilidade preditivas, por exemplo uma avaliação heurística, para avaliar antecipadamente a usabilidade da ferramenta. Com relação ao *projeto de algoritmo* a ameaça principal é de que o algoritmo utilizado para criar a visualização não possui um bom desempenho de memória utilizada. Uma maneira de validação é analisar a complexidade computacional do algoritmo utilizado.

A técnica de validação de modelos agrupados serve para garantir que os problemas encontrados no processo de validação com usuários não sejam problemas que poderiam ser corrigidos durante a criação da visualização ou da criação e/ou leitura dos dados. Munzner (2009) afirma que essas quatro camadas estão ligadas. Por exemplo, se é realizada uma má escolha com relação à abstração das informações, mesmo o melhor algoritmo para a criação da visualização não irá resolver o problema criado na camada de abstração.

3.6 Considerações finais

Técnicas de VisSoft podem ser utilizadas para representar os múltiplos aspectos da interface do usuário (dados multidimensionais), porém poucos trabalhos apresentam mais de um aspecto (Rooke et al., 2011). Na literatura, as técnicas hierárquicas (Seção 3.2.2) são utilizadas na VisSoft para apresentar a estrutura e outras hierarquias de um software. As técnicas baseada em grafos (Seção 3.2.1) são utilizadas na VisSoft para representar as relações entre informações de um software, como chamadas de métodos e eventos relacionados (Delfim, 2013). As técnicas hierárquicas e baseadas em grafos foram selecionadas para representar visualmente as informações dos modelos navegacional, comportamental e de diálogo por conta do tipo de informação extraída da interface de uma aplicação *web* (Capítulo 2).

Neste trabalho são utilizadas técnicas de interação e de coordenação para possibilitar o uso de duas ou mais visões em conjunto e ampliar a compreensão das

informações apresentadas. Dentre as técnicas descritas na Seção 3.3 e na Seção 3.4 foram utilizadas:

- Técnica de *Zoom* e *Pan* para interação nas visões
- Técnica de *linking* para a coordenação entre diferentes visões
- Técnica *Drill-down* para apresentação dos subníveis de abstração dos modelos de comportamento e de diálogo.

Para encontrar possíveis problemas de interação do usuário com a aplicação das técnicas de visualização, foram pesquisadas técnicas de avaliação das visualizações (Seção 3.5). As técnicas de avaliação de algoritmos, entre outras técnicas, não foram descritas por serem aplicadas técnicas de visualização já conhecidas e validadas. A aplicação das técnicas de avaliação é descrita no Capítulo 5 com os resultados obtidos nas avaliações.

CAPÍTULO 4

Ferramentas criadas

Nesse capítulo são apresentados as ferramentas criadas em conjunto com a representação visual ModelUI$_{VIZ}$. Para a extração dos dados de uma interface *web* foi desenvolvido o *plug-in WebModelUI Data* para o Navegador *Google Chrome*, apresentado na Seção 4.3. Esse *plug-in* extraí os dados da interface utilizando as técnicas descritas no Capítulo 2.

A ferramenta *WebModelUI Tool*, apresentada na Seção 4.4, foi criada para interpretar os dados extraídos do *plug-in* e apresentar o modelo visual do site utilizando a ModelUI$_{VIZ}$. Nesse trabalho a ferramenta foi separada da especificação da ModelUI$_{VIZ}$ para que essa representação possa ser utilizada independente do ambiente da ferramenta. A especificação da ModelUI$_{VIZ}$ é apresentada no Capítulo 5 em conjunto com as avaliações realizadas.

4.1 Modelo de referência

Para representar os dados da interface e apresenta-los usando a ModelUI$_{VIZ}$ esses dados devem passar por uma sequência de etapas, representados pela especificação do modelo de referência do processo de Visualização de Informação apresentado por Card et al. (1999) e mostrado na Figura 4.1. Nessa figura a **fonte de dados** é obtida a partir da extração dos dados das páginas *web* usando o *plug-in WebModelUI Data*. A tarefa de **seleção** indica quais dados da página *web* são extraídos de acordo com a opção selecionada de *Crawler* ou *Tracer*, implementado no *plug-in*. A etapa de **transformações de dados** indica que os **dados selecionados** são organizados em **tabelas de dados**, usando arquivos XML.

Durante o **mapeamento visual**, as **tabelas de dados** são transformadas em **abstrações visuais** usando atributos visuais pela ferramenta *WebModelUI Tool*. Diferentes tipos de **abstrações visuais** são criadas de acordo com a seleção de dados feita

anteriormente.

O último passo é a **transformação de visões** para criar a estrutura visual, o ModelUI$_{VIZ}$, de acordo com **abstração visual** e o parâmetro definido pelo usuário no *plug-in*. Na Figura 4.1 há duas linhas tracejadas que ligam as **visões**, o que significa que é possível explorar as características entre as **visões**, utilizando as técnicas de coordenação.

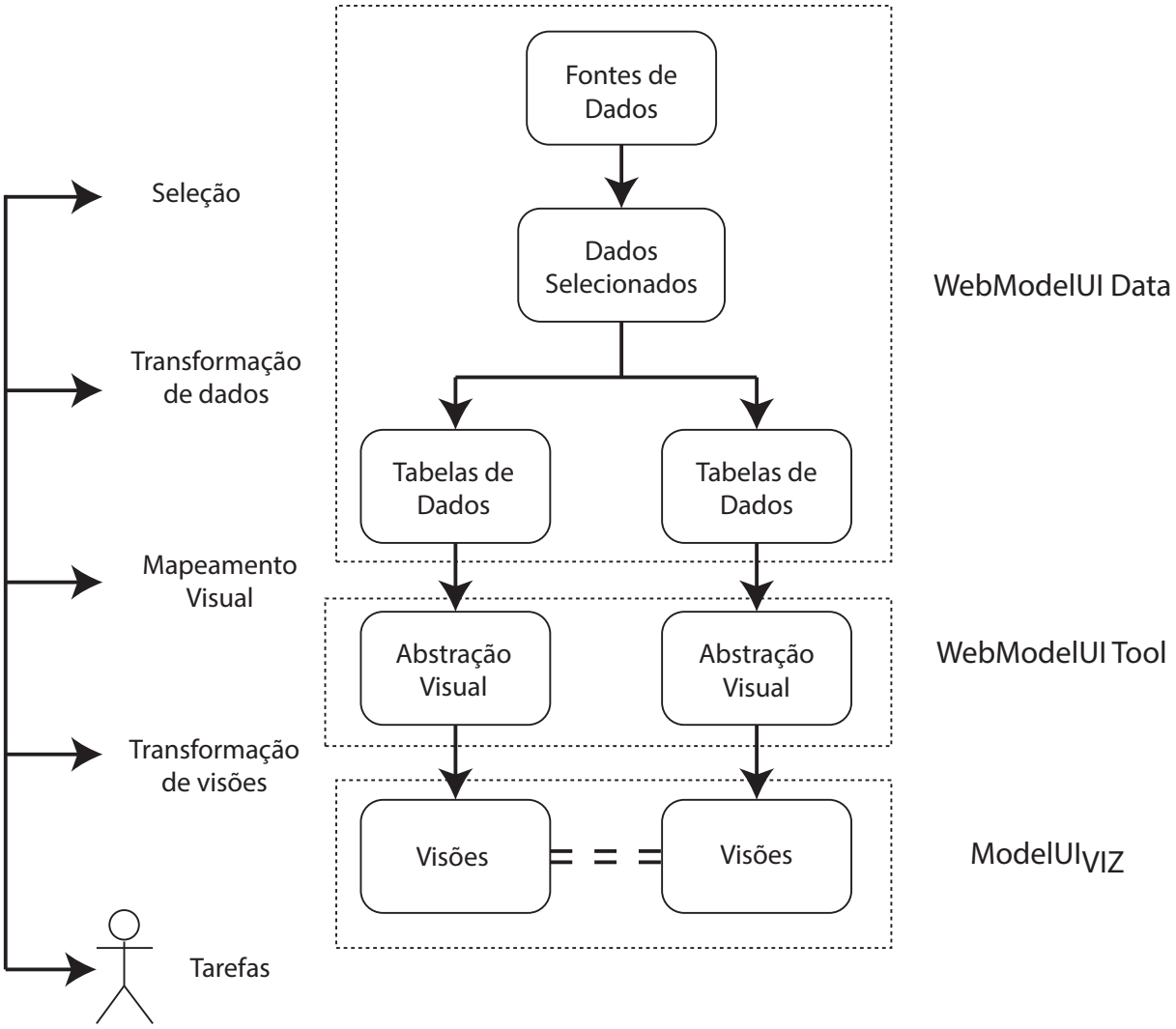

Figura 4.1: *Pipeline* da ModelUI$_{VIZ}$, baseado em Card et al. (1999).

4.2 Avaliação de alternativas ao *plug-in*

Inicialmente, foi prevista a adaptação da ferramenta *FireDetective* (Zaidman et al., 2013) para adicionar o processo de mapeamento do *crawler* e extrair as informações dos modelos de interface das páginas. Porém, a *FireDetective* foi criada especificamente para uma versão antiga do navegador *Mozilla Firefox*, com a linguagem de programação Java. Atualmente, existe a previsão que o navegador *Mozilla Firefox* permitirá a utilização apenas da linguagem *JavaScript* para o desenvolvimento de *plug-ins*.

A adaptação da *FireDetective* para a versão mais atual do *Mozilla Firefox* foi avaliada, porém descartada por problemas relacionados a API. A API de desenvolvimento do *Mozilla Firefox* não permite o acesso à lista de eventos associados aos componentes de uma página, que é necessário para o mapeamento do *site* com o *crawler*.

Antes da criação do *plug-in* WMUID para o navegador *Google Chrome*, foi criada uma versão totalmente *web* do *crawler*, como alternativa a adaptação da ferramenta *FireDetective*. Porém, por questão de segurança, os navegadores não permitem o acesso a lista de eventos associados a um componente em uma página *web*. O acesso aos eventos existentes de um *site* é bloqueado, evitando-se a inclusão de códigos maliciosos durante a interação do usuário com a página.

No navegador *Google Chrome*, no início do processo de instalação de um *plug-in* o usuário concede o acesso aos dados das páginas *web* para o *plug-in*, por meio da API do navegador. Por esse motivo, foi criado um *plug-in* para esse navegador.

4.3 *Plug-in WebModelUI Data*

O *WebModelUI Data* (WMUID) é um *plug-in* do navegador *Google Chrome*, criado para extrair os dados da interface de aplicações *web*. Foi desenvolvido com a linguagem de programação *JavaScript* e com HTML e CSS para a interface.

O WMUID utiliza a API (*Application Programming Interface*) do painel de desenvolvedor do navegador *Google Chrome* para ter acesso a todos os elementos e a estrutura das páginas *web*. O *plug-in* possui duas opções para a extração dos dados das páginas *web*: *crawler* e *tracer*, sendo apresentada a interface do *plug-in* na Figura 4.2.

Figura 4.2: Interface do *plug-in* WMUID.

A opção *crawler* gera um arquivo XML com a estrutura das páginas, os elementos interativos de cada página e as conexões entre as páginas, sendo descrita com mais detalhes na Seção 4.3.1. A opção *tracer* faz o rastreio da navegação e as interações do usuário com interface do *site*, gerando um arquivo XML esses dados. O *tracer* é descrito na Seção 4.3.2.

4.3.1 *Crawler*

O *crawler* mapeia o conteúdo das páginas *web*, lendo a estrutura HTML em busca de elementos interativos. O *plug-in* identifica como elemento interativo:

- Os nativamente interativos do HTML, como botões e *links*.
- Os que possuem eventos associados a esses elementos, como *click* e *keypress*.

Para o mapeamento das conexões entre as páginas o *crawler* utiliza a estratégia de rastreamento em largura (*Breadth-First*), descrito na Seção 2.3.2. Ambos rastreamentos (em largura ou profundidade) podem ser utilizados para o rastreamento nas páginas, mas neste trabalho a estratégia em largura foi escolhida para facilitar a descoberta de páginas novas a partir da navegação principal do *site*. Durante o mapeamento, o *crawler* não dispara nenhum evento associado a um elemento interativo. Por conta disso, as conexões entre as páginas são feitas apenas para identificação da *tag* < a >, gerando uma lista com todos as conexões encontradas. A cada página mapeada, essa lista com as conexões é atualizada se necessário.

A identificação dos elementos interativos (nativamente ou não) pelo *crawler* é feita pelo acesso a lista de eventos que a API do navegador *Google Chrome* disponibiliza. Para os elementos interativos nativos do HTML são atribuídos os estados padrões disponíveis pela documentação da W3C (W3C, 2016b). Caso o elemento não seja nativamente interativo, os estados são atribuídos de acordo com os eventos associados. Um exemplo de elemento interativo não-nativo do HTML é mostrado na Listagem 4.1, na qual foi atribuído o evento *click* à uma *div*. Para esse tipo de evento são atribuídos os estados *selected* e *notselected* para aquele componente.

Listagem 4.1: Exemplo de elemento interativo não-nativo identificado pelo *crawler* do WMUID

```
<component type="div">
    <event name="click"></event>
    <state name="selected"></state>
    <state name="notselected"></state>
</component>
```

Na Figura 4.3 é mostrada a transferência de dados entre os arquivos do *crawler*. O *controler.js* controla a interface do *plug-in* WMUID. Após iniciar o *crawler* pelo *controler.js*, *background.js* é iniciado, intermediando a comunicação entre o *crawler.js* e o *dev.js* por questões de segurança do navegador. O *dev.js* acessa os recursos a nível de HTML e *JavaScript* pela API do *Google Chrome*. O *crawler.js* controla a leitura dos elementos interativos da página e a atualização da lista de páginas a serem mapeadas. Após todas as páginas da lista serem acessadas, os dados são enviados para o *filesaver.js* criar o arquivo XML.

O arquivo XML criado é composto por 3 partes mostrado na Listagem 4.2. A primeira parte é a descrição dos elementos interativos que existem nas páginas, com os eventos e estados que cada página e componentes (elementos interativos) possuem. Na segunda parte são mostradas as conexões entre páginas, com a indicação

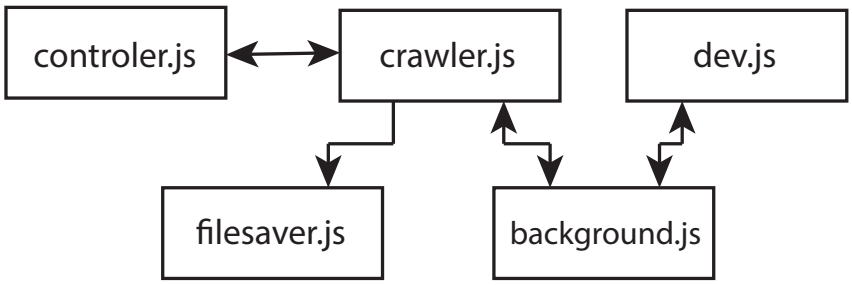

Figura 4.3: Estrutura de transferência de dados entre os arquivos *JavaScript* do *crawler – plug-in* WMUID.

dos componentes (listados na primeira parte do XML) e os eventos acionados pela mudança de páginas. A última parte, apresenta a estrutura e a localização dos elementos interativos nas páginas, sendo criada para a Versão 3 da ModelUI$_{VIZ}$ (Seção 5.6).

Listagem 4.2: Exemplo da estrutura do arquivo XML criado pelo *crawler* do WMUID

```
<?xml version="1.0" encoding="UTF-8"?>
<site url="" titulo="" tipo="crawler">
    <pages>
        <page>
            <component>
                <event></event>
                <state></state>
            </component>
        </page>
    </pages>
    <edges>
        <edge source="" target="">
            <data>event</data>
        </edge>
    </edges>
    <structure>
        <page>
            <node></node>
        </page>
    </structure>
</site>
```

4.3.2 *Tracer*

O *tracer* monitora o comportamento da interface durante a interação do usuário na interface. Para iniciar o monitoramento é necessário carregar o arquivo XML criado pelo *crawler*, para que não seja necessário realizar o mapeamento da interface novamente. Por conta disso, o ponto de início do *tracer* deve ser o mesmo que do *crawler*. Após o carregamento do XML, o monitoramento é iniciado e o usuário pode parar a qualquer momento, gerando ao final o arquivo XML.

Na Figura 4.4 é mostrada a transferência de dados entre os arquivos do *tracer*. O *controler.js* e *filesaver.js* são os mesmo da parte do *crawler* que controlam, respectivamente, a interface do *plug-in* e a criação do arquivo XML. O arquivo *tracer.js* monitora as interações dos usuários e o comportamento da interface de acordo com as interações.

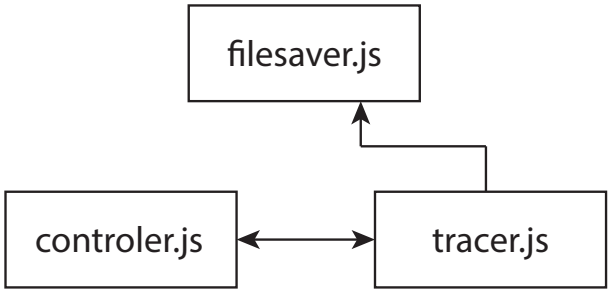

Figura 4.4: Estrutura de transferência de dados entre os arquivos *JavaScript* da parte do *tracer* do *plug-in* WMUID.

O arquivo XML criado é composto por duas partes, mostrado na Listagem 4.3. A primeira parte é a mesma primeira parte do arquivo do *crawler*. A segunda parte possui o registro das interações realizadas pelo usuário na interface, com as informações de estados iniciais e finais, eventos acionados na interação e, se houve uma mudança de página, para qual página o usuário foi direcionado.

Listagem 4.3: Exemplo da estrutura do arquivo XML criado pelo *tracer* do WMUID

```
<?xml version="1.0" encoding="UTF-8"?>
<site url="" titulo="" tipo="tracer">
    <pages>
        <page>
            <component>
                <event></event>
                <state></state>
            </component>
        </page>
    </pages>
    <interactions>
        <interaction></interaction>
    </interactions>
</site>
```

4.4 *WebModelUI Tool*

A ferramenta *WebModelUI Tool* (WMUIT) foi desenvolvida utilizando *JavaScript* como linguagem de programação e HTML e CSS para a interface. Para a criação da ModelUI$_{VIZ}$ foi utilizada a biblioteca D3.js (Bostock, 2016) como base. A D3.js é uma biblioteca *JavaScript* que utiliza HTML, CSS e SVG (*Scalable Vector Graphics*

– (W3C, 2016a)) para apresentar os dados. Atualmente a biblioteca está na versão 4.+, mas para a criação da ModelUI$_{VIZ}$ foi utilizada a versão 3.5.17, última versão estável. Durante o desenvolvimento do projeto as versões da biblioteca D3.js foi atualizada várias vezes e se mantém até o momento em constante modificação. Essas versões eram disponibilizadas gradualmente, ou seja, apenas uma parte da biblioteca era atualizada. Quando todas as partes utilizadas neste projeto foram atualizadas na biblioteca, a ModelUI$_{VIZ}$ já estava no processo de finalização e início das avaliações. Assim, para que o projeto permanecesse estável para as avaliações optou-se por aguardar e realizar a atualização da ModelUI$_{VIZ}$ para a versão mais estável da D3.js. Na Figura 4.5 é mostrada a interface da ferramenta *WebModelUI Tool*.

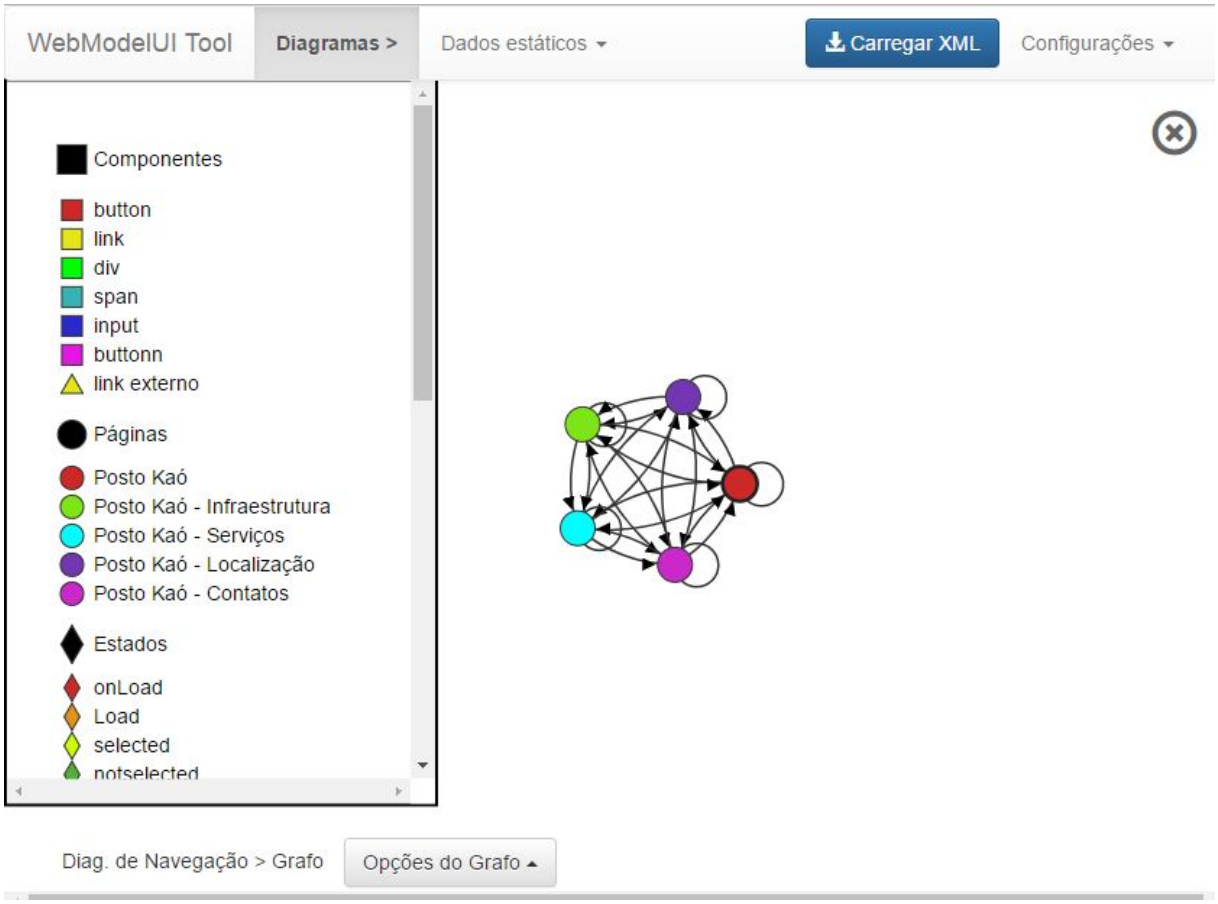

Figura 4.5: Interface da ferramenta *WebModelUI Tool*.

Durante o desenvolvimento do projeto foram produzidas três versões diferentes para a ModelUI$_{VIZ}$, apresentado no próximo capítulo.

4.5 Considerações finais

Neste capítulo foram apresentados o *plug-in* WMUID e a ferramenta WMUIT. A ferramenta WMUIT e a ModelUI$_{VIZ}$ são apresentadas separadamente para que a representação visual desenvolvida não fique limitada apenas a ferramenta WMUIT desenvolvida. Futuramente, outras ferramentas podem ser criadas para apresentar a ModelUI$_{VIZ}$.

Durante o processo de avaliação da ferramenta WMUIT e da ModelUI$_{VIZ}$ foram identificadas as necessidades de novas funcionalidades para melhorar a usabilidade da WMUIT e novas visões para melhorar a compreensão da ModelUI$_{VIZ}$. A especificação da ModelUI$_{VIZ}$ e sua evolução da são apresentadas no próximo capítulo.

CAPÍTULO 5

ModelUI$_{VIZ}$ e Avaliações

Nesse capítulo é apresentado a especificação do modelo visual ModelUI$_{VIZ}$ e as evoluções realizadas após avaliações dos experimentos controlados e os resultados obtidos pelos experimentos controlados. Para extração e transformação dos dados apresentados pela ModelUI$_{VIZ}$ passam por uma sequência de etapas, apresentadas no capítulo anterior.

Foram realizados três experimentos controlados para a avaliação de cada uma das versões. O 1º experimento, apresentado na Seção 5.3, foi realizado com 10 participantes. O objetivo foi de avaliar o projeto experimental, assim como o material utilizado como instrumentação, servindo como uma preparação para os outros experimentos. Com os resultados desse experimento foram realizadas modificações na ModelUI$_{VIZ}$ apresentadas na Seção 5.4.

No 2º experimento, realizado com 9 participantes, houve modificações na estrutura seguindo as lições aprendidas do 1º experimento (Seção 5.3.1), como mostrado na Seção 5.5. Porém, pela pequena quantidade de participantes e com os resultados apresentados no 2º experimento (Seção 5.5.1), foi observada a necessidade de modificações na ModelUI$_{VIZ}$ (Seção 5.6) e da realização de um 3º experimento para avaliar as modificações feitas. No último experimento houve 18 participantes com resultados favoráveis a utilização da ModelUI$_{VIZ}$, como mostrado na Seção 5.7.1.

5.1 ModelUI$_{VIZ}$ - Versão 1

Na ModelUI$_{VIZ}$ são utilizadas 4 símbolos para representar os elementos existentes nas páginas:

- Círculo: representa uma página
- Quadrado: representa um componente

- Losango: representa um estado
- Cruz: representa um evento

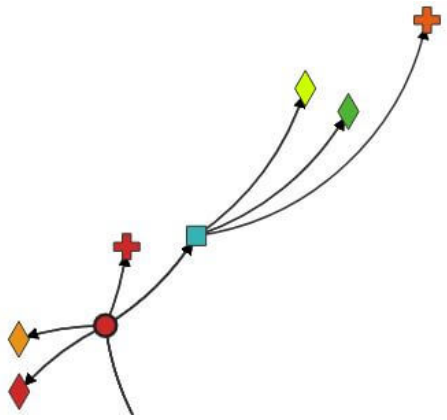

Figura 5.1: Exemplo dos símbolos utilizados para representar as páginas, componentes, estados e eventos na ModelUI$_{VIZ}$.

Na Figura 5.1 são mostrados exemplos dos símbolos utilizados. A leitura das informações nos grafos é feita de acordo com a direção que as setas apontam. Na Figura 5.1 é mostrada uma página (círculo vermelho) que possui dois estados (duas setas partem do círculo para os losangos laranja e vermelho) e um evento (uma seta parte do círculo para uma cruz vermelha). Essa página possui um componente (seta aponta para quadrado azul) e este componente possui dois estados (duas setas partem do quadrado para os losangos amarelo e verde) e um evento (uma seta parte do quadrado para a cruz laranja). Na Figura 5.4 é possível observar todos as páginas, componentes, estados e eventos em de um site apresentados em uma visão.

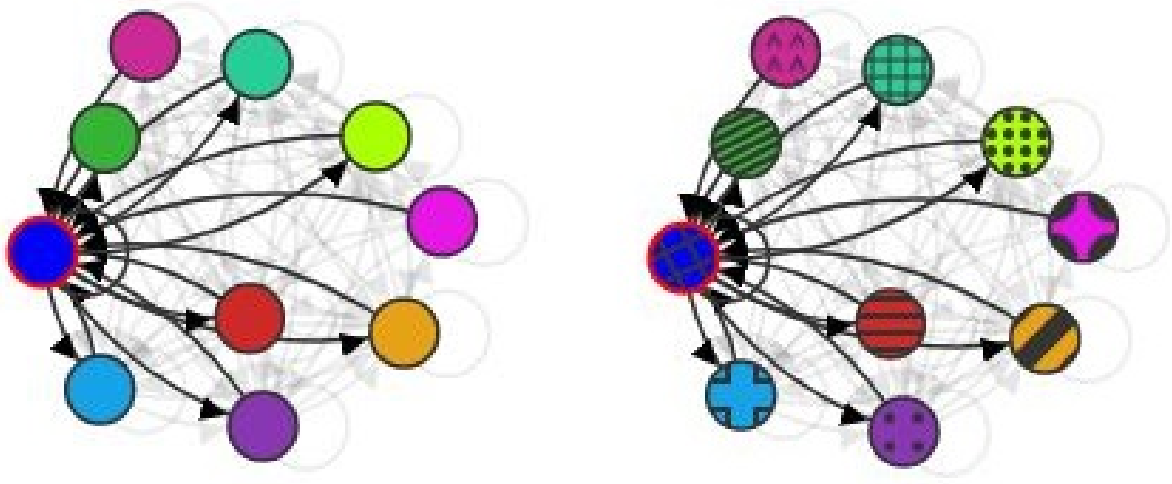

Figura 5.2: Do lado esquerdo é mostrada a estrutura de navegação sem a aplicação da textura e no lado direito é mostrada a mesma estrutura com a aplicação de textura nas páginas do *site* Multicobra (Multicobra, 2017).

Na ModelUI$_{VIZ}$ o atributo de cor é utilizado para separar os componentes, estados e eventos em categorias. Nas páginas o atributo de cor é utilizado para diferenciá-las.

Na WMUIT é possível aplicar texturas nas páginas para facilitar a diferenciação entre elas. Na Figura 5.2 é mostrado um exemplo da aplicação de texturas nas páginas na representação da navegação do *site* Multicobra (Multicobra, 2017), apresentada em um grafo. Do lado esquerdo as páginas são mostradas sem a aplicação de textura e do lado direito com aplicação da textura.

Figura 5.3: Trecho da legenda do *site* Posto Kaó (Posto Kaó, 2016).

As cores para cada categoria de componentes, eventos e estados variam de acordo com a estrutura de cada *site*. As cores são atribuídas de acordo com a quantidade de elementos existentes no *site*. Quanto maior a quantidade de elementos, menor a diferença entre as cores dos elementos. Para facilitar a identificação dos elementos é apresentada uma legenda para cada uma das categorias e páginas para cada site. Na Figura 5.3 é mostrado um trecho da legenda do *site* Posto Kaó (Posto Kaó, 2016).

5.1.1 Visões da ModelUI$_{VIZ}$

A visão por páginas mostra todas as informações relacionadas à uma determinada página, como mostrada na Figura 5.4. Nessa figura são mostradas todas as conexões entre as outras páginas do *site*, os componentes, os eventos e os estados existentes na página *Posto Kaó*.

Nas visões de diálogo detalhado e de comportamento como existem muitas repetições de componentes nas páginas, os componentes são mostrados de modo agregado. Ou seja, de acordo com a categoria e a página que um componente possui, ele é agregado para apenas um componente representativo daquela categoria. Na Figura 5.5 é apresentada a visão de diálogo com todos os componentes e eventos do *site* Posto Kaó e na Figura 5.6 também é apresentada a visão de diálogo detalhado do mesmo *site*, mas com os componentes agregados em categorias. Os eventos e estados dos componentes agregados são mostrados no componente resultante da agregação. A agregação de atributos dos componentes foi feita para melhorar o

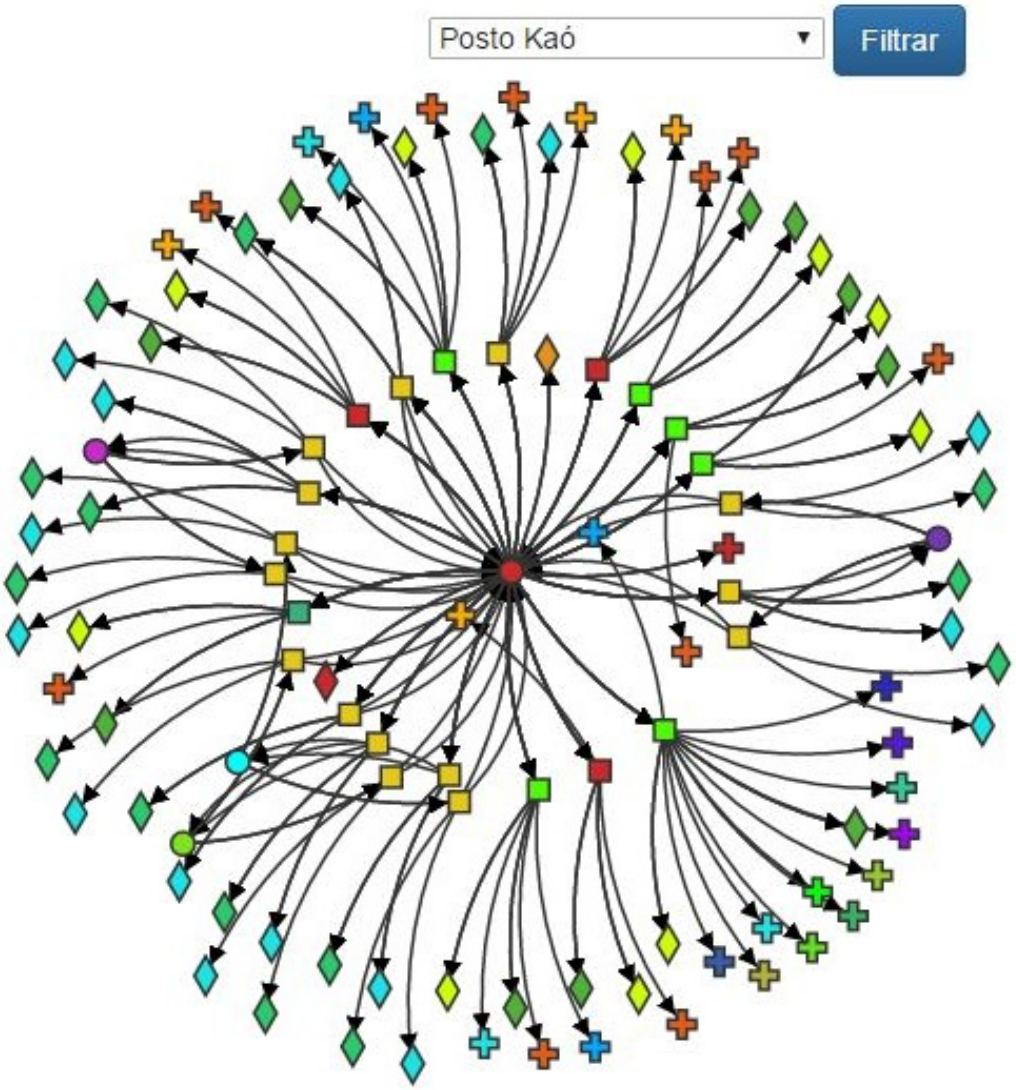

Figura 5.4: Visão de todos os componentes, eventos, estados e conexões com outras páginas da página Posto Kaó.

processamento da ferramenta na criação das representação e para melhorar o entendimento das informações.

Nas visões de diálogo detalhado e de comportamento é possível habilitar um "*zoom*" para ver os elementos sem a agregação em categorias. Quando um componente é selecionado nas representações com informações agregadas, é mostrada uma janela com todos os componentes, eventos ou estados que foram agregados no elemento selecionado. Na Figura 5.7 são apresentados todos os elementos *div* da página *Posto Kaó* existentes e todos os eventos relacionados à cada um dos elementos.

A visão de navegação utilizando a estrutura de árvore apresenta inicialmente o primeiro nível de conexões com a página inicial. A árvore pode ser expandida, clicando em cada um dos nós da árvore com maior tamanho. Na Figura 5.8 é mostrado um exemplo da diferença de tamanho entre os nós que podem ser expandidos e os nós que não possuem filhos. Os nós vermelhos *Posto Kaó* não possuem filhos por isso eles possuem um tamanho menor que os nós *Posto Kaó - Serviços* (azul - claro)

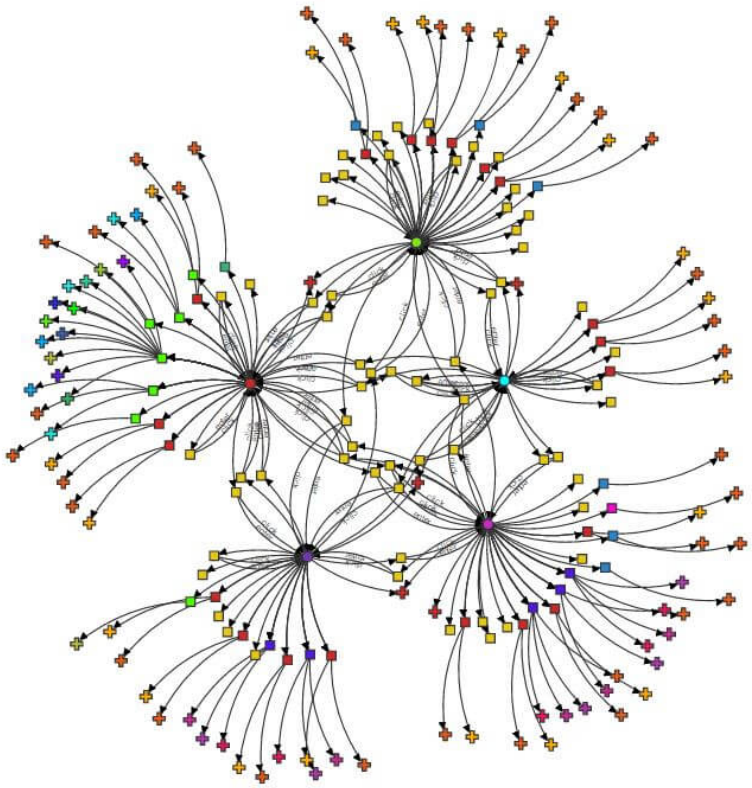

Figura 5.5: Representação da estrutura do *site* Posto Kaó com os componentes, eventos e conexões entre páginas sem agregação de informações.

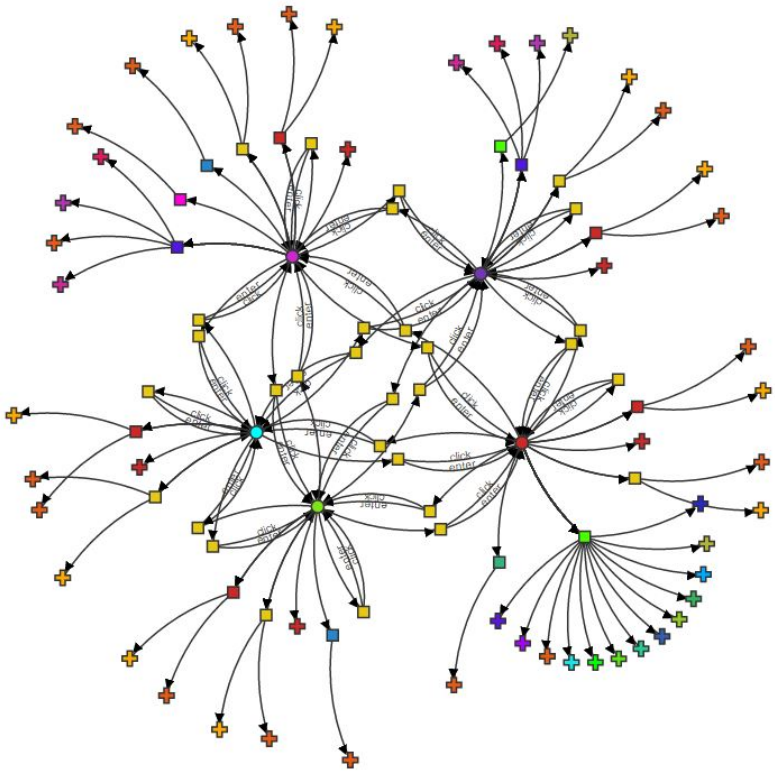

Figura 5.6: Representação da estrutura do *site* Posto Kaó com os componentes, eventos e conexões entre páginas com agregação de informações.

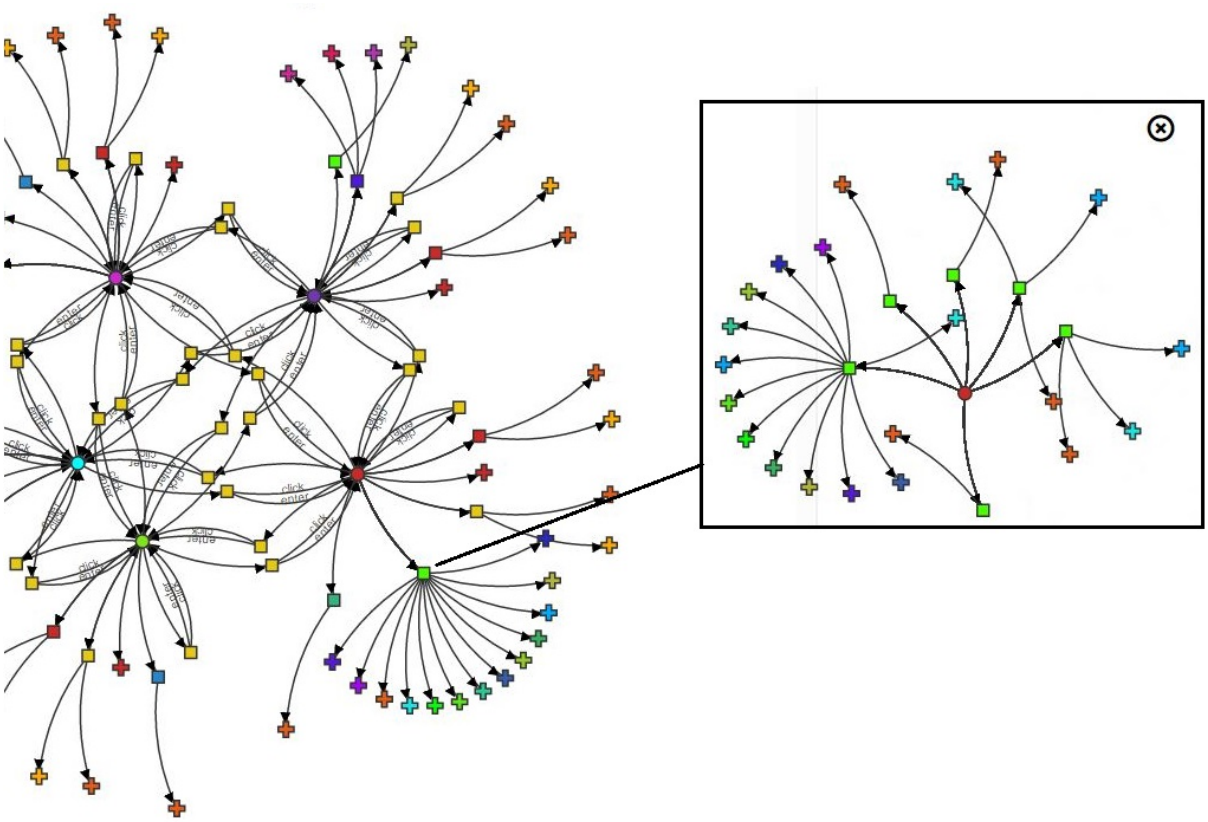

Figura 5.7: Exemplo de detalhamento de um componente *div*, apresentado em uma janela sobreposta a visão principal da Figura 5.6.

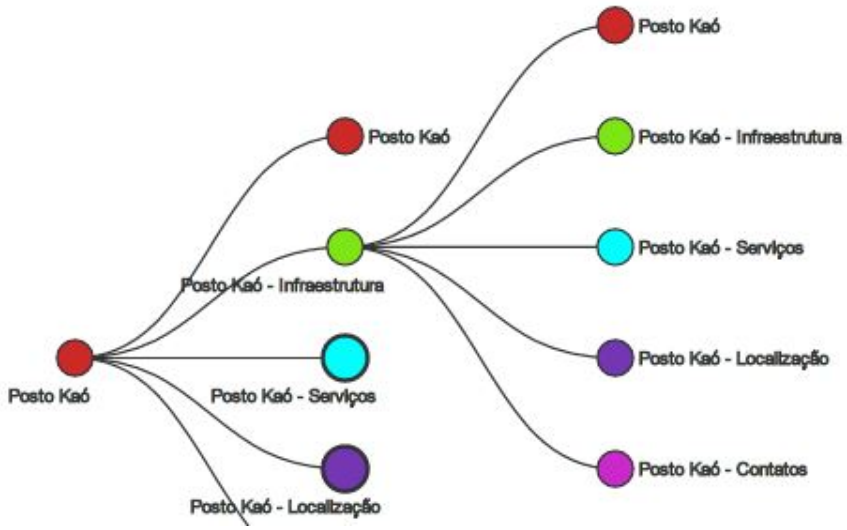

Figura 5.8: Trecho da árvore de navegação do *site* Posto Kaó, com as páginas *Posto Kaó* (vermelho) e *Posto Kaó - Infraestrutura* (verde) expandidas.

e *Posto Kaó - Localização* (roxo). O nó *Posto Kaó - Infraestrutura* (verde) foi expandido e por isso ele possui o tamanho do nó *Posto Kaó* (vermelho) (Posto Kaó, 2016).

Os dados gerados pelo *crawler* do *plug-in* WMUID não mostram quais são os estados iniciais e finais dos componentes e das páginas. Quando o *tracer* realiza o monitoramento das páginas, os dados iniciais e finais são identificados. Por isso, o diagrama de interação é mostrado apenas quando o XML criado pelo *tracer* é carre-

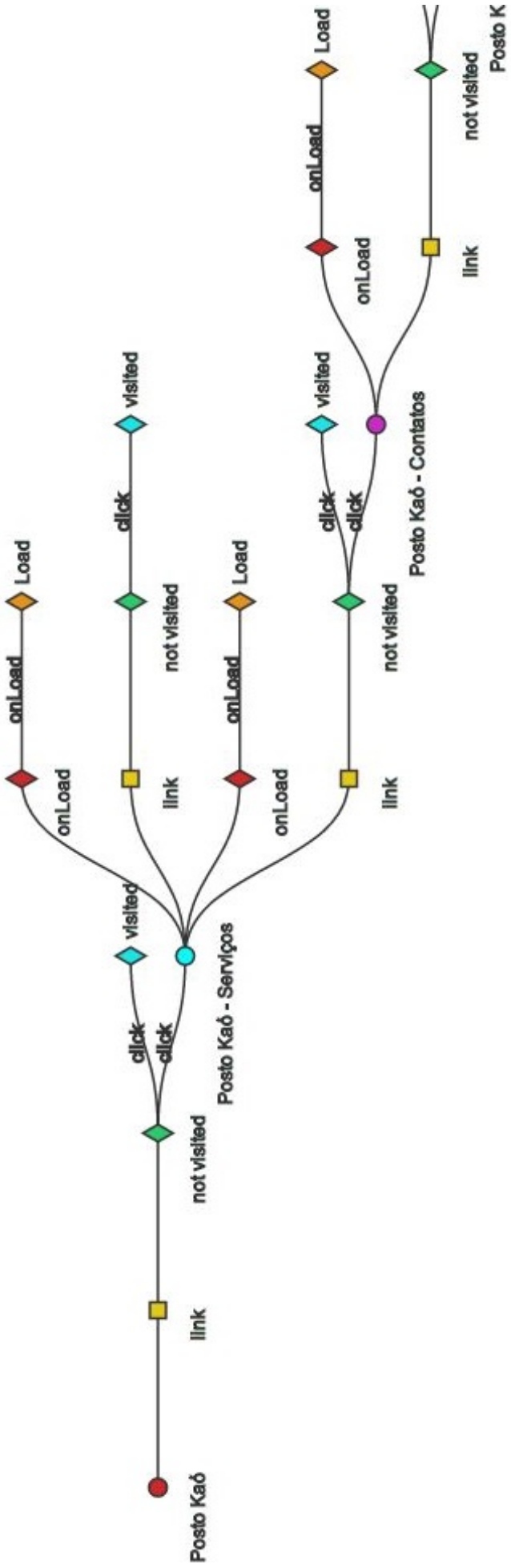

Figura 5.9: Trecho da representação da interação no *site* Posto Kaó.

gado na WMUIT. Nessa visão é apresentado o comportamento da interface de acordo com a interação do usuário. Um exemplo dessa estrutura de navegação pelo *site* do Posto Kaó é apresentado na Figura 5.9.

5.1.2 Interações e coordenações nas representações

Figura 5.10: Estado *not visited* foi selecionado e aparece um painel de detalhes com informações do estado selecionado.

Os elementos páginas, componentes, eventos e estados podem ser selecionados para obtenção de mais detalhes em um painel, como mostrado na Figura 5.10.

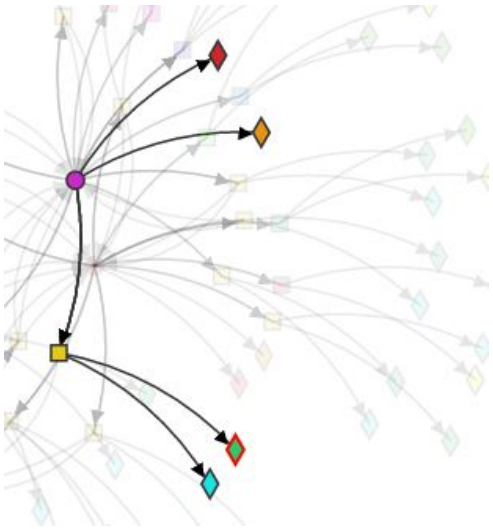

Figura 5.11: Imagem mostra um estado selecionado e os outros elementos relacionados também são destacados.

Quando um elemento é selecionado, os elementos relacionados a ele também são destacados. Na Figura 5.11 é mostrado um estado de um componente *link* selecionado. Além do estado selecionado, também são destacados: i) o componente que aquele estado selecionado pertence ii) outros estados do componente iii) a página que o componente faz parte iv) estados que pertencem a página do estado selecionado. A seleção do elemento é removida quando: i) o painel de detalhes é fechado, ii) é clicado

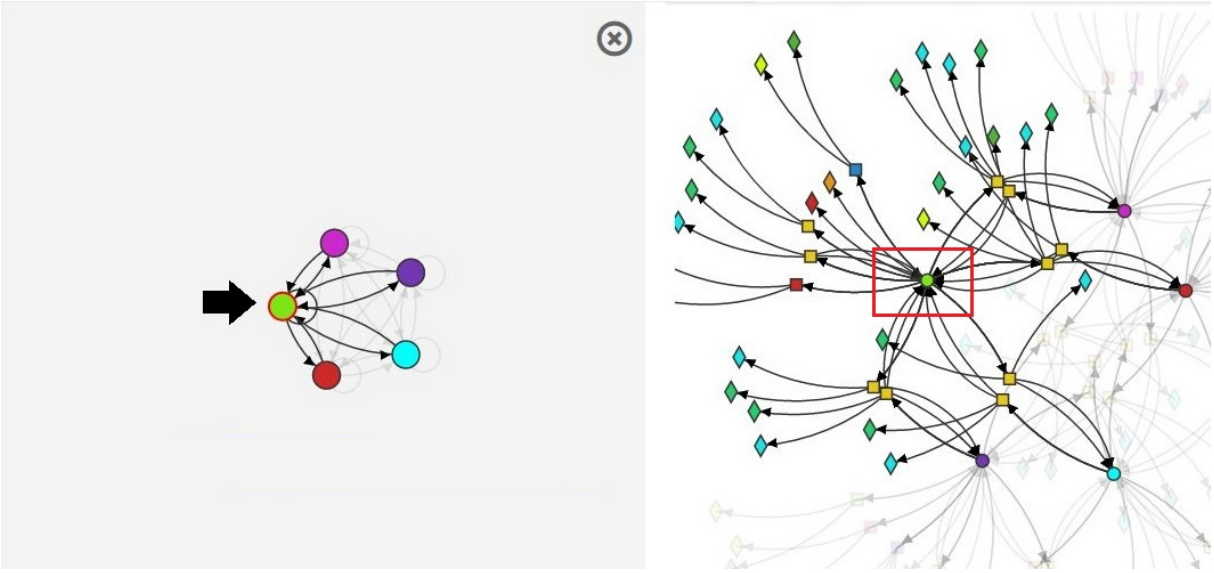

Figura 5.12: Exemplo de coordenação entre duas visões, na qual uma página é selecionada no lado direito e todos os componentes e estados que fazem parte dessa página são destacados do lado esquerdo.

novamente no mesmo elemento ou iii) outro elemento é selecionado. Atualmente, a seleção é realizada com um elemento por vez.

Além dessas interações, é possível mover, ampliar ou diminuir as representações por meio das técnicas *Pan* e *Zoom* (Seção 3.3).

Na ferramenta WMUIT é possível colocar duas visões lado a lado para observar a relação entre elas, como mostrado na Figura 5.12. Nessa figura foi realizada a seleção de uma página no lado direito. No lado esquerdo da tela a mesma página e os componentes e estados que pertencem a ela são destacados.

Figura 5.13: Exemplo de coordenação do diagrama de interação com outras visões. No lado esquerdo é mostrado o mesmo diagrama de interações apresentado na Figura 5.9. No lado direito é mostrado o diagrama de navegação do mesmo site, com todas as páginas que existem no diagrama de interação na área em destaque. Ou seja, o usuário não acessou a página *Posto Kaó - Localização* (cor roxa) por ela não estar na área em destaque.

No diagrama de interações (Figura 5.9) são apresentados apenas os elementos

que o usuário interagiu. Por conta disso, quando essa visão é coordenada com outras, os itens em comum são identificados por uma área em azul, como mostrada na Figura 5.13 - lado direito. Nessa figura, apenas a página *Posto Kaó - Localização* (roxo) não está na área em azul (lado direito), pois ela não aparece no diagrama de interações.

5.2 Definição do experimento

O objetivo dos experimentos realizados consistiam em analisar a compreensão da estrutura e dos componentes da interface apresentados pela ModelUI$_{VIZ}$, comparando a abordagem visual com a abordagem manual, à respeito da quantidade correta de componentes identificados, da perspectiva do experimentador, realizado no laboratório de pesquisa da Universidade Estadual Paulista - Unesp, campus de Presidente Prudente.

Os modelos de diálogo e de comportamento foram escolhidos como artefatos de criação dos experimentos por apresentarem informações mais detalhadas que o modelo navegacionais. A representação do modelo navegacional na ModelUI$_{VIZ}$ por ser muito explícita deixaria em desvantagem os participantes que utilizariam a abordagem manual.

As abordagens utilizadas para comparação se referem à:

- Abordagem manual: a partir do código-fonte (HTML e *JavaScript*) das páginas de um *site* online, os participantes criam os modelos de interface de diálogo e de comportamento.
- Abordagem visual: a partir dos arquivos XML – gerados pelo *plug-in* WMUID com o conteúdo das páginas de um *site* – e com o apoio da WMUIT, os participantes criam os modelos de interface de diálogo e de comportamento com base na ModelUI$_{VIZ}$.

Com base nesse objetivo, as seguintes hipóteses foram formuladas:

- **Hipótese Nula (H_0^1):** A utilização da ModelUI$_{VIZ}$ não provê auxílio a compreensão da interface do usuário.
- **Hipótese Alternativa (H_1^1):** A utilização da ModelUI$_{VIZ}$ provê auxílio a compreensão da interface do usuário.

Para análise dos modelos de interface criados pelos participantes (diálogo e comportamento) foram desenvolvidos indicadores quantitativos para cada um dos modelos. Assim, os indicadores para o modelo de diálogo são:

1. Quantos *links* internos foram identificados corretamente?
2. Quantos *links* externos foram identificados corretamente?

3. Quantos componentes (diferentes de *links*) foram identificados corretamente?

Os indicadores para o modelo de comportamento são:

1. Quantos *links* possuem os estados e eventos declarados corretamente?
2. Quantos *links* para páginas externas foram criados?
3. Os *links* estão conectados com outras páginas (direcionando para o estado *onload*?) – Sim/Não
4. Quantos estados e eventos de outros componentes (diferentes de *links*) foram adicionados corretamente?
5. Foi atribuído o histórico para os *links*? – Sim/Não
6. Foram criados os estados das páginas externas (ao menos *onload*/ *load*)? – Sim/Não
7. Os componentes foram colocados dentro do estado *load* da página? – Sim/Não

Para cada *site* selecionado como artefato para os experimentos foram calculados os valores para cada um dos indicadores. Assim, a variável independente dos experimentos realizados é a soma dos componentes existentes para cada um dos indicadores em cada uma das páginas do *site*. Ou seja, $Valor_m = \sum_1^n(\sum_1^c(Valor_p))$, sendo m é o modelo de interface analisado (diálogo ou comportamento), c é o número de indicadores para cada modelo, p é a referência para cada página e n é o número total de páginas.

As variáveis dependentes desse experimento são: a quantidade de componentes identificados corretamente. Com relação aos componentes, cada participante deve identificar o máximo de componentes de maneira correta utilizando a abordagem manual e a visual. A formula utilizada para calcular os acertos para cada modelo é: $Valor_i = \sum_1^n(\frac{\sum_1^c(Valor_p)}{\sum_1^c(Valor_pOraculo)}) * n$, onde i é o identificador de cada participante, c é o número de indicadores para cada modelo analisado, p é o identificador de cada página do *site* e n é o total de páginas analisadas pelo participante.

Tabela 5.1: Exemplo de tabela de valores para o oráculo do modelo de diálogo

	Link interno	**Link externo**	**Outros componentes**	**Total**
Página 1	15	10	10	**35**
Página 2	15	5	5	**25**
Página 3	10	5	5	**20**

Na Tabela 5.1 é apresentado um exemplo dos valores identificados para um *site* do modelo de diálogo. Na Tabela 5.2 é apresentado um exemplo dos valores associados ao modelo de diálogo criado por um participante. Para realizar o calculo é feita a proporção para cada página criada pelo participante no modelo e multiplicado pela

Tabela 5.2: Exemplo de tabela de valores de um participante para o modelo de diálogo

	Link interno	**Link externo**	**Outros componentes**	**Total**
Página 1	10	5	5	**20**
Página 2	10	5	5	**20**
Página 3				

quantidade de páginas criadas pelo participante, o resultado para o exemplo acima é: $Total = (20/35 + 20/25) * 2 = 2,743$.

Os participantes foram convidados em aulas das turmas de 2º e 3º ano de Ciência da Computação. Ao chegarem para o experimento eles eram direcionados para os grupos de acordo com a ordem de chegada, para que houvesse um equilíbrio nos grupos. Assim, os participantes iniciavam a criação dos modelos de interface utilizando a abordagem manual ou a abordagem visual, de acordo com seu grupo. Na segunda sessão do experimento, as abordagens eram trocadas, como apresentado na Tabela 5.3.

Tabela 5.3: Organização das sessões do experimento

	Grupo 1	Grupo 2
Site A	Abordagem Manual	Abordagem Visual
Site B	Abordagem Visual	Abordagem Manual

5.2.1 Artefatos comuns aos experimentos

Os artefatos utilizados nos experimentos foram definidos de acordo com os requisitos necessários para a execução de cada uma das sessões dos experimentos, que são: (i) Arquivos XML gerados pelo *Crawler* e *Tracer* do *plug-in* WMUID (específicos para cada *site*), (ii) Softwares (relacionados na Tabela 5.4), (iii) Formulários (relacionados na Tabela 5.5), (iv) Materiais de treinamento e Planilhas de registro e controle.

Tabela 5.4: Softwares utilizados para realização dos experimentos

Software	**Utilidade**
Editor UML	Escrever os modelos de diálogo e de comportamento
WMUIT	Realizar a leitura dos arquivos XML gerados pela WMUID
Programa descompactador	Para descompactar os arquivos XML disponibilizados para download
Navegador *Google Chrome* ou *Mozila Firefox*	Executar a ferramenta WMUIT.

Os artefatos foram criados e planejados para que os resultados dos experimentos fossem os mesmos, independente de sua utilização nos experimentos. Para que o tempo de criação dos arquivos XML gerados pelo *plug-in* WMUID não afetassem o desenvolvimento dos modelos, esses arquivos eram criados previamente ao início do experimento.

Foram desenvolvidos alguns formulários para colaborar com a documentação do experimento, coletando informações sobre os participantes, o registro formal das atividades realizadas pelos participantes, o termo de consentimento necessário, bem como uma avaliação da ModelUI$_{VIZ}$ e o treinamento realizado. A relação dos formulários é apresentada na Tabela 5.5.

Tabela 5.5: Formulários utilizados para a documentação dos experimentos.

Documento	Objetivo
1	Termo de consentimento livre e esclarecido
2	Planejamento do experimento (para acompanhamento do condutor)
3	Perfil dos participantes
4	Formulário de acompanhamento do Site A
5	Formulário de acompanhamento do Site B
6	Formulário final

Os formulários de acompanhamento do *Site* A e do *Site* B (Tabela 5.5 – itens 4 e 5) foram utilizados para registrar o tempo de criação dos modelos pelos participantes e são apresentados no Apêndice A.1.1. O formulário final (Tabela 5.5 – item 6), apresentado no Apêndice A.1.2, foi utilizado para que o participantes pudesse avaliar o treinamento e a ModelUI$_{VIZ}$.

5.3 1º experimento controlado

Como dito anteriormente, foi realizado um 1º experimento para avaliar os artefatos criados e o tempo total para a realização do experimento. Por isso, convém salientar que esse experimento não teve como objetivo obter resultados estatísticos, mas sim o objetivo de melhorar o projeto do experimento e os artefatos.

O 1º experimento foi conduzido conforme apresentado na Tabela 5.6 e foram utilizados os artefatos apresentados na Seção 5.2.1. O cronograma do experimento foi dividido em 3 dias, nos períodos da manhã e noite. Foram realizados treinamentos pertinentes a cada um dos conteúdos relevantes e exercícios para fixação dos conteúdos.

Nesse experimento houve 10 participantes, sendo a maioria alunos da graduação em Ciência da Computação (n = 9) e apenas um com graduação completa em Ciência da Computação. O perfil geral dos participantes é:

- A metade dos participante não conhecia os modelos de interface. A outra metade afirmou possuir pouco conhecimento sobre os modelos de UI.
- Um deles já utilizou no trabalho ou em pesquisa acadêmica.
- A maioria (6 de 10) já desenvolveu uma interface *web*. O restante nunca desenvolveu para o ambiente *web*.
- Todos possuem conhecimento em UML.

Tabela 5.6: Cronograma do experimento

Dia	Atividade	Tempo
1	Introdução	20 min
	Preenchimento de perfil	20 min
	Treinamento - Modelos de Interface	50 min
	Exercício 1 - modelo de diálogo	90 min
	Treinamento - Visualização de Informação	40 min
2	Exercício 2 - modelo de comportamento	60 min
	Treinamento - ModelUI$_{VIZ}$	40 min
	Exercício 3 - modelo de diálogo com a ModelUI$_{VIZ}$	50 min
	Exercício 4 - modelo de comportamento com a ModelUI$_{VIZ}$	50 min
3	Aplicação 1 - Site A - modelo de Diálogo	45 min
	Aplicação 1 - Site A - modelo de comportamento	45 min
	Preenchimento de formulário sobre sessão 1	10 min
	Aplicação 2 - Site B - modelo de Diálogo	45 min
	Aplicação 2 - Site B - modelo de comportamento	45 min
	Preenchimento de formulário sobre sessão 2	10 min
	Formulário final	10 min

- Metade dos participantes (5 de 10) nunca viram a aplicação das técnicas da VisInfo. A outra metade já viu a aplicação de técnicas da VisInfo em uma ferramenta.
- Dentre os que viram a aplicação de técnicas, 2 participantes nunca interagiram com a ferramenta.

Para os exercícios foi utilizado o *site* do *Hotel Kaó* (Posto Kaó, 2016) como base. Para as sessões do experimento foram selecionados os *sites* mostrados na Tabela 5.7 como artefatos e no Apêndice A.2.

Tabela 5.7: Estrutura dos *sites* selecionados para o 1º experimento.

	Números de páginas	Itens do Modelo de Diálogo	Itens do Modelo de Comportamento
Site A	5	101	121
Site B	8	433	450

Houve uma taxa de abandono de 55 % do início do treinamento até o último dia. No entanto, foram contabilizados apenas os participantes que vieram em todos os dias e, por isso, não há um equilíbrio na distribuição dos participantes nos grupos, apresentada na Tabela 5.8.

Tabela 5.8: Números de participantes por grupos para o 1º experimento.

Grupo	Número de participantes
1	4
2	6

5.3.1 Resultados do 1º experimento

Nesse experimento a diferença de tamanho dos *sites* selecionados como artefatos teve uma grande influência nos resultados. Como mostrado na Tabela 5.7, o *Site* B possui uma quantidade de componentes quase 4 vezes maior que o *Site* A. Na análise estatística dos resultados do experimento, o tamanho dos sites influenciou para que a abordagem visual obtivesse melhores resultados que a abordagem manual – utilizando Teste-F para análise de hipóteses (Wohlin et al., 2012). Porém, analisando qualitativamente os modelos criados e os gráficos de caixa (*box-plot*), percebe-se que os participantes tiveram melhores resultados com a abordagem manual. Assim, nos próximos experimentos a diferença de tamanho dos *sites* foi observada com maior atenção.

Na Figura 5.14 e 5.15 são mostrados os valores de acertos (eixo y) para cada um dos modelos de separados por grupos. Com os resultados obtidos foi observada uma dificuldade na criação do modelo de diálogo em comparação ao modelo de comportamento. Na Figura 5.14 do modelo de diálogo foram identificados menos componentes em comparação com a Figura 5.15 do modelo de comportamento. Isso pode ser atribuído ao fato do modelo de comportamento ser mais conhecido aos participantes do que o modelo de diálogo, por ser mais utilizado na Engenharia de Software.

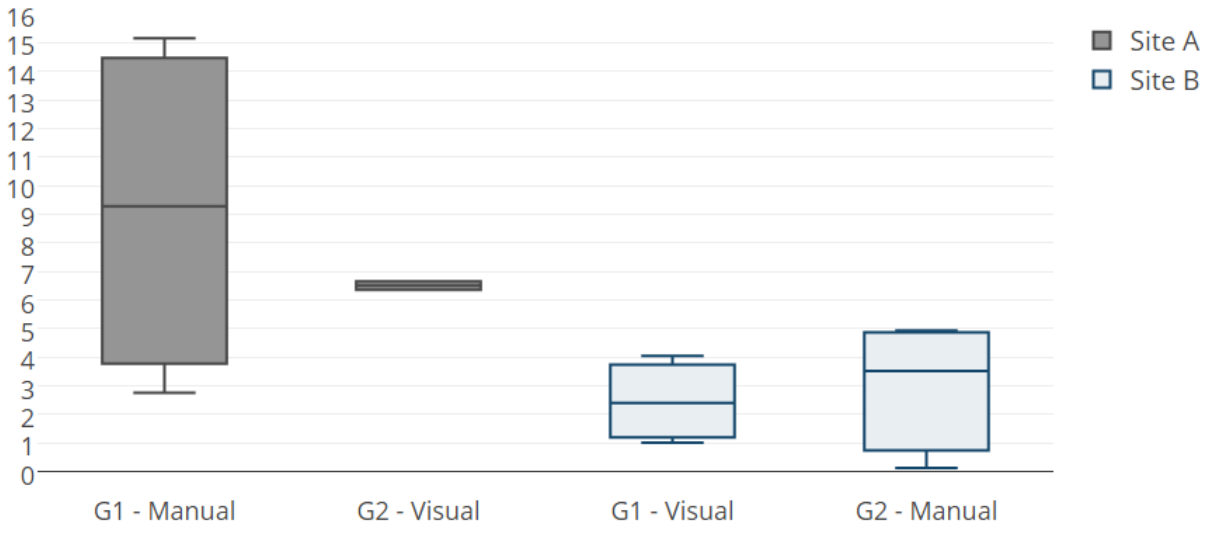

Figura 5.14: Resultado do 1º experimento com relação ao modelo de diálogo.

Com relação ao modelo de diálogo, pode-se observar uma identificação muito baixa do grupo 2 (G2), que iniciou as sessões do experimento utilizando a abordagem visual, em comparação ao grupo 1 (G1) para o *Site* A (Figura 5.14). O principal motivo dessa diferença foi a dificuldade em identificar componentes diferentes de *links* e diferenciar os *links* internos dos *links* externos. Além disso, os participantes do grupo 1 (G1) que iniciaram com a abordagem manual relataram uma dificuldade na identificação dos componentes com a abordagem visual para o *Site* B. Isso ocorreu pela

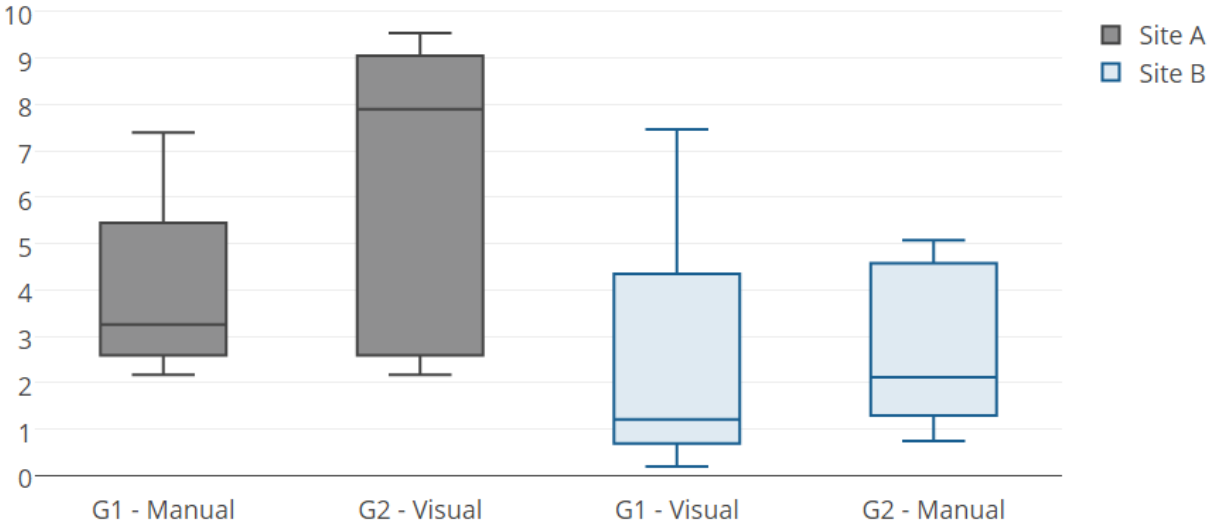

Figura 5.15: Resultado do 1º experimento com relação ao modelo de comportamento.

maior quantidade de componentes do *Site* B em comparação ao *Site* A e pela falta de filtros para facilitar a interação com a ModelUI$_{VIZ}$.

Com relação ao modelo de comportamento, o uso da abordagem visual teve maior influência na identificação dos componentes dos *sites*, como mostrada na Figura 5.15. Essa influência pode ser observada no grupo 2 (G2), que usou a abordagem visual na primeira sessão do experimento e depois a abordagem manual na segunda sessão. No entanto, os problemas de identificação dos componentes no modelo de diálogo, descritos anteriormente, também ocorreram no modelo de comportamento.

5.3.2 Lições aprendidas

A informação que diferenciava os *links* entre internos e externos era apresentada apenas como texto no painel de detalhes dos objetos (Figura 5.10). Ou seja, para obter a informação se determinado elemento da categoria *link* era interno ou não, ele deveria ser selecionado individualmente. Em *sites* com uma grande quantidades de elementos do tipo *link*, como o *Site* B, a ação de seleção e identificação de cada um dos elementos se torna mais difícil. Para facilitar a diferenciação entre os *links* internos e os *links* externos foi adicionada a forma triangular para representar os *links* externos. Assim, a mesma cor é utilizada para representar a categoria de *links*, sendo diferentes as formas para os *links* internos (quadrado) e os *links* externos (triângulo). Também foi adicionado o filtro para cada um dos itens existentes na representação visual, reduzindo a carga de informações nas representações. Essas modificações foram realizadas na Versão 1 da ferramenta WMUIT e a ModelUI$_{VIZ}$ para a Versão 2, apresentada na Seção 5.4.

Com relação a estrutura do experimento, pela alta taxa de desistência durante o treinamento (55% de desistência) foram realizadas modificações na estrutura do treinamento, para que fosse realizado em apenas 1 dia. Foram retirados os exercícios

de apoio do treinamento, sendo descritos com mais detalhes os exemplos nas apresentações. No entanto, por conta dos exercícios no treinamento é possível perceber que não houve o efeito de aprendizagem durante as sessões de aplicação do experimento. Assim, pela retirada dos exercícios o efeito de aprendizagem pode ser maior nos experimentos futuros.

Nesse experimento os *sites* selecionados como objetos de estudo tiveram um grande impacto na análise estatística. Por isso, eles foram alterados para outros *sites* com tamanhos similares nos experimentos futuros.

O resultado dessas mudanças na estrutura do experimento e na Versão 2 da WMUIT e da ModelUI$_{VIZ}$ são apresentados na próxima seção.

5.4 ModelUI$_{VIZ}$ - Versão 2

Na segunda versão da ModelUI$_{VIZ}$ foi adicionada uma função de filtro na legenda e uma nova forma na ModelUI$_{VIZ}$.

A função de filtro foi adicionada para melhorar a compreensão principalmente da visão por páginas. É comum nessa visão uma densidade muito alta de componentes, eventos e estados apresentados em uma página. Para melhorar a compreensão da ModelUI$_{VIZ}$ foi adicionado um filtro para cada um dos itens da legenda. Nesta versão não foi adicionada a possibilidade de mostrar ou ocultar todos os itens de uma categoria.

Na Versão 1 da ModelUI$_{VIZ}$ essa informação ficava disponível apenas no painel de detalhes do elemento selecionado (Figura 5.10). Ou seja, a informação de *link* interno ou externo ficava disponível apenas na seleção do componente *link*. Por isso, foi adicionada a forma de triângulo para representar os links externos, mantendo a cor de identificação da categoria *link*, como apresentado na Figura 5.16.

Figura 5.16: Trecho da legenda que mostra todos os componentes do *site* Kaó Hotel.

5.5 2º experimento controlado

O segundo experimento controlado teve como objetivo avaliar estatisticamente a ModelUI$_{VIZ}$ após a implementação das modificações da Versão 2, descritas na Se-

ção 5.4.

Considerando a experiência obtida no 1º experimento, foram realizadas modificações na estrutura desse experimento para que fosse realizado em apenas 1 dia, como mostrada na Tabela 5.9. Os exercícios foram retirados do treinamento, mas os exemplos de fixação para os modelos de interface foram descritos com mais detalhes. O tempo para a criação dos modelos nas sessões foi mantido para avaliar quantos componentes foram identificados com um tempo limitado.

Tabela 5.9: Cronograma do 2º experimento.

Dia	Atividade	Tempo
1	Introdução	10 min
	Preenchimento de perfil	10 min
	Treinamento - Modelos de Interface	30 min
	Treinamento - Visualização de Informação	30 min
	Treinamento - ModelUI$_{VIZ}$	40 min
	Aplicação 1 - Site A - modelo de Diálogo	45 min
	Aplicação 1 - Site A - modelo de comportamento	45 min
	Preenchimento de formulário sobre sessão 1	10 min
	Aplicação 2 - Site B - modelo de Diálogo	45 min
	Aplicação 2 - Site B - modelo de comportamento	45 min
	Preenchimento de formulário sobre sessão 2	10 min
	Formulário final	10 min

Por conta dos problemas identificados nos *sites* utilizados no 1º experimento, foram selecionados outros *sites* para o 2º experimento. Eles são apresentados na Tabela 5.10 e no Apêndice A.3. Os outros artefatos utilizados no experimento foram apresentados na Seção 5.2.1.

Tabela 5.10: Estrutura dos *sites* selecionados para o 2º experimento.

	Números de páginas	**Itens do Modelo de Diálogo**	**Itens do Modelo de Comportamento**
Site A	10	607	647
Site B	15	497	557

Nesse experimento houve 11 participantes, sendo a maioria alunos da graduação em Ciência da Computação (n = 9) e apenas um possui graduação completa em Ciência da Computação. Salientando que os participantes desse experimento não participaram do 1º experimento, o perfil dos participantes é:

- A maioria dos participantes (8 de 11) afirmaram que conhecem um pouco sobre os modelos de interface e já utilizou para algum trabalho acadêmico ou profissional.
  - 2 de 11 afirmaram que conhecem sobre modelos de interface, porém nunca os utilizaram.

  - Apenas um participante nunca tinha ouvido falar sobre esse tipo de modelo.

- A maioria dos participantes (7 de 11) já desenvolveram uma interface *web* para a área acadêmica ou profissionalmente.
- Todos os participantes possuem conhecimentos sobre UML.
- A maioria (8 de 11) já interagiu com uma ferramenta InfoVis.
  - Do restante 1 participante já viu uma ferramenta de InfoVis, mas nunca interagiu e 2 participantes nunca viram uma ferramenta de InfoVis.
- 3 dos 11 participantes já aplicaram uma ou mais técnicas de InfoVis para fins acadêmicos.

Os participantes foram distribuídos aos grupos de acordo com a ordem de chegada, de modo aleatório. Assim, a distribuição dos participante nos grupos é apresentada na Tabela 5.11

Tabela 5.11: Números de participantes por grupos para o 2º experimento.

Grupo	Número de participantes
1	6
2	5

5.5.1 Resultados do 2ª experimento

Para a análise dos modelos de diálogo e de comportamento criados pelos participantes, foram utilizadas as categorias e a fórmula descritas na Seção 5.2.

Tabela 5.12: Resultados por participantes dos modelos de Diálogo e de Comportamento (Comp.) para cada um dos *sites* do 2º experimento.

	Site A			Site B		
Participantes	**Diálogo**	**Comp.**	**Abordagem**	**Diálogo**	**Comp.**	**Abordagem**
1	0,19	0,63	Manual	3,34	0,36	Visual
2	0,76	0,47	Manual	0,56	0,57	Visual
3	0,00	0,27	Manual	0,00	0,60	Visual
4	0,84	0,79	Manual	0,51	0,47	Visual
5	0,93	0,07	Manual	0,53	0,04	Visual
6	0,25	0,27	Manual	0,35	0,36	Visual
7	0,00	0,45	Visual	0,00	0,38	Manual
8	0,03	2,85	Visual	0,28	3,12	Manual
9	0,12	0,91	Visual	0,35	2,79	Manual
10	0,94	0,41	Visual	5,21	0,74	Manual
11	0,22	0,25	Visual	0,63	0,45	Manual

Pelos dados reportados na Tabela 5.12 e apresentados nas figuras 5.17, 5.18 e 5.19 é possível observar que os participantes não reportaram mais componentes

utilizando a abordagem visual. Para os modelo de diálogo (Figura 5.17) foram reportados menos componentes que o modelo de comportamento (Figura 5.19). Um dos fatores identificados por essa diferença é a maior familiaridade dos participantes com o modelo de comportamento. Esse fator pode ser identificado nos participantes 3 e 7, que tiveram dificuldades de identificar e reportar as informações apenas para o modelo de diálogo. Essa dificuldade também foi observada no 1º experimento e por isso, as modificações realizadas no treinamento não podem ser apontados como único fator.

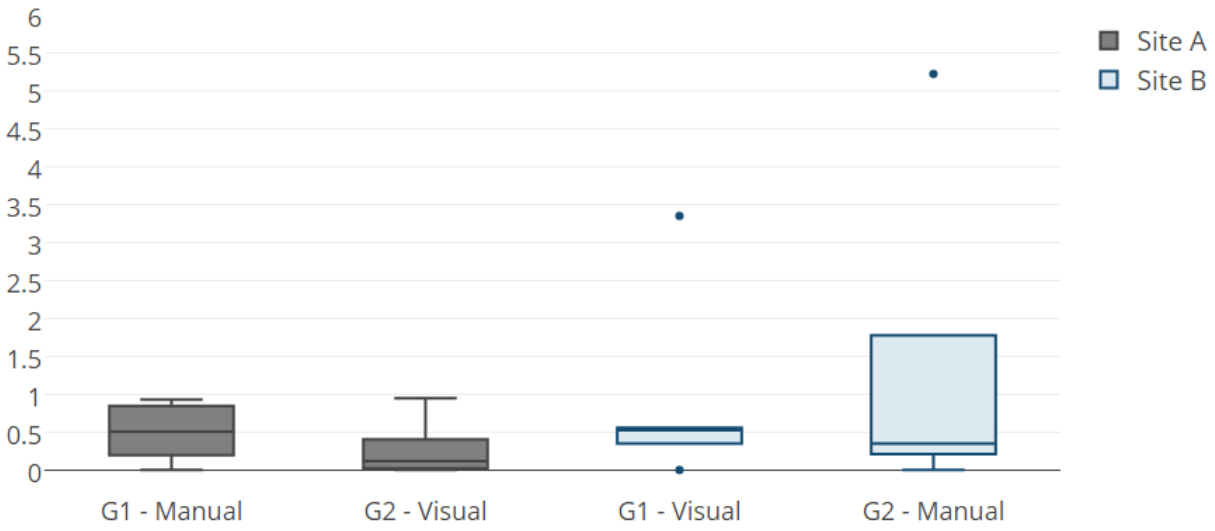

Figura 5.17: Resultado do 2ª experimento com relação ao modelo de diálogo.

Os *outliers* apresentados na Figura 5.17 são os valores discrepantes dos resultados dos participantes que estão fora do primeiro e último quartil do gráfico de caixa. Quando retirados os *outliers* referente ao *Site* B do modelo de diálogo é possível analisar como a média dos participantes foi na criação desse modelo. Na Figura 5.18 é possível observar com maior clareza que foram identificados mais componentes com a abordagem manual, independente da abordagem iniciada pelos grupos.

Nos resultados do modelo de comportamento, apresentado na Figura 5.19, é possível observar que, diferente dos resultados apresentados no 1º experimento (Figura 5.15), a abordagem manual apresentou melhores resultados independente da abordagem iniciada pelos grupos.

Analisando qualitativamente os modelos criados pelos participantes foram identificados problemas com relação à compreensão da estrutura das páginas no uso da abordagem visual. Nesses modelos, os participantes não reportaram ou não foram replicadas estruturas comuns à todas as páginas, como menus.

Outro problema observado foi com relação à identificação a qual página do *site* um determinado componente pertence. Essa dificuldade foi reportada observando os modelos criados pelos usuários, mas principalmente, durante o uso da visão por páginas pelos participantes (Seção 5.1.1).

Com as mudanças realizadas na Versão 2, os problemas identificados no 1º experimento sobre a diferenciação entre os *links* internos e os *links* externos foram sanados.

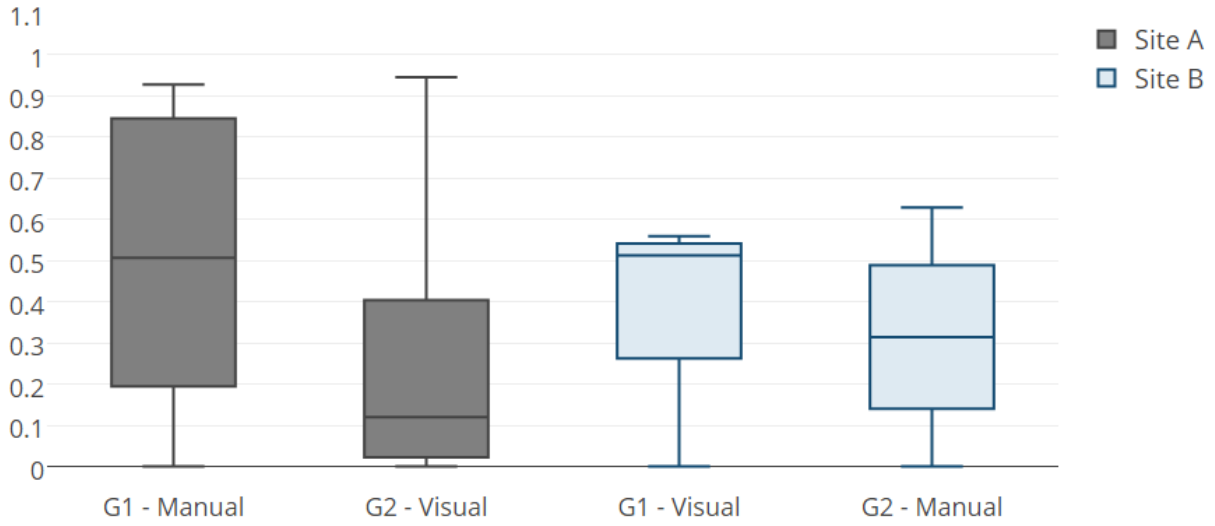

Figura 5.18: Resultado do 2ª experimento com relação ao modelo de diálogo, sem os *outliers*.

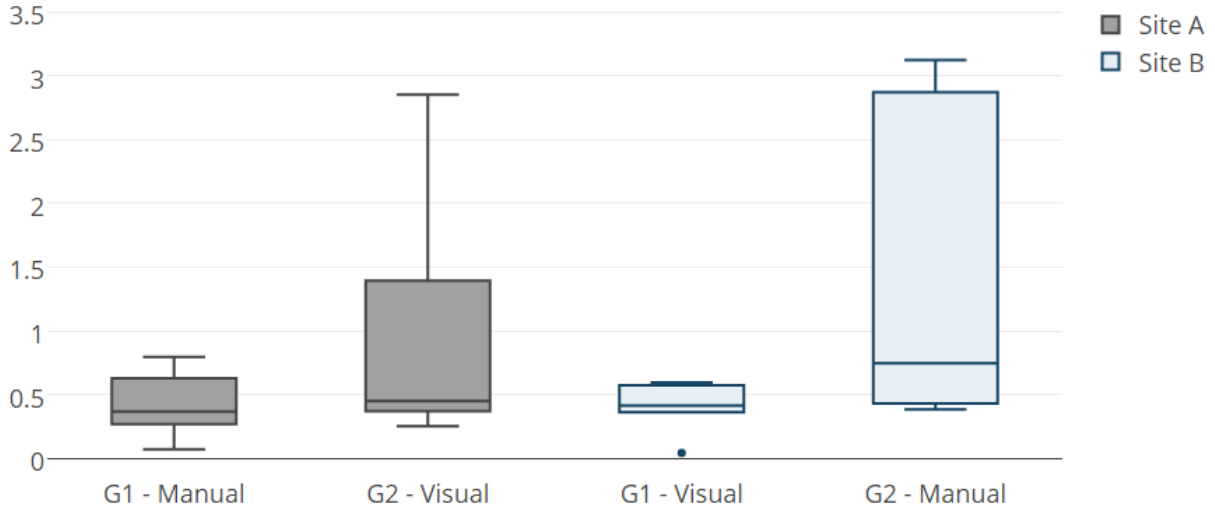

Figura 5.19: Resultado do 2ª experimento com relação ao modelo de comportamento.

Porém, os participantes manifestaram a necessidade de filtro pelas categorias gerais e não apenas pelos itens de cada categoria.

5.5.2 Lições aprendidas

De acordo com os resultados obtidos nesse experimento, podemos identificar dois fatores que dificultaram a criação dos modelos: i) a compreensão de uma nova representação – modelo diálogo e ii) a retirada dos exercícios de fixação do treinamento. Assim, para o futuro experimento, além de descrito com mais detalhes a composição dos modelos de diálogo e de comportamento, foram adicionados mais exemplos práticos, feitos durante a apresentação do conteúdo. O objetivo é sanar as principais dúvidas que surgiam durante a criação dos modelos com os exemplos práticos.

Na ModelUI$_{VIZ}$ foram adicionadas áreas coloridas visão por páginas para diferen-

ciar a quais páginas os componentes apresentados pertencem. A cor da área é a mesma cor da página, ou seja, se o componente estiver em uma área colorida, ele pertence a outra página da mesma cor de sua área. Nas outras visões foram realizados testes para adicionar as áreas coloridas nas representações, mas pela quantidade de páginas apresentadas (todas) as informações ficavam muito confusas e a proposta de colaboração era perdida.

No formulário final do experimento foi sugerido pelos participantes a criação de filtros por categorias gerais, como eventos, páginas, etc. Essa modificação além de melhorar o uso do filtro, também colabora para melhor identificação a qual página um componente pertence.

De acordo com a análise qualitativa, foi identificada a necessidade da criação de novas visões que apresentassem a estrutura das páginas e não apenas sua composição. Assim, foram criadas 4 novas visões com o objetivo de apresentar a estrutura das páginas e identificar a localização dos componentes interativos. Essas novas representações utilizam a seção *structure* do arquivo XML gerado pelo *crawler*.

As modificações realizadas na ferramenta WMUIT e na ModelUI$_{VIZ}$ da versão 2 para a versão 3 estão descritas na Seção 5.6.

5.6 ModelUI$_{VIZ}$ - Versão 3

Figura 5.20: Trecho da legenda com toda a categoria de componentes e com a página *Posto Kaó - Infraestrutura* oculta nas visões.

Na terceira versão da ModelUI$_{VIZ}$ foi adicionado o filtro por categorias na legenda, sendo possível o mostrar ou ocultar toda uma categoria ou apenas um item de uma categoria, como mostrada na Figura 5.20 e o resultado do filtro na Figura 5.21.

A visão de páginas foi modificada para adicionar uma área em torno dos componentes que não pertencem aquela página. A área possui a cor da página à qual o componente pertence. Na Figura 5.22 é mostrado um exemplo dessas modificações

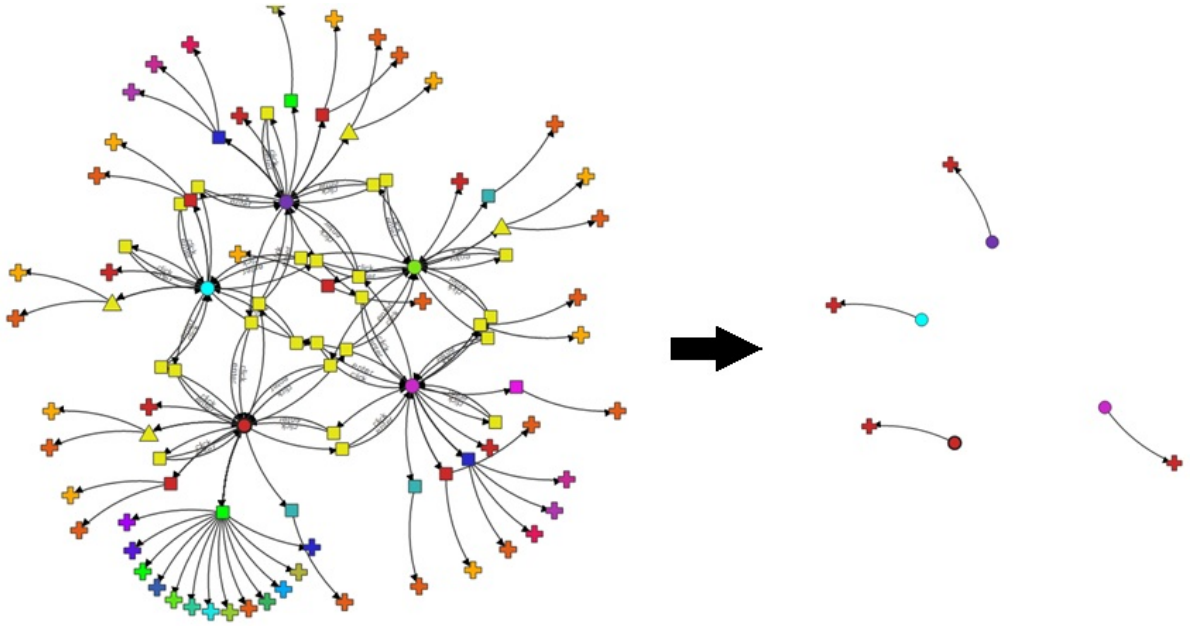

Figura 5.21: Resultado da aplicação do filtro apresentado na Figura 5.20.

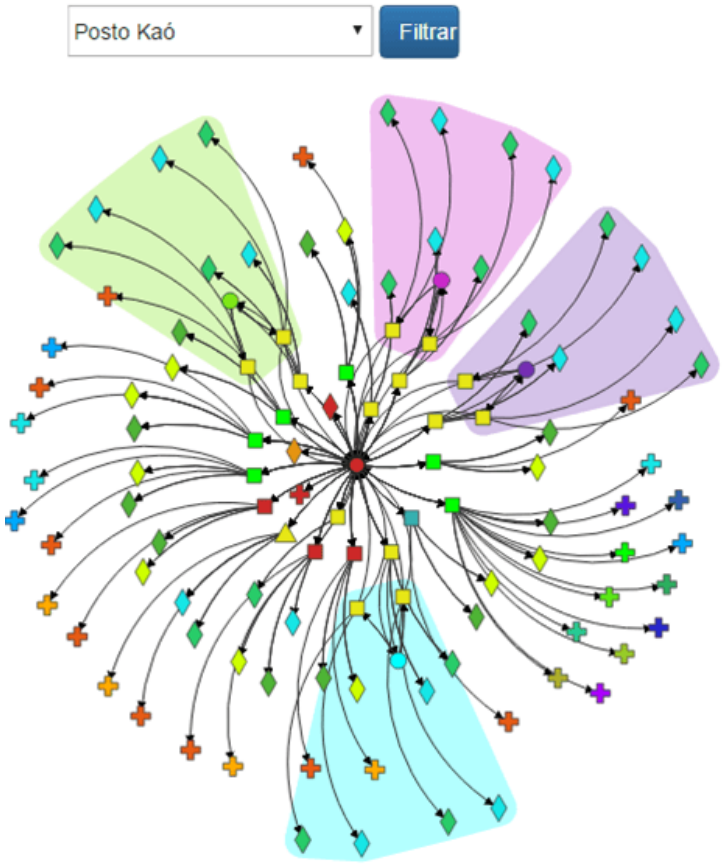

Figura 5.22: Existem 4 páginas com componentes *links* que direcionam para a página *Posto Kaó*.

comparando com a Figura 5.4.

Nessa versão foram criadas 4 visões estruturais para mostrar a localização dos componentes nas estruturas das páginas. Foram criadas as visões de Pacote em círculos, *Treemap*, Agrupamento Hierárquico e Agrupamento Hierárquico Radial. Es-

sas visões foram adicionadas para facilitar o entendimento de como as páginas foram criadas e como estão dispostos todos os componentes.

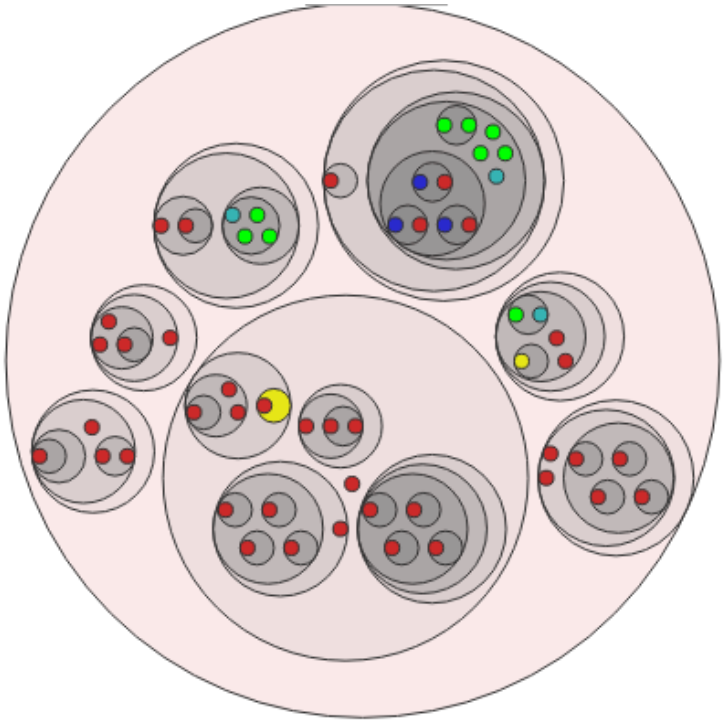

Figura 5.23: Visão estrutural da página utilizando a representação de Pacotes em Círculos (*Circle Packing*).

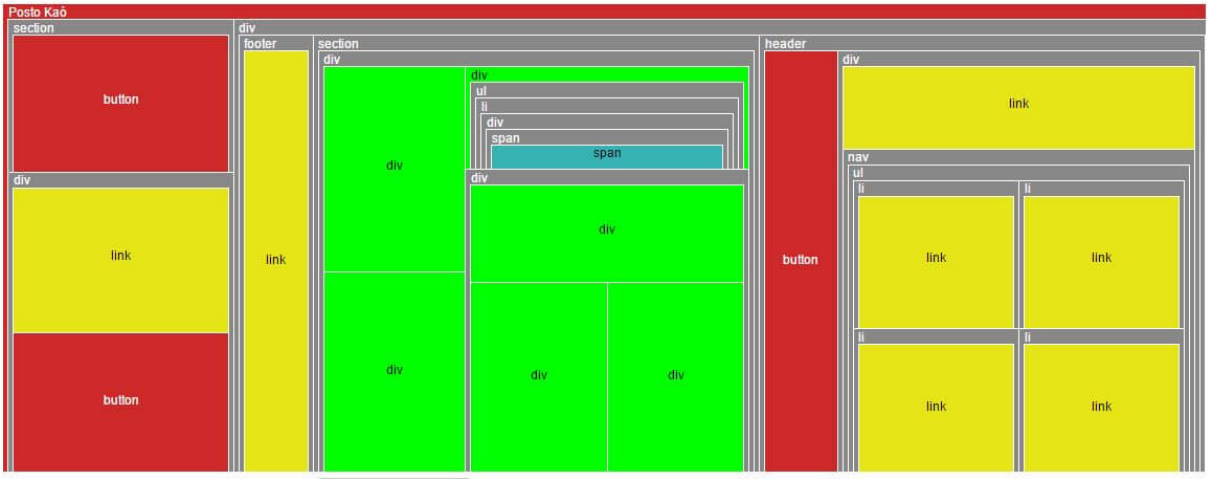

Figura 5.24: Visão estrutural da página para utilizando a representação *Treemap*.

A visão de Pacote em círculos foi criada com base na representação *Circle Packing* da bibliteca D3.js [1]. Essa representação e o *Treemap* na D3.js não permitem que a forma dos componentes seja alterada. Assim, apenas as cores identificam a localização dos componentes na estrutura das páginas, como mostrada na Figura 5.23 e na Figura 5.24, que mostram a mesma página com representações diferentes. Essas representações exigem um esforço cognitivo maior que as outras representações

[1] https://bl.ocks.org/mbostock/7607535

por utilizarem apenas a cor como identificação, sendo que a ModelUI$_{VIZ}$ possui uma combinação de cor e formas na sua especificação.

As visões de Agrupamento Hierárquico e Agrupamento Hierárquico Radial foram criadas com base em uma árvore e a técnica hierárquica de agrupamento de linhas (*Hierarchical Edge Bundle*) [2] (Seção 3.2.2.1). Nessas representações as formas e cores dos componentes e das páginas se mantêm iguais a legenda, como mostrado na Figura 5.25.

Uma vantagem dessa adaptação de representações é a identificação dos elementos pela categoria e pela cor, aumentando a acessibilidade dessa representação. Nessas visões, a linha azul identifica a conexão entre os componentes e as páginas. Quando um componente com a linha azul é selecionado a linha muda a cor para vermelha, identificando para quais páginas ela direciona, como mostrado na Figura 5.26. Quando uma página que possui as linhas azuis é selecionada as linhas mudam de cor para verde, identificando quais componentes apontam para aquela página.

5.7 3º experimento controlado

Para o 3º experimento foram realizadas pequenas modificações no material dos treinamentos sobre os modelos de interface, a Visualização de Informação e a ModelUI$_{VIZ}$. Essas modificações foram feitas para facilitar o entendimento dos participantes com relação aos modelos interface, sem o aumento dos dias para a realização dos treinamentos, como apresentado na Tabela 5.13.

Tabela 5.13: Cronograma do 3º experimento.

Dia	Atividade	Tempo
1	Introdução	05 min
	Preenchimento de perfil	10 min
	Treinamento - Modelos de Interface	50 min
	Treinamento - Visualização de Informação	25 min
	Treinamento - ModelUI$_{VIZ}$	50 min
	Preparação para os experimentos	10 min
	Aplicação 1 - Site A - modelo de Diálogo	45 min
	Aplicação 1 - Site A - modelo de comportamento	45 min
	Preenchimento de formulário sobre experimento 1	05 min
	Aplicação 2 - Site B - modelo de Diálogo	45 min
	Aplicação 2 - Site B - modelo de comportamento	45 min
	Preenchimento de formulário sobre experimento 2	05 min
	Formulário final	10 min

Apesar dos participantes desse experimento não serem os mesmos do 1º experimento e do 2º experimento, foram escolhidos *sites* diferentes aos experimentos anteriores. As informações sobre os *sites* selecionados para o experimento são apresentados na Tabela 5.14 e no Apêndice A.4. O restante dos artefatos utilizados no

[2] https://bl.ocks.org/mbostock/7607999

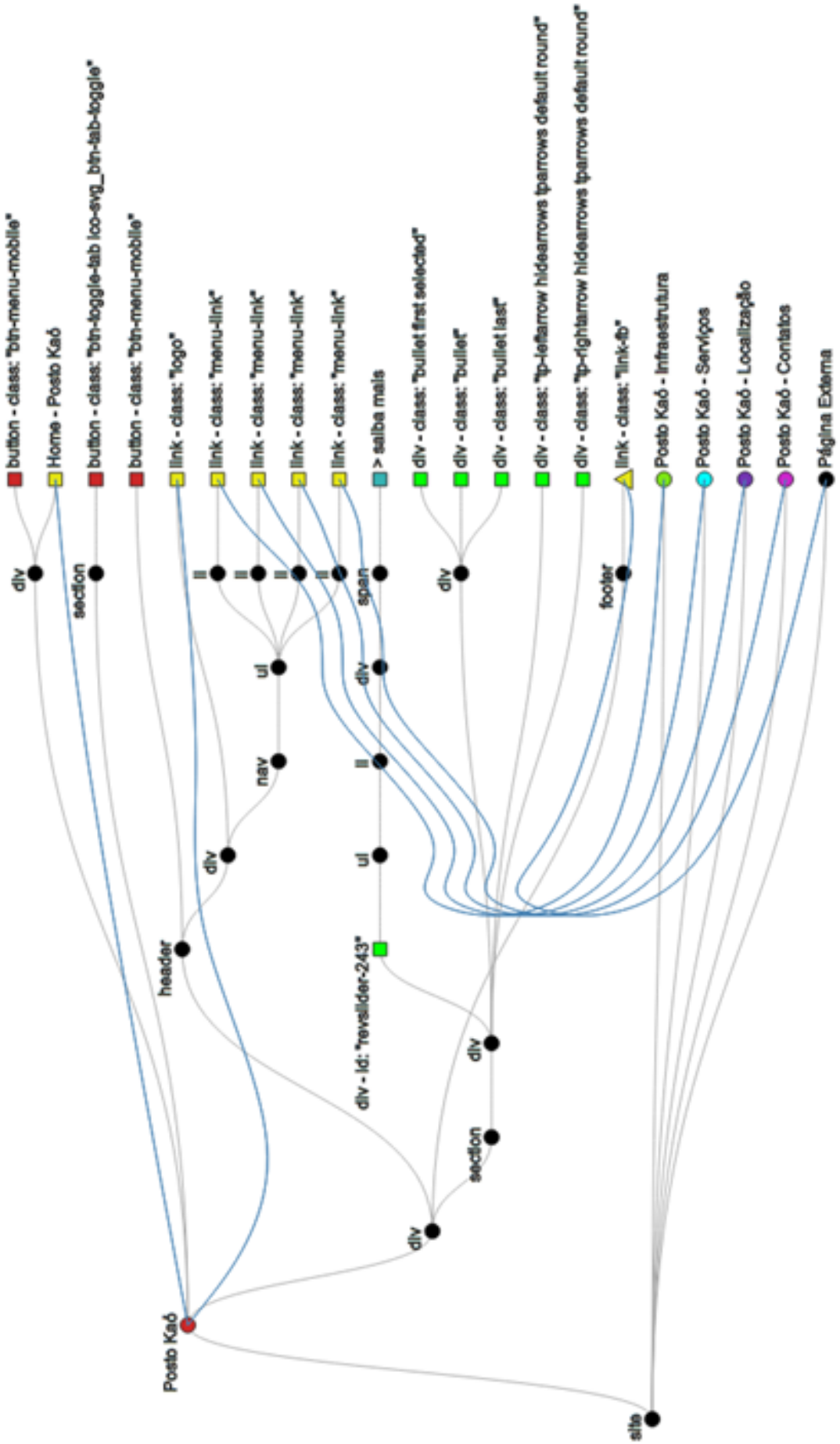

Figura 5.25: Visão estrutural da página inicial do *site* Posto Kaó, utilizando a representação em árvore e adaptação da técnica hierárquica de agrupamento de linhas.

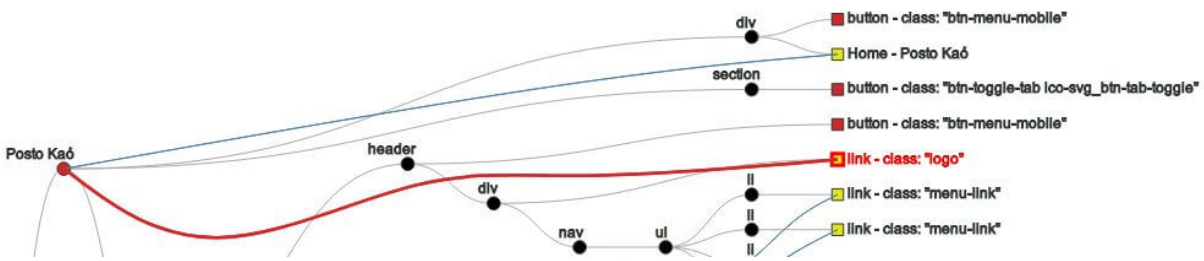

Figura 5.26: Trecho da visão de Agrupamento Hierárquico com o componente *link* selecionado.

experimento são apresentados na Seção 5.2.1.

Tabela 5.14: Estrutura dos *sites* selecionados para o 3º experimento

	Números de páginas	**Itens do Modelo de Diálogo**	**Itens do Modelo de Comportamento**
Site A	16	395	459
Site B	15	395	455

Os participantes foram distribuídos nos grupos de maneira aleatória, conforme a ordem de chegada para o experimento. A organização dos grupos é apresentada na Tabela 5.15. A distribuição dos participantes nos grupos não ficou equilibrada pois, algumas pessoas desistiram no meio do experimento e não foram contabilizadas no experimento.

Tabela 5.15: Números de participantes por grupos para o 3º experimento

Grupo	**Número de participantes**
1	7
2	11

Nesse experimento houve 18 participantes no total, sendo a maioria alunos da graduação em Ciência da Computação (n = 17) e um aluno do Mestrado em Ciência da Computação. Com relação ao perfil dos participantes:

- A maioria dos participantes (13 de 18) possuem algum conhecimento sobre modelos de interface. Dentre eles, 5 já utilizaram os modelos de interface para pesquisa acadêmica ou profissionalmente.
  - Apenas 5 de 18 não tinham conhecimento sobre modelos de interface.
- Apenas 1 participante nunca desenvolveu uma interface *web*.
- Todos possuem conhecimentos sobre UML.
- A maioria dos participantes (11 de 18) já viu a aplicação de técnicas de InfoVis em uma ferramenta. Desses 11, 4 nunca interagiram com a ferramenta e 3 estão desenvolvendo sua própria ferramenta de InfoVis.
  - O restante nunca viu ou interagiu com uma ferramenta InfoVis.

5.7.1 Resultados da 3º experimento

Para análise dos modelos de diálogo e de comportamento foram utilizadas os indicadores apresentados na Seção 5.2. Os resultados obtidos nos modelos de diálogo e de comportamento são apresentados na Tabela 5.16.

Tabela 5.16: Resultados por participantes dos modelos de Diálogo e de Comportamento (Comp.) para cada um dos *sites* do 3º experimento.

	Site A			Site B		
Participantes	**Diálogo**	**Comp.**	**Abordagem**	**Diálogo**	**Comp.**	**Abordagem**
1	0,54	2,73	Manual	0,37	1,20	Visual
2	0,14	0,10	Manual	0,42	2,95	Visual
3	0,31	0,67	Manual	14,56	46,74	Visual
4	7,50	9,19	Manual	8,67	9,66	Visual
5	0,92	0,26	Manual	3,23	10,98	Visual
6	0,37	0,54	Manual	0,47	0,78	Visual
7	4,34	2,31	Manual	4,32	3,12	Visual
8	0,29	0,05	Visual	0,32	0,66	Manual
9	0,43	0,03	Visual	0,37	0,04	Manual
10	0,91	0,77	Visual	5,71	0,58	Manual
11	2,48	2,23	Visual	3,65	3,28	Manual
12	0,00	26,45	Visual	4,14	14,55	Manual
13	0,31	0,23	Visual	2,29	0,39	Manual
14	0,09	6,77	Visual	0,32	0,35	Manual
15	3,09	0,51	Visual	0,95	7,20	Manual
16	0,43	2,49	Visual	1,95	1,97	Manual
17	0,66	0,28	Visual	1,00	0,91	Manual
18	0,23	1,11	Visual	0,53	1,64	Manual

No modelo de diálogo, foram identificados mais componentes para o segundo *site* (*Site* B), independentemente das abordagens utilizadas, como apresentado na Figura 5.27. Nessa figura, os grupos G1 e G2 identificaram mais componentes no *site* B que o *site* A. O principal motivo foi por não existirem os exercícios de fixação durante o treinamento e, assim, o conhecimento para a criação do modelo de diálogo foi adquirido durante a aplicação do experimento.

No entanto, mesmo com o efeito de aprendizado no modelo de diálogo, os participantes do G2 conseguiram identificar muito mais com a abordagem visual do que com a abordagem manual. Supondo apenas o efeito de aprendizagem como causa da diferença entre as abordagens, o resultado poderia ser um pouco maior ou igual a abordagem inicial. Por isso, é possível afirmar que a abordagem visual colaborou para que os participantes identificassem mais do que com a abordagem manual.

Pela familiaridade dos modelos de comportamentos da Engenharia de Software, o aprendizado adquirido durante as sessões do experimento foi menor em comparação ao modelo de diálogo. No modelo de comportamento a abordagem visual teve mais

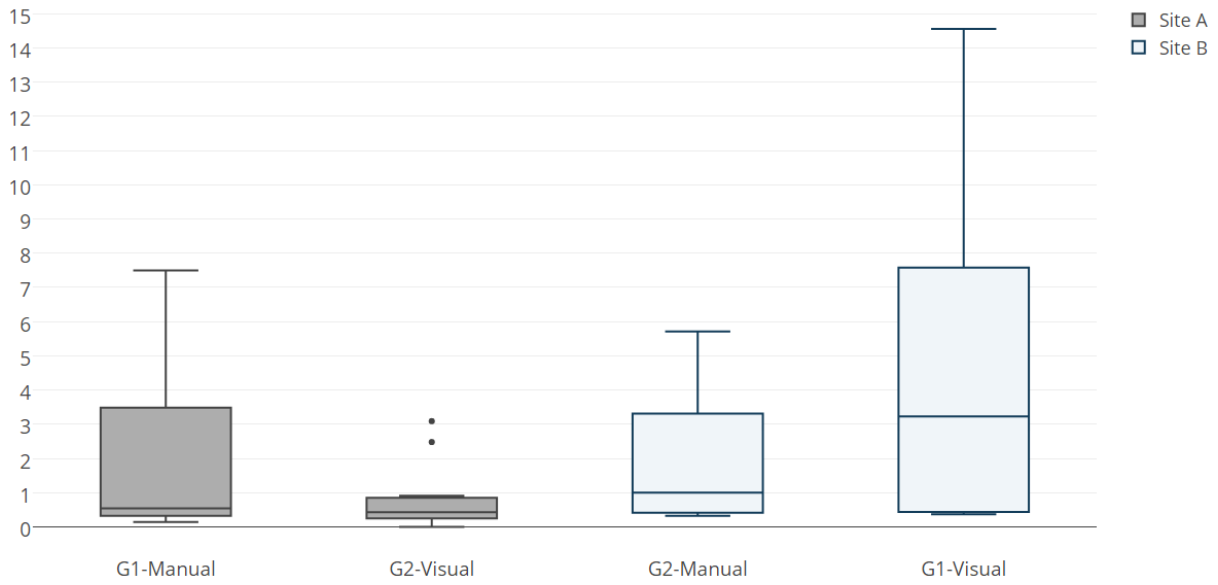

Figura 5.27: Resultado do 3ª experimento com relação ao modelo de diálogo.

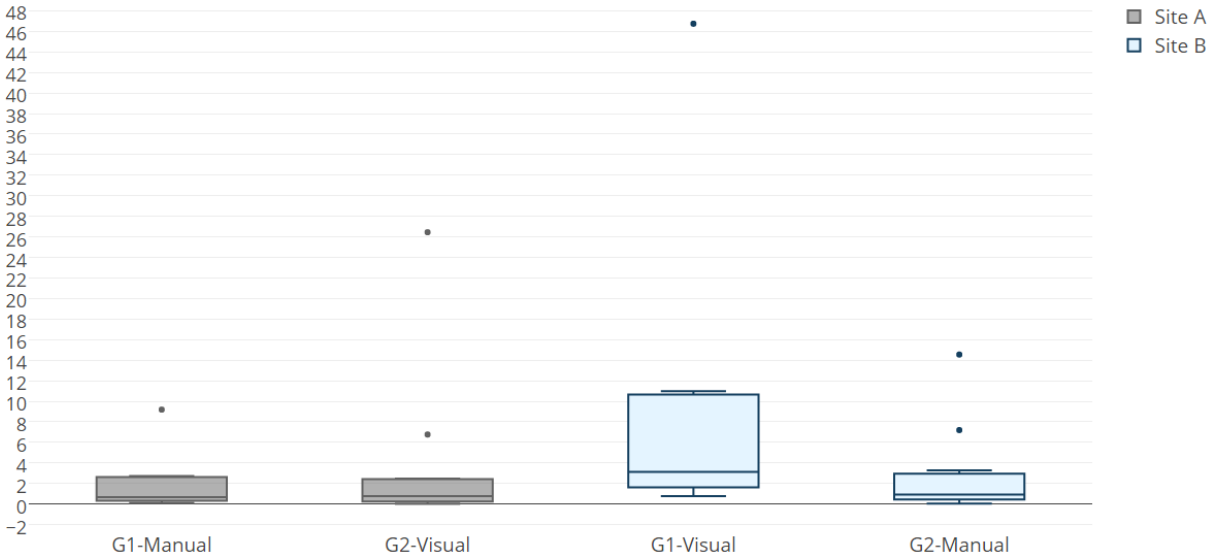

Figura 5.28: Resultado do 3ª experimento com relação ao modelo de comportamento.

impacto para a identificação de mais componentes da interface do que a abordagem manual, conforme apresentado na Figura 5.28.

Na Figura 5.29 são apresentados os resultados dos modelos de comportamento sem os *outliers*. Nessa figura é possível observar o quanto a abordagem visual facilitou a compreensão dos componentes da interface, principalmente para o grupo 1 (G1).

Em ambos modelos o grupo 1 (G1), que possui menos participantes que o grupo 2 (G2), teve melhores resultados com a abordagem visual. Isso demonstra que a diferença do número de participantes nos grupos não teve impacto nos resultados.

Na análise qualitativa dos modelos, o modelo de diálogo apresentou mais erros do que o modelo de comportamento. No modelo de diálogo, 50% (9 de 18) dos participantes cometeram erros usando apenas a abordagem manual. Sobre os outros participantes, 3 de 18 cometeram erros em ambas as abordagens e apenas 2 participantes

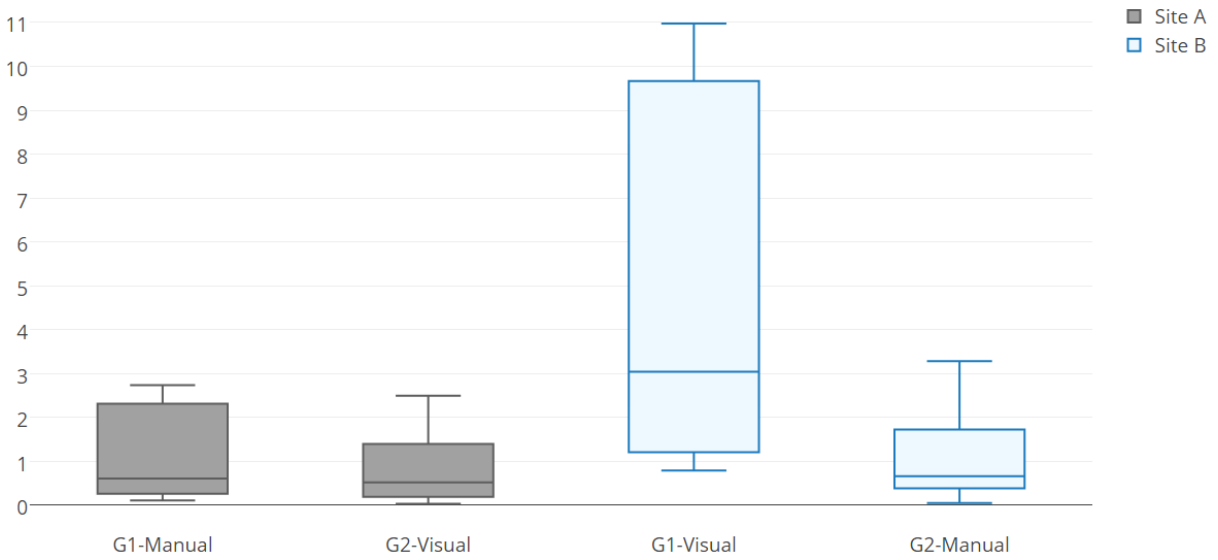

Figura 5.29: Resultado do 3ª experimento com relação ao modelo de comportamento, sem os *outliers*.

cometeram erros apenas com a abordagem visual. Para o modelo de comportamento foram identificados apenas 2 erros em ambas abordagens.

Nesse experimento foram realizados testes estatísticos para avaliar as hipóteses estabelecidas anteriormente. Pela quantidade de participantes no experimento foi utilizado o Teste-F para análise de hipóteses (Wohlin et al., 2012). Pelos resultados obtidos pelo Teste-F, apresentado na Tabela 5.17, é possível afirmar que existe uma diferença significativa entre os conjuntos de valores, para *p-value* igual a 0,05. Por ambos modelos apresentarem resultados abaixo de 0,5 para o *p-value* é possível rejeitar a hipótese nula (H_0^1).

Tabela 5.17: Resultado do Teste-F para o 3ª experimento controlado.

	Teste - F
Modelo de diálogo	0,01578
Modelo de comportamento	0,00001374

Pelos resultados apresentados no Teste-F é possível perceber que o modelo de comportamento teve melhores resultados que o modelo de diálogo. Ou seja, no modelo de comportamento teve mais elementos identificados corretamente que o modelo de diálogo. Assim, é possível afirmar que a ModelUI$_{VIZ}$ é efetiva para ambos modelos, mas com maior impacto para o modelo de comportamento.

Com as modificações realizadas na versão 2 e 3, os participantes identificaram *links* internos, *links* externos e outros componentes das páginas com maior facilidade. Em ambos modelos, os participantes diferenciaram os *links* internos dos *links* externos. Nos modelos de diálogo, 8 de 18 participantes identificaram mais componentes diferentes *links* usando apenas a abordagem visual. Os outros participantes (4 de 6) não identificaram nenhum outro componente diferente de *link*. O restante dos participantes identificaram esses componentes em ambas abordagens. Com relação ao modelo de

comportamento, apenas 3 dos 18 participantes identificaram componentes diferentes dos *links* usando a abordagem visual. Para o modelo de comportamento, o restante dos participantes identificou apenas os *links* (internos e externos). Esses resultados demonstram como a abordagem visual causou uma diferença positiva para a criação dos modelos.

5.7.2 Lições aprendidas

A avaliação dos modelos criados pelos participantes sofreu uma grande mudança nesse último experimento. Inicialmente, além das categorias utilizadas para avaliação dos modelos (descritas na Seção 5.2) foram incluídas algumas categorias repetidas. Por exemplo, ao invés de somar apenas a quantidade de componentes *links* corretos, eram somadas também as partes que pertenciam ao *link* separadamente, como eventos e estados. Ou seja, ao invés de existirem apenas 10 *links* corretos, existiam na soma final 10 *link* corretos + 10 eventos corretos + 20 estados corretos. O resultado final ficava muito distorcido pela contagem repetida dos elementos encontrados. Assim, as categorias de avaliação foram simplificadas para as apresentadas anteriormente.

Pelos resultados apresentados no experimento foi possível observar a importância do material de apoio estar bem detalhado e possuir o maior número de exemplos possíveis. Esse detalhamento colaborou para que os participantes entendessem melhor o processo de criação dos modelos.

Durante a condução do experimento foi possível observar que as novas visões desenvolvidas na Versão 3 colaboraram para melhor identificação dos componentes interativos e suas ligações. As visões Agrupamento Hierárquico (AH) e Agrupamento Hierárquico Radial (AHR) apresentam as descrições das categorias em conjunto às cores das categorias dos elementos, permitindo que as informações sejam diferenciadas independente das cores. Além disso, os participantes relataram que compreenderam melhor as conexões entre páginas utilizando as visões AH e AHR do que com a visão por páginas.

Durante o experimento não foi realizada uma contagem estatística com relação a qual visão foi mais utilizada pelos participantes. Mas pelos relatos e observações durante a execução das atividades do experimento, as visões AH e AHR foram mais utilizadas que a Pacote em Círculos e *Treemap*.

5.8 Ameaças à Validade

As ameaças sobre a validade de construção são relacionadas à análise do relacionamento entre o tratamento e o resultado, sem que nenhum outro fator tenha influenciado esse resultado (Wohlin et al., 2012). Essa ameaça foi minimizada pelos modelos de diálogo e de comportamento não possuírem uma relação direta entre eles. As informações do modelo de diálogo podem ser utilizadas para o modelo de comportamento e vice-versa, porém, não existe uma obrigatoriedade entre essa relação. A

ameaça com relação a diferença do tamanho dos *sites* escolhidos como artefatos no 1º experimento foi mitigada nos experimentos seguintes, sendo escolhidos *sites* com tamanho mais próximos.

De acordo com Wohlin et al. (2012), as ameaças à validade interna afetam as variáveis independentes do experimento e são relacionadas ao desempenho dos participantes durante as tarefas. Para mitigar essa ameaça foram realizados treinamentos em todos os experimentos, com os tópicos relacionados a todo conteúdo do experimento e não apenas sobre a ModelUI$_{VIZ}$. Outra ameaça à validade interna está relacionada à quantidade de tempo necessário para conduzir o experimento. No 1º experimento foram usadas quase 12 horas para o treinamento e as atividades das sessões do experimento, desencorajando 55% das pessoas que participaram do primeiro dia do experimento. Por esse motivo, o treinamento e as sessões dos experimentos foram modificados para utilizarem menos de 6 horas. Adicionalmente, a aleatoriedade na formação dos grupos para os três experimentos colaborou para evitar o agrupamento tendencioso dos participantes.

As ameaças à validade de conclusão estão relacionadas às dificuldades para obter uma conclusão correta dos resultados do experimento (Wohlin et al., 2012). No 1º experimento, os *sites* selecionados para as atividades das sessões foram considerados uma ameaça devido à grande diferença de tamanho entre eles. Por este motivo, os sites utilizados nos outros experimentos foram alterados e eram mais proporcionais entre eles, como mostrado nas tabelas 5.10 e 5.14.

As ameaças à validade externa são relacionadas às condições, como tempo, lugar e pessoas que limitam a capacidade de generalizar os resultados do experimento (Wohlin et al., 2012). Essa ameaça é reduzida se o ambiente do experimento for o mais próximo possível do ambiente real. Porém, neste estudo como os participantes dos experimentos foram alunos ou ex-alunos de Ciência da Computação essa ameaça não pode ser minimizada a tempo. Em estudos futuros essa ameaça pode ser mitigada.

5.9 Considerações finais

Neste capítulo são apresentados os experimentos realizados para avaliar a compreensão das versões desenvolvidas da ferramenta WMUIT e da ModelUI$_{VIZ}$. Foram realizados 3 experimentos com o objetivo de comparar a abordagem manual e a abordagem visual, utilizando a ModelUI$_{VIZ}$.

O primeiro experimento, apresentado na Seção 5.3, foi realizado para avaliar a versão 1 da WMUIT e da ModelUI$_{VIZ}$ (Seção 5.1). Esse experimento foi considerado como piloto para análise da estrutura e dos artefatos utilizados. Com os resultados identificados nesse experimento (Seção 5.3.1), foram realizadas alterações na estrutura, nos artefatos utilizados e na ModelUI$_{VIZ}$.

O 2º experimento, apresentado na Seção 5.5, foi realizado para avaliar a versão 2 da WMUIT e da ModelUI$_{VIZ}$ (Seção 5.4), sendo obtidos resultados negativos para a identificação de componentes com a abordagem visual. Por conta desses resultados (Seção 5.5.1), foi desenvolvida a versão 3 da ModelUI$_{VIZ}$ (Seção 5.6) e realizado o 3º experimento. Neste último experimento os resultados obtidos foram positivos com relação a utilização da abordagem visual em comparação com a abordagem manual, apresentado na Seção 5.7.1.

CAPÍTULO 6

Conclusões

A utilização das representações tradicionais da literatura, como UML, para as informações da interface do usuário não colabora para processo de compreensão e manutenção da interface. As ferramentas identificadas na literatura, em sua maioria, utilizam essas representações como artefatos para testes, para o processo de documentação e de manutenção da interface. Porém, essas documentações formais são poucos utilizadas e atualizadas, principalmente com relação ao ambiente *web* (Theunissen e van Heesch, 2016). Além disso, não foram identificados estudos formais afirmando que essas representações tradicionais são as melhores para a representação das informações da interface.

Neste trabalho foi apresentada a ModelUI$_{VIZ}$, uma representação das informações da interface do usuário utilizando as técnicas de Visualização de Informação. Junto à essa representação foram desenvolvidas a ferramenta WMUIT, que apresenta a ModelUI$_{VIZ}$, e o *plug-in* WMUID, que extrai as informações de interfaces *web*. Foram realizados três experimentos controlados para a avaliação da ModelUI$_{VIZ}$ em comparação a abordagem manual (observação direto do código-fonte) para compreensão da interface. A cada experimento foram realizadas evoluções na representação ModelUI$_{VIZ}$ e na WMUIT. Com base na análise estatística dos resultados obtidos no último experimento é possível afirmar que a ModelUI$_{VIZ}$ colabora na compreensão da interface do usuário em comparação a observação direta do código-fonte da interface.

6.1 Contribuições e limitações

A ModelUI$_{VIZ}$ é a maior contribuição deste trabalho, que inclui 11 visões para apresentar diferentes informações com relação a interface do usuário. Principalmente as visões desenvolvidas para representar a estrutura das páginas. O *plug-in* WMUID é outra contribuição pela utilização de diferentes técnicas de extração de informações em uma mesma ferramenta. A ferramenta WMUIT é também uma contribuição

para implementação das técnicas de Visualização de Informação utilizando a biblioteca D3.js, que é bem conhecida na indústria de software.

Além disso, foi possível observar que a ModelUI$_{VIZ}$ permitiu a identificação de problemas relacionados à má utilização dos padrões *web* da W3C. Embora esse fato tenha sido observado durante na condução dos experimentos e, por isso, não possui comprovação estatística.

As avaliações realizadas foram limitadas à compreensão da ModelUI$_{VIZ}$ e, por isso, a avaliação da usabilidade da ferramenta WMUID e do desempenho do *plug-in* WMUID não constam no estudo. Futuramente, esses estudos e avaliações podem ser realizados para ambas.

Os resultados gerados nas avaliações também são considerados como uma limitação deste projeto. Os resultados foram obtidos a partir do uso único dos *sites* selecionados para os experimentos controlados. Assim, a proporção dos dados gerados para essas avaliações foram muito reduzidas, principalmente se comparados ao uso cotidiano de outros *sites* ou com uma grande quantidade de páginas. Por conta disso, a generalização do resultado para o uso industrial pode ser comprometido. Entretanto, como houve uma recepção positiva para a utilização da ModelUI$_{VIZ}$ entre os participantes, o mesmo pode ocorrer para o uso industrial da ModelUI$_{VIZ}$.

6.2 Trabalhos futuros

Considerando as possibilidades que o *plug-in* WMUID, a ferramenta WMUIT e a ModelUI$_{VIZ}$ podem alcançar, várias modificações e aprimoramentos podem ser realizados visando o melhor uso para usuário.

A criação da visão de evolução da interface foi identificada como uma grande contribuição futura para análise das modificações realizadas na interface no processo de manutenção, utilizando os arquivos gerados anteriormente pelo *plug-in* WMUID. Além dela, outras interações para aprimorar a interação e a navegação entre as visões também foram identificadas, como por exemplo, adicionar a seleção de múltiplos objetos.

O estudo da escalabilidade da ModelUI$_{VIZ}$ permitirá verificar o nível de compreensão do modelo visual de acordo com o tamanho do *site*, observando a necessidade da inclusão de outras técnicas complementares para melhorar a apresentação dos dados. As análises de performance do *plug-in* WMUID e de usabilidade da ferramenta WMUIT também permitirá melhorias no desempenho e utilização das ferramentas.

6.3 Produções bibliográficas

Adicionalmente, um artigo relacionados aos modelos de interface foi publicado, como segue:

- Martins, L. C. G.; Garcia, R. E. *Validation of User Interface Model: a Systematic Literature Review*. In: Proceedings of the International Conference on Software Engineering Research and Practice (SERP), Las Vegas, EUA, 2015,pp 145 - 152.

- Martins, L. C. G.; Garcia, R. E.; Marçal, I. *Using Information Visualization to comprehend user interface layer: na application to web-based systems*. No prelo.

Ressalta-se que o segundo artigo foi submetido para um evento e está em processo de avaliação.

Referências Bibliográficas

AHMED, S.; ASHRAF, G. Model-based user interface engineering with design patterns. *Journal of Systems and Software*, v. 80, n. 8, p. 1408–1422, 2007.

AHO, P.; MENZ, N.; RÄTY, T.; SCHIEFERDECKER, I. Automated java gui modeling for model-based testing purposes. In: *Proceedings of the Eighth International Conference on Information Technology: New Generations*, ITNG '11, Washington, DC, USA: IEEE Computer Society, 2011, p. 268–273 (*ITNG '11*, v.).

AHO, P.; SUAREZ, M.; KANSTREN, T.; MEMON, A. M. Murphy tools: Utilizing extracted GUI models for industrial software testing. In: *IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, Cleveland, OH, USA: IEEE Computer Society, 2014, p. 343–348.

AKPAN, I. J.; BROOKS, R. J. Experimental evaluation of user performance on two-dimensional and three-dimensional perspective displays in discrete-event simulation. *Decision Support Systems*, v. 64, n. C, p. 14–30, 2014.

ALALFI, M. H.; CORDY, J. R.; DEAN, T. R. Automated reverse engineering of UML sequence diagrams for dynamic web applications. In: *International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Washington, DC, USA: IEEE Computer Society, 2009, p. 287–294.

ALMENDROS-JIMÉNEZ, J. M.; IRIBARNE, L. An extension of uml for the modeling of wimp user interfaces. *Journal of Visual Languages & Computing*, v. 19, n. 6, p. 695–720, 2008.

ALMENDROS-JIMENEZ, J. M.; IRIBARNE, L. UML modeling of user and database interaction. *The Computer Journal*, v. 52, n. 3, p. 348–367, 2009.

AMALFITANO, D.; FASOLINO, A. R.; POLCARO, A.; TRAMONTANA, P. Comprehending ajax web applications by the DynaRIA tool. In: *International Conference Quality of Information and Communications Technology (QUATIC)*, New York, NY, USA: IEEE Computer Society, 2010a, p. 122–131.

AMALFITANO, D.; FASOLINO, A. R.; POLCARO, A.; TRAMONTANA, P. DynaRIA: A tool for ajax web application comprehension. In: *IEEE 18th International Conference on Program Comprehension*, New York, NY, USA: IEEE Computer Society, 2010b, p. 46–47.

AMALFITANO, D.; FASOLINO, A. R.; TRAMONTANA, P. Reverse engineering finite state machines from rich internet applications. In: *Proceedings of the 15th Working Conference on Reverse Engineering*, WCRE '08, Washington, DC, USA: IEEE Computer Society, 2008, p. 69–73 (*WCRE '08*, v.).

AMALFITANO, D.; FASOLINO, A. R.; TRAMONTANA, P. An iterative approach for the reverse engineering of rich internet application user interfaces. In: *Fifth International Conference on Internet and Web Applications and Services*, New York, NY, USA: IEEE Computer Society, 2010c, p. 401–410.

AMALFITANO, D.; FASOLINO, A. R.; TRAMONTANA, P. Techniques and tools for rich internet applications testing. In: *International Symposium on Web Systems Evolution (WSE)*, New York, NY, USA: IEEE Computer Society, 2010d, p. 63–72.

AMALFITANO, D.; FASOLINO, A. R.; TRAMONTANA, P. Using dynamic analysis for generating end user documentation for web 2.0 applications. In: *13th IEEE International Symposium on Web Systems Evolution (WSE)*, Williamsburg, VI, USA: IEEE Computer Society, 2011, p. 11–20.

ANDREWS, K. Evaluating information visualisations. In: *Proceedings of the AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, BELIV '06, New York, NY, USA: ACM, 2006, p. 1–5 (*BELIV '06*, v.).

ANTONIOL, G.; PENTA, M. D.; ZAZZARA, M. Understanding web applications through dynamic analysis. In: *Proceedings of the 12th IEEE International Workshop on Program Comprehension*, IWPC '04, Washington, DC, USA: IEEE Computer Society, 2004, p. 120–130 (*IWPC '04*, v.).

ARCAINI, P.; GARGANTINI, A.; VAVASSORI, P. Validation of models and tests for constrained combinatorial interaction testing. In: *IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, IEEE Computer Society, 2014, p. 98–107.

BALDONADO, M. Q. W.; WOODRUFF, A.; KUCHINSKY, A. Guidelines for using multiple views in information visualization. In: *Proceedings of the working conference on Advanced visual interfaces*, AVI '00, New York, NY, USA: ACM Press, 2000, p. 110–119 (*AVI '00*, v.).

BECK, F.; BURCH, M.; DIEHL, S.; WEISKOPF, D. The state of the art in visualizing dynamic graphs. *EuroVis STAR*, p. 83–103, 2014.

BECKER, R. A.; CLEVELAND, W. S.; WILKS, A. R. Dynamic graphics for data analysis. *Statistical Science*, v. 2, n. 4, p. 355–383, 1987.

BELLETTINI, C.; MARCHETTO, A.; TRENTINI, A. Validation of reverse engineered web application models. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, v. 1, n. 4, p. 1070–1073, 2007. Disponível em http://waset.org/Publications?p=4

BOSTOCK, M. Biblioteca d3.js. 2016.
Disponível em http://d3js.org/

BOUILLON, L.; VANDERDONCKT, J. Retargeting web pages to other computing platforms with VAQUITA. In: *Proceedings of the Ninth Working Conference on Reverse Engineering*, WCRE '02, Washington, DC, USA: IEEE Computer Society, 2002, p. 339–348 (*WCRE '02*, v.).

BURCH, M.; VEHLOW, C.; BECK, F.; DIEHL, S.; WEISKOPF, D. Parallel edge splatting for scalable dynamic graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, v. 17, n. 12, p. 2344–2353, 2011.

CARD, S. K.; MACKINLAY, J. D.; SHNEIDERMAN, B., eds. *Readings in information visualization: using vision to think*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

CASERTA, P.; ZENDRA, O. Visualization of the static aspects of software: a survey. *IEEE Transactions on Visualization and Computer Graphics*, v. 17, n. 7, p. 913–933, 2011.

CHEN, C.-H.; HÄRDLE, W. K.; UNWIN, A. *Handbook of data visualization*. Santa Clara, CA, USA: Springer Science & Business Media, 2007.

CHIAPPINI, A.; CIMATTI, A.; MACCHI, L.; REBOLLO, O.; ROVERI, M.; SUSI, A.; TONETTA, S.; VITTORINI, B. Formalization and validation of a subset of the european train control system. In: SOCIETY, I. C., ed. *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, ACM Press, 2010, p. 109–118 (*ICSE'10*, v.2).

CHIKOFSKY, E. J.; CROSS, J. H.; ET AL. Reverse engineering and design recovery: A taxonomy. *IEEE Software*, v. 7, n. 1, p. 13–17, 1990.

CHOUDHARY, S.; DINCTURK, M. E.; MIRTAHERI, S. M.; MOOSAVI, A.; VON BOCHMANN, G.; JOURDAN, G.-V.; ONUT, I. V. Crawling rich internet applications:

the state of the art. In: *Proceedings of the Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '12, Riverton, NJ, USA: IBM Corp., 2012, p. 146–160 (*CASCON '12*, v.).
Disponível em http://dl.acm.org/citation.cfm?id=2399776.2399790

CHUANG, K.; SHIH, C.; HUNG, S. User behavior augmented software testing for user-centered gui. In: *Proceedings of the ACM Symposium on Research in Applied Computation*, RACS '11, New York, NY, USA: ACM, 2011, p. 200–208 (*RACS '11*, v.).

CLOUTIER, J.; KPODJEDO, S.; EL BOUSSAIDI, G. Wavi: A reverse engineering tool for web applications. In: *IEEE 24th International Conference on Program Comprehension (ICPC)*, Austin, Texas, EUA.: IEEE Computer Society, 2016, p. 1–3.

CONVERTINO, G.; CHEN, J.; YOST, B.; RYU, Y. S.; NORTH, C. Exploring context switching and cognition in dual-view coordinated visualizations. In: *Proceedings International Conference on Coordinated and Multiple Views in Exploratory Visualization - CMV 2003* -, IEEE Computer Society, 2003, p. 55–62.

DELFIM, F. M. *Uma abordagem usando visualização de software como apoio à refatoração para aspectos*. Dissertação de mestrado, Universidade Estadual Paulista (UNESP), 2013.

DELFIM, F. M.; GARCIA, R. E. Uma ferramenta para mineração de aspectos. *Interciência & Sociedade*, v. 3, n. 1, p. 81–90, 2014.

DEMEYER, S.; DUCASSE, S.; LANZA, M. A hybrid reverse engineering approach combining metrics and program visualisation. In: *Sixth Working Conference on Reverse Engineering*, IEEE Comput. Soc, 1999, p. 175–186.

DIEHL, S. *Software visualization: Visualizing the structure, behaviour, and evolution of software*. Secaucus, NJ, USA: Springer-Verlag, 2007.

DINCTURK, M. E.; JOURDAN, G.-V.; BOCHMANN, G. V.; ONUT, I. V. A model-based approach for crawling rich internet applications. *ACM Transactions Web*, v. 8, n. 3, p. 1–39, 2014.

DOGRUSOZ, U.; FENG, Q.; MADDEN, B.; DOORLEY, M.; FRICK, A. Graph visualization toolkits. *IEEE Computer Graphics and Applications*, v. 22, n. 1, p. 30–37, 2002.

EGYED, A. Automatically detecting and tracking inconsistencies in software design models. *IEEE Transactions on Software Engineering*, v. 37, n. 2, p. 188–204, 2011.

ELKOUTBI, M.; KELLER, R. K. User interface prototyping based on UML scenarios and high-level petri nets. In: *Proceedings of the 21st International Conference on Application and Theory of Petri Nets*, ICATPN'00, Berlin, Heidelberg: Springer-Verlag, p. 166–186, 2000.
Disponível em http://dl.acm.org/citation.cfm?id=1754589.1754602

ELLIS, G.; DIX, A. An explorative analysis of user evaluation studies in information visualisation. In: *Proceedings of the AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, BELIV '06, New York, NY, USA: ACM Press, 2006, p. 1–7 (*BELIV '06*, v.).

ELMQVIST, N.; FEKETE, J.-D. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics*, v. 16, n. 3, p. 439–454, 2010.

GALLAGHER, K.; HATCH, A.; MUNRO, M. Software architecture visualization: An evaluation framework and its application. *IEEE Transactions on Software Engineering*, v. 34, n. 2, p. 260–270, 2008.

GRILO, A. M.; PAIVA, A. C.; FARIA, J. P. Reverse engineering of GUI models for testing. In: *Iberian Conference on Information Systems and Technologies*, IEEE Computer Society, 2010, p. 1–6.

GUGLIELMO, G. D.; FUJITA, M.; GUGLIELMO, L. D.; FUMMI, F.; PRAVADELLI, G.; MARCONCINI, C.; FOLTINEK, A. Model-driven design and validation of embedded software. In: *Proceeding of the 6th international workshop on Automation of software test*, AST '11, New York, NY, USA: ACM, 2011, p. 98–104 (*AST '11*, v.).

HEINRICH, J.; WEISKOPF, D. State of the art of parallel coordinates. In: *Proceedings of Eurographics*, Eurographics Association, 2013, p. 95–116.

HOLTEN, D. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, v. 12, n. 5, p. 741–748, 2006.

HORROCKS, I. *Constructing the user interface with statecharts*. 1 ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

HURTADO, N.; RUIZ, M.; ORTA, E.; TORRES, J. Using simulation to aid decision making in managing the usability evaluation process. *Information and Software Technology*, v. 57, p. 509–526, 2015.

JESCHKE, S.; VIERITZ, H.; PFEIFFER, O. Developing accessible applications with user-centered architecture. In: *Seventh IEEE/ACIS International Confe-*

rence on Computer and Information Science (icis 2008), IEEE Computer Society, 2008, p. 684–689.

JOHNSON, B.; SHNEIDERMAN, B. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In: *Proceedings of the 2Nd Conference on Visualization '91*, VIS '91, Los Alamitos, CA, USA: IEEE Computer Society Press, 1991, p. 284–291 (*VIS '91*, v.).
Disponível em http://dl.acm.org/citation.cfm?id=949607.949654

KAZMAN, R.; O'BRIEN, L.; VERHOEF, C. *Architecture reconstruction guidelines*. Relatório técnico, DTIC Document, 2002.

KEIM, D. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, v. 8, n. 1, p. 1–8, 2002.

KEIM, D. A.; KRIEGEL, H. P. Visualization techniques for mining large databases: a comparison. *IEEE Transactions on Knowledge and Data Engineering*, v. 8, n. 6, p. 923–938, 1996.

KEIM, D. A.; PANSE, C.; SIPS, M. Information Visualization : Scope, Techniques and Opportunities for Geovisualization. In: DYKES, J., ed. *Exploring Geovisualization*, Oxford: Elsevier, p. 1–17, 2004.

KHAN, M.; KHAN, S. S. Data and information visualization methods, and interactive mechanisms: A survey. *International Journal of Computer Applications*, v. 34, n. 1, p. 1–14, 2011.
Disponível em http://www.ijcaonline.org/archives/volume34/number1/4061-5722

KIENLE, H. M.; MULLER, H. A. Requirements of software visualization tools: A literature survey. In: *4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, IEEE Computer Society, 2007, p. 2–9.

KNIGHT, C.; MUNRO, M. Visualising software: a key research area. In: *Proceedings of the IEEE International Conference on Software Maintenance*, Citeseer, 1999, p. 437.

KOF, L.; GACITUA, R.; ROUNCEFIELD, M.; SAWYER, P. Ontology and model alignment as a means for requirements validation. In: *IEEE Fourth International Conference on Semantic Computing*, IEEE Computer Society, 2010, p. 46–51.

LALIOTI, V.; LOUCOPOULOS, P. Visualisation for validation. In: *Advanced Information Systems Engineering*, Springer-Verlag, 1993, p. 143–164.

LAM, H.; BERTINI, E.; ISENBERG, P.; PLAISANT, C.; CARPENDALE, S. Empirical studies in information visualization: Seven scenarios. *IEEE Transactions on Visualization and Computer Graphics*, v. 18, n. 9, p. 1520–1536, 2012.

LÄMMEL, R.; VISSER, E.; VISSER, J. The essence of strategic programming, Draft, 2002.
Disponível em http://www.cwi.nl/~ralf/

LÄMMEL, R.; VISSER, J. Design patterns for functional strategic programming. In: *Proceedings of the ACM SIGPLAN workshop on Rule-based programming*, RULE '02, New York, NY, USA: ACM Press, 2002, p. 1–14 (*RULE '02*, v.).

LANGE, C.; CHAUDRON, M.; MUSKENS, J. In practice: UML software architecture and design description. *IEEE Software*, v. 23, n. 2, p. 40–46, 2006.

LEOPOLD, H.; MENDLING, J.; POLYVYANYY, A. Supporting process model validation through natural language generation. *IEEE Transactions on Software Engineering*, v. 40, n. 8, p. 818–840, 2014.

LEUNG, K. R. P. H.; HUI, L. C. K.; YIU, S. M.; TANG, R. W. M. Modeling web navigation by statechart. In: *24th International Computer Software and Applications Conference*, COMPSAC '00, Washington, DC, USA: IEEE Computer Society, 2000, p. 41–47 (*COMPSAC '00*, v.).
Disponível em http://dl.acm.org/citation.cfm?id=645982.675079

LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions, and reversals. In: *Soviet physics doklady*, 1966, p. 707–710.

LUCCA, G. A. D.; FASOLINO, A. R.; TRAMONTANA, P. Reverse engineering web applications: the WARE approach. *Journal of Software Maintenance and Evolution: Research and practice*, v. 16, n. 12, p. 71–101, 2004.

MARAS, J.; ŠTULA, M.; CRNKOVIC, I. phpModeler - a web model extractor. In: *IEEE/ACM International Conference on Automated Software Engineering*, IEEE Computer Society, 2009, p. 660–661.

MARCHETTO, A.; TONELLA, P.; RICCA, F. ReAjax: a reverse engineering tool for Ajax Web applications. *IET Software*, v. 6, p. 33–49, 2012.

MARKOPOULOS, P.; MARIJNISSEN, P. UML as a representation for Interaction Designs. In: *Proceedings Australian Conference Computer-Human Interaction*, 2000, p. 240–249.

MARTINS, L. C. G.; GARCIA, R. E. Validation of User Interface Model: a Systematic Literature Review. In: *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, The Steering Committee

of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015, p. 145–152.

MATTHIJSSEN, N.; ZAIDMAN, A.; STOREY, M.-A.; BULL, I.; VAN DEURSEN, A. Connecting traces: Understanding client-server interactions in ajax applications. In: *IEEE 18th International Conference on Program Comprehension*, Braga, Portugal, Portugal: IEEE Computer Society, 2010, p. 216–225.

MAZZA, R. *Introduction to Information Visualization*. Springer Science & Business Media, 2009.

MEMON, A.; BANERJEE, I.; NAGARAJAN, A. GUI ripping: reverse engineering of graphical user interfaces for testing. In: *Proceedings of the 10th Working Conference on Reverse Engineering*, WCRE '03, Washington, DC, USA: IEEE Computer Society, 2003, p. 260–269 (*WCRE '03*, v.).

MEMON, A. M. An event-flow model of GUI-based applications for testing. *Software testing, verification and reliability*, v. 17, n. 3, p. 137–157, 2007.

MESBAH, A.; VAN DEURSEN, A.; LENSELINK, S. Crawling Ajax-based web applications through dynamic analysis of user interface state changes. *ACM Transactions on the Web*, v. 6, n. 1, p. 1–30, 2012.

MICHAIL, A. Browsing and Searching Source Code of Applications Written Using a GUI Framework. In: *Proceedings of the 24th international conference on Software engineering*, ICSE'02, New York, NY, USA: ACM, 2002, p. 327–337 (*ICSE'02*, v.).

MICROSOFT UI Automation Overview. 26/01/2016, 2016.
Disponível em https://msdn.microsoft.com/en-us/library/ms747327(v=vs.110).aspx

MORGADO, I. C.; PAIVA, A. C.; FARIA, J. P. Dynamic reverse engineering of graphical user interfaces. *International Journal On Advances in Software*, v. 5, n. 3 - 4, p. 224–236, 2012.

MOTTI, V. G.; RAGGETT, D.; CAUWELAERT, S. V.; VANDERDONCKT, J. Simplifying the development of cross-platform web user interfaces by collaborative model-based design. In: *Proceedings of the 31st ACM international conference on Design of communication*, SIGDOC'13, ACM, 2013, p. 55–64 (*SIGDOC'13*, v.).

MULTICOBRA Multicobra cobranÃ§as. 2017.
Disponível em www.multicobra.com.br

MUNZNER, T. A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, v. 15, n. 6, p. 921–928, 2009.

NORTH, C.; SHNEIDERMAN, B. Snap-together visualization: can users construct and operate coordinated visualizations? *International Journal of Human-Computer Studies*, v. 53, n. 5, p. 715–739, 2000.

NUNES, N. J. Representing user-interface patterns in UML. In: *Object-Oriented Information Systems*, Geneva, Switzerland: Springer-Verlag, p. 142–151, 2003.

OBJECT MANAGEMENT GROUP Object Management Group - OMG. 2014. Disponível em http://www.omg.org/

PAGANELLI, L.; PATERNÒ, F. Tools for remote usability evaluation of web applications through browser logs and task models. *Behavior Research Methods, Instruments, & Computers*, v. 35, n. 3, p. 369–378, 2003.

PATEL, R.; COENEN, F.; MARTIN, R.; ARCHER, L.; NX, W. C. *Reverse Engineering of Web Applications: A Technical Report*. Relatório técnico, Computer Science Department, University of Liverpool, 2007.

PATERNÒ, F. ConcurTaskTrees: an engineered approach to model-based design of interactive systems. *The Handbook of Analysis for Human-Computer Interaction, Lawrence Erlbaum Associates*, p. 483–500, 2002.

PATERNÒ, F.; MANCINI, C.; MENICONI, S. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In: *Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, INTERACT '97, Boston, MA: Chapman & Hall, 1997, p. 362–369 (*INTERACT '97*, v.).

PETRE, M. UML in practice. In: *35th International Conference on Software Engineering (ICSE)*, ICSE '13, Piscataway, NJ, USA: IEEE Press, 2013, p. 722–731 (*ICSE '13*, v.).

PLEUSS, A.; WOLLNY, S.; BOTTERWECK, G. Model-driven development and evolution of customized user interfaces. In: *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '13, New York, NY, USA: ACM Press, 2013, p. 13–22 (*EICS '13*, v.).

POSTO KAÓ Posto kaó. 2016.
Disponível em http://www.postokao.com.br/

PUERTA, A. R. The MECANO Project: Comprehensive and Integrated Support for Model-Based Interface Development. In: *CADUI*, Citeseer, 1996, p. 19–36.

PUERTA, A. R. A model-based interface development environment. *IEEE Software*, v. 14, n. 4, p. 40–47, 1997.

RANEBURGER, D.; POPP, R.; VANDERDONCKT, J. An automated layout approach for model-driven WIMP-UI generation. In: *Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '12, New York, NY, USA: ACM Press, 2012, p. 91–100 (*EICS '12*, v.).

ROBERTS, J. C. Visualization display models-ways to classify visual representations. *International Journal of Computer Integrated Design and Construction*, v. 2, n. 4, p. 241–250, 2000.

ROBERTS, J. C. Chapter 8 - exploratory visualization with multiple linked views. In: DYKES, J.; MACEACHREN, A. M.; KRAAK, M.-J., eds. *Exploring Geovisualization*, International Cartographic Association, Oxford: Elsevier, p. 159–180, 2005.

ROBERTS, J. C. State of the art: Coordinated & multiple views in exploratory visualization. In: *Proceedings of the Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization*, CMV '07, Washington, DC, USA: IEEE Computer Society, 2007, p. 61–71 (*CMV '07*, v.).

ROBERTSON, G. G.; MACKINLAY, J. D.; CARD, S. K. Cone Trees: Animated 3D Visualizations of Hierarchical Information. In: *Proceedings of the SIGCHI conference on Human factors in computing systems Reaching through technology*, CHI '91, New York, NY, USA: ACM Press, 1991, p. 189–194 (*CHI '91*, v.).

ROOKE, M.; GROSSMAN, T.; FITZMAURICE, G. AppMap: Exploring User Interface Visualizations. In: *Proceedings of Graphics Interface*, GI '11, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada: Canadian Human-Computer Communications Society, 2011, p. 111–118 (*GI '11*, v.). Disponível em http://dl.acm.org/citation.cfm?id=1992917.1992936

SALIHU, I. A.; IBRAHIM, R. *Comparative analysis of gui reverse engineering techniques*, cáp. 24 Cham: Springer International Publishing, p. 295–305, 2016.

SANTOS, B. S.; ZAMFIR, F.; FERREIRA, C.; MEALHA, O.; NUNES, J. Visual application for the analysis of web-based information systems usage: a preliminary usability evaluation. In: *Proceedings. Eighth International Conference on Information Visualisation*, IV '04, Washington, DC, USA: IEEE Computer Society, 2004, p. 812–818 (*IV '04*, v.).

SASIREKHA, N.; ROBERT, A. E.; HEMALATHA, M. Program slicing techniques and its applications. *International Journal of Software Engineering & Applications*, v. 2, n. 3, p. 50–64, 2011.

SCHERR, M. *Multiple and coordinated views in information visualization*. tech-report LMU-MI-2010-1, Department of Computer Science, University of Munich, 2010.

SCHLUNGBAUM, E. *Model-based user interface software tools current state of declarative models*. Relatório Técnico GIT-GVU-96-30, Graphics, Visualization & Usability Center, Georgia Institute of Technology, Atlanta, GA, 1996.

SHNEIDERMAN, B. The eyes have it: A task by data type taxonomy for information visualizations. In: *Proceedings of the IEEE Symposium on Visual Languages*, VL '96, Washington, DC, USA: IEEE Computer Society, 1996, p. 336–343 (*VL '96*, v.).

SILVA, C. E.; CAMPOS, J. C. Combining static and dynamic analysis for the reverse engineering of web applications. In: *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '13, New York, NY, USA: ACM Press, 2013, p. 107–112 (*EICS '13*, v.).

SILVA, J. C. *GUISURFER: A Generic Framework for Reverse Engineering of Graphical User Interfaces*. Tese de doutorado, Universidade do Minho, 2010.

SILVA, J. C.; SILVA, C. C.; GONÇALO, R. D.; SARAIVA, J.; CAMPOS, J. C. The GUISurfer Tool: Towards a Language Independent Approach to Reverse Engineering GUI Code. In: *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems*, EICS '10, New York, NY, USA: ACM Press, 2010, p. 181–186 (*EICS '10*, v.).

SILVA, P. P. D.; PATON, N. W. Improving UML Support for User Interface Design: A Metric Assessment of UMLi. In: *Proceedings of Workshop on Bridging the Gaps Between Software Engineering and Human-Computer Interaction (ICSE)*, 2003a, p. 76–83.

SILVA, P. P. D.; PATON, N. W. User interface modeling in UMLi. *IEEE Software*, v. 20, n. 4, p. 62–69, 2003b.

SILVA, T. S. D.; SILVEIRA, M. S. Validação de um método para identificação de problemas de usabilidade a partir de diagramas uml. In: *Proceedings of the Symposium on Human Factors in Computing Systems*, Porto Alegre, Brazil: Brazilian Computer Society, 2010, p. 179–188.

SYSTÄ, T.; TAMPERENSIS, U. *Static and dynamic reverse engineering techniques for Java software systems*. Tese de doutorado, University of Tampere, report A-2000-4, 2000.

THEUNISSEN, T.; VAN HEESCH, U. *The disappearance of technical specifications in web and mobile applications* Cham: Springer International Publishing, p. 265–273, 2016.

W3C, W. Scalable vector graphics (svg). 2016a.
Disponível em https://www.w3.org/Graphics/SVG/

W3C, W. UI Events. 04/04/2016, 2016b.
Disponível em https://www.w3.org/TR/uievents/

WARD, M. O.; GRINSTEIN, G.; KEIM, D. *Interactive data visualization: foundations, techniques, and applications*. CRC Press, 2010.

WINCKLER, M.; PALANQUE, P. *Statewebcharts: A formal description technique dedicated to navigation modelling of web applications*, cáp. 5 Springer Berlin, Heidelberg: Springer-Verlag, p. 61–76, 2003.

WINCKLER, M. A.; PALANQUE, P.; FREITAS, C. M. D. S. Tasks and scenario-based evaluation of information visualization techniques. In: *Proceedings of the 3rd Annual Conference on Task Models and Diagrams*, TAMODIA '04, New York, NY, USA: ACM Press, 2004, p. 165–172 (*TAMODIA '04*, v.).

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. *Experimentation in software engineering*. Berlin, Germany: Springer Science & Business Media, 2012.

WOLFF, A.; FORBRIG, P. Deriving user interfaces from task models. In: *Proceedings of International Conference on Intelligent User Interfaces*, 2009 (*IUI'09*, v.9).

YI, J. S.; A. KANG, Y.; STASKO, J. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, v. 13, n. 6, p. 1224–1231, 2007.

ZAIDMAN, A.; MATTHIJSSEN, N.; STOREY, M.-A.; VAN DEURSEN, A. Understanding Ajax applications by connecting client and server-side execution traces. *Empirical Software Engineering*, v. 18, n. 2, p. 181–218, 2013.

APÊNDICE

A

Apêndice A

A.1 Formulários dos experimentos

A.1.1 Formulário de acompanhamento

1. E-mail do participante:
2. Horário de início da criação do modelo de diálogo
3. Horário de término da criação do modelo de diálogo
4. Horário de início da criação do modelo de comportamento
5. Horário de término da criação do modelo de comportamento
6. Grupo que pertence:
   (a) Grupo 1
   (b) Grupo 2
7. Quais páginas do site foram documentadas completamente do modelo de diálogo criado?
8. Quais páginas do site foram documentadas parcialmente do modelo de diálogo criado?
9. Quais páginas do site foram documentadas completamente do modelo de comportamento criado?
10. Quais páginas do site foram documentadas parcialmente do modelo de comportamento criado?

A.1.2 Formulário final

1. E-mail do participante:
   (a) Grupo 1
   (b) Grupo 2
2. Avalie como foi o nível de compreensão da ModelUI$_{VIZ}$ durante o treinamento?
   (a) Nenhuma compreensão
   (b) Pouca compreensão
   (c) Compreensão considerável
   (d) Compreensão total
3. O treinamento foi suficiente para desenvolver as atividades propostas?
   (a) Sim
   (b) Não
4. Como você classificaria o treinamento:
   (a) Ruim
   (b) Regular
   (c) Bom
   (d) Ótimo
   (e) Excelente
5. Na sua opinião, o que poderia melhorar no treinamento realizado?
6. Das ferramentas utilizadas no experimento como apoio para criação dos diagramas, qual você utilizaria novamente no futuro?
   (a) Utilizaria a ferramenta de desenvolvedor do navegador Chrome como apoio
   (b) Utilizaria a ModelUI$_{VIZ}$ como apoio
7. Avalie como foi o nível de colaboração provida pela ModelUI$_{VIZ}$ para a criação dos modelos de interface
   (a) Nenhuma colaboração
   (b) Pouca colaboração
   (c) Colaboração considerável
   (d) Colaboração total

8. Na sua opinião, quais são os pontos positivos na utilização da ModelUI$_{VIZ}$ para a criação dos modelos de interface?

9. Na sua opinião, quais são os pontos negativos na utilização da ModelUI$_{VIZ}$ para a criação dos modelos de interface?

10. Quais os seus comentários para melhorar a ferramenta WebModelUI Tool ou a ModelUI$_{VIZ}$?

A.2 Experimento Piloto

A.2.1 *Site* A

Figura A.1: Página inicial do *site* Posto Kaó utilizado no experimento piloto.

Listagem A.1: Estrutura do arquivo XML criado pelo *crawler* do *site* Posto Kaó

```
<?xml version="1.0" encoding="UTF-8"?>
<site url="http://postokao.com.br/" titulo="Posto Kaó" tipo="crawler">
  <pages>
    <page url="http://postokao.com.br/" titulo="Posto Kaó" node_id="1" index="true">
      <event name="onLoad" node_id="1" item_id="null" event_id="1"/>
      <state name="onLoad" node_id="1" item_id="null" state_id="1"/>
      <state name="Load" node_id="1" item_id="null" state_id="2"/>
      <component type="button" dom_id="1" node_id="1" item_id="1" name="">
        <event name="click" node_id="1" item_id="1" event_id="2"></event>
        <event name="enter" node_id="1" item_id="1" event_id="3"></event>
        <state name="selected" node_id="1" item_id="1" state_id="3"></state>
        <state name="notselected" node_id="1" item_id="1" state_id="4"></state>
      </component>
    ...
    </page>

  </pages>

  <edges>
    <edge ed_id="1" source="1" target="1" ref_item_id="2">
      <data event_id="4">click</data>
      <data event_id="5">enter</data>
    </edge>
    <edge ed_id="2" source="1" target="1" ref_item_id="5">
      <data event_id="10">click</data>
      <data event_id="11">enter</data>
    </edge>
    ...
  </edges>

  <structure>
    <page url="http://postokao.com.br/" titulo="Posto Kaó" node_id="1" index="true">
      <node name="div" class="menu-mobile" id="" node_id="1" type="" item_id="" str_id="1">
        <node name="component" id="" class="btn-menu-mobile" type="button" dom_id="1" node_id="1"
        item_id="1"  externo="undefined"  str_id="2"></node>
        <node name="component" id="" class="logo" type="link" dom_id="4" node_id="1" item_id="2"
```

```
        externo="false"  str_id="3"></node>
      </node>
        <node name="section" class="site-top-section" id="" node_id="1" type="" item_id="" str_id="4">
        <node name="component" id="" class="btn-toggle-tab ico-svg_btn-tab-toggle" type="button"
        dom_id="7" node_id="1" item_id="3"  externo="undefined"  str_id="5"></node>
      </node>
    ...
    </page>
  </structure>

</site>
```

Listagem A.2: Estrutura do arquivo XML criado pelo *tracer* do *site* Posto Kaó

```
<?xml version="1.0" encoding="UTF-8"?>
<site url="http://postokao.com.br/" titulo="Posto Kaó" tipo="tracer">
  <pages>
    <page url="http://postokao.com.br/" titulo="Posto Kaó" node_id="1" index="true" novo="false">
      <event name="onLoad" node_id="1" item_id="null" event_id="1" novo="false"/>
      <state name="onLoad" node_id="1" item_id="null" state_id="1" novo="false"/>
      <state name="Load" node_id="1" item_id="null" state_id="2" novo="false"/>
  ...
  </page>

  </pages>
  <interactions>
    <interaction id_int="1" initialState="13" finalState="14" event_source="" event_target="12"
    source_id="6" type_source="component" target_id="2" type_target="page">8985</interaction>
    <interaction id_int="2" initialState="37" finalState="38" event_source="12" event_target="44"
    source_id="2" type_source="page" target_id="2" type_target="page">9035</interaction>
    <interaction id_int="3" initialState="89" finalState="90" event_source="" event_target="83"
    source_id="43" type_source="component" target_id="2" type_target="page">13637</interaction>
    <interaction id_int="4" initialState="37" finalState="38" event_source="83" event_target="44"
    source_id="2" type_source="page" target_id="2" type_target="page">13687</interaction>
  ...
  </interactions>
</site>
```

A.2.2 *Site* B

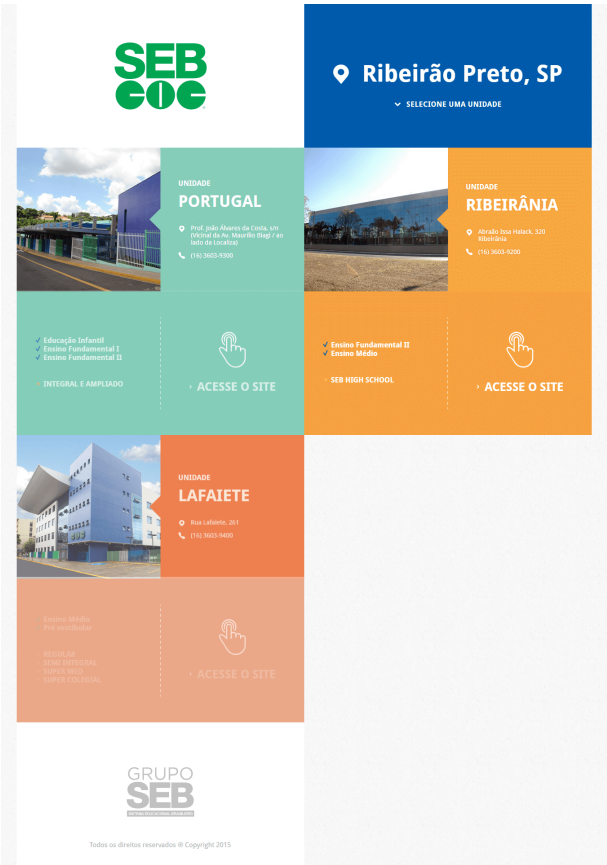

Figura A.2: Página inicial do *site* SEB-COC utilizado no experimento piloto.

Listagem A.3: Estrutura do arquivo XML criado pelo *crawler* do *site* SEB-COC

```
<?xml version="1.0" encoding="UTF-8"?>
<site url="http://reppublica.com.br/sebcoc/escolas/ribeirao/site/" titulo="SEB COC Ribeirão
Preto"
tipo="crawler">
  <pages>
    <page url="http://reppublica.com.br/sebcoc/escolas/ribeirao/site/"
    titulo="SEB COC Ribeirão Preto" node_id="1" index="true">
        <event name="onLoad" node_id="1" item_id="null" event_id="1"/>
        <state name="onLoad" node_id="1" item_id="null" state_id="1"/>
        <state name="Load" node_id="1" item_id="null" state_id="2"/>
      <component type="link" dom_id="8" node_id="1" item_id="1" name="Selecione uma unidade"
      externo="false">
        <event name="click" node_id="1" item_id="1" event_id="2"><![CDATA[#unidades]]></event>
        <event name="enter" node_id="1" item_id="1" event_id="3"><![CDATA[#unidades]]></event>
        <state name="not visited" node_id="1" item_id="1" state_id="3"></state>
        <state name="visited" node_id="1" item_id="1" state_id="4"></state>
      </component>
    ...
      <component type="button" dom_id="21" node_id="14" item_id="413" name="">
        <event name="click" node_id="14" item_id="413" event_id="813"></event>
        <event name="enter" node_id="14" item_id="413" event_id="814"></event>
        <state name="selected" node_id="14" item_id="413" state_id="924"></state>
        <state name="notselected" node_id="14" item_id="413" state_id="925"></state>
      </component>
```

```
    </page>
  </pages>

  <edges>
    <edge ed_id="1" source="1" target="1" ref_item_id="1">
      <data event_id="2">click</data>
      <data event_id="3">enter</data>
    </edge>
    <edge ed_id="2" source="1" target="2" ref_item_id="2">
      <data event_id="4">click</data>
      <data event_id="5">enter</data>
    </edge>
  ...
    <edge ed_id="199" source="9" target="10" ref_item_id="378">
      <data event_id="691">click</data>
      <data event_id="692">enter</data>
    </edge>
  </edges>
  <structure>
    <page url="http://reppublica.com.br/sebcoc/escolas/ribeirao/site/" titulo="SEB␣COC␣Ribeirão
␣␣␣␣Preto"   node_id="1" index="true">
      <node name="form" class="container-asp" id="landingPage" node_id="1" type="" item_id=""
      str_id="1">
        <node name="div" class="secao␣clearfix" id="" node_id="1" type="" item_id="" str_id="2">
          <node name="div" class="bloco-50␣altura-25␣bloco-100-360␣altura-50-360␣bkg-azul-1␣left
␣␣␣␣␣␣␣␣␣␣␣no-float-360" id="" node_id="1" type="" item_id="" str_id="3">
            <node name="div" class="absolute␣full-w␣middle␣txt-center␣select-cidade" id=""
            node_id="1" type="" item_id="" str_id="4">
              <node name="component" id="" class="upper␣bold␣txt-branco␣link-scroll
␣␣␣␣␣␣␣␣␣␣␣␣␣␣link-unidades" type="link" dom_id="8" node_id="1" item_id="1" externo="false"
                str_id="5"></node>
            </node>
          </node>
        </node>
  ...
      </node>
    </page>
  </structure>
</site>
```

Listagem A.4: Estrutura do arquivo XML criado pelo *tracer* do *site* SEB-COC

```
?xml version="1.0" encoding="UTF-8"?>
<site url="http://reppublica.com.br/sebcoc/escolas/ribeirao/site/" titulo="SEB␣COC␣Ribeirão␣Preto"
tipo="tracer">
  <pages>
    <page url="http://reppublica.com.br/sebcoc/escolas/ribeirao/site/" titulo="SEB␣COC␣Ribeirão␣Preto"
     node_id="1" index="true" novo="false">
      <event name="onLoad" node_id="1" item_id="null" event_id="1" novo="false"/>
      <state name="onLoad" node_id="1" item_id="null" state_id="1" novo="false"/>
      <state name="Load" node_id="1" item_id="null" state_id="2" novo="false"/>
      <component type="link" dom_id="8" node_id="1" item_id="1" name="Selecione uma unidade"
      externo="false" novo="false">
        <event name="click" node_id="1" item_id="1" event_id="2" novo="false"><![CDATA[#unidades]]>
        </event>
        <event name="enter" node_id="1" item_id="1" event_id="3" novo="false"><![CDATA[#unidades]]>
        </event>
        <state name="not␣visited" node_id="1" item_id="1" state_id="3" novo="false"/>
        <state name="visited" node_id="1" item_id="1" state_id="4" novo="false"/>
      </component>
  ...
      <component type="button" dom_id="21" node_id="14" item_id="413" name="" novo="false">
```

```
        <event name="click" node_id="14" item_id="413" event_id="813" novo="false"/>
        <event name="enter" node_id="14" item_id="413" event_id="814" novo="false"/>
        <state name="selected" node_id="14" item_id="413" state_id="924" novo="false"/>
        <state name="notselected" node_id="14" item_id="413" state_id="925" novo="false"/>
      </component>
    </page>
  </pages>
  <interactions>
    <interaction id_int="1" initialState="7" finalState="8" event_source="" event_target="6"
    source_id="3" type_source="component" target_id="3" type_target="page">28785</interaction>
    <interaction id_int="2" initialState="175" finalState="176" event_source="6" event_target="158"
    source_id="3" type_source="page" target_id="3" type_target="page">28835</interaction>
    <interaction id_int="3" initialState="229" finalState="230" event_source="" event_target="213"
    source_id="112" type_source="component" target_id="3" type_target="page">9945</interaction>
    <interaction id_int="4" initialState="309" finalState="310" event_source="" event_target="218"
    source_id="152" type_source="component" target_id="7" type_target="page">9945</interaction>
  ...
  </interactions>
</site>
```

A.3 2ª experimento

A.3.1 *Site* A

Figura A.3: Página inicial do *site* Multicobra utilizado no 2º experimento.

Listagem A.5: Estrutura do arquivo XML criado pelo *crawler* do *site* Multicobra

```
<?xml version="1.0" encoding="UTF-8"?>
<site url="http://www.multicobra.com.br/" titulo="Grupo␣Multicobra" tipo="crawler">
  <pages>
    <page url="http://www.multicobra.com.br/" titulo="Grupo␣Multicobra" node_id="1" index="true">
      <event name="onLoad" node_id="1" item_id="null" event_id="1"/>
      <state name="onLoad" node_id="1" item_id="null" state_id="1"/>
      <state name="Load" node_id="1" item_id="null" state_id="2"/>
      <component type="link" dom_id="13" node_id="1" item_id="1" name="Ouvidoria" externo="false">
        <event name="click" node_id="1" item_id="1" event_id="2"><![CDATA[]]></event>
        <event name="enter" node_id="1" item_id="1" event_id="3"><![CDATA[]]></event>
        <state name="not␣visited" node_id="1" item_id="1" state_id="3"></state>
        <state name="visited" node_id="1" item_id="1" state_id="4"></state>
      </component>
      <component type="link" dom_id="14" node_id="1" item_id="2" name="Fale␣Conosco" externo="false">
        <event name="click" node_id="1" item_id="2" event_id="4">
        <![CDATA[http://www.multicobra.com.br/fale-conosco/]]></event>
        <event name="enter" node_id="1" item_id="2" event_id="5">
        <![CDATA[http://www.multicobra.com.br/fale-conosco/]]></event>
        <state name="not␣visited" node_id="1" item_id="2" state_id="5"></state>
        <state name="visited" node_id="1" item_id="2" state_id="6"></state>
      </component>
  ...
      <component type="button" dom_id="388" node_id="10" item_id="655" name="">
        <event name="click" node_id="10" item_id="655" event_id="1602"></event>
        <event name="enter" node_id="10" item_id="655" event_id="1603"></event>
        <state name="selected" node_id="10" item_id="655" state_id="1839"></state>
        <state name="notselected" node_id="10" item_id="655" state_id="1840"></state>
      </component>
    </page>
  </pages>
  <edges>
    <edge ed_id="1" source="1" target="1" ref_item_id="1">
      <data event_id="2">click</data>
      <data event_id="3">enter</data>
    </edge>
    <edge ed_id="2" source="1" target="2" ref_item_id="2">
```

```
      <data event_id="4">click</data>
      <data event_id="5">enter</data>
    </edge>
  ...
  </edges>
  <structure>
    <page url="http://www.multicobra.com.br/" titulo="Grupo␣Multicobra" node_id="1" index="true">
      <node name="div" class="off-canvas-wrap" id="" node_id="1" type="" item_id="" str_id="1">
        <node name="div" class="inner-wrap" id="" node_id="1" type="" item_id="" str_id="2">
          <node name="header" class="main-top␣homepage-top␣show-for-large-up" id="" node_id="1"
           type="" item_id="" str_id="3">
            <node name="div" class="top-contact-bar" id="" node_id="1" type="" item_id=""
            str_id="4">
              <node name="div" class="row␣collapse" id="" node_id="1" type="" item_id=""
              str_id="5">
      ...
        </node>
      </node>
    </page>
  </structure>
</site>
```

Listagem A.6: Estrutura do arquivo XML criado pelo *tracer* do *site* Multicobra

```
<?xml version="1.0" encoding="UTF-8"?>
<site url="http://www.multicobra.com.br/" titulo="Grupo␣Multicobra" tipo="tracer">
  <pages>
    <page url="http://www.multicobra.com.br/" titulo="Grupo␣Multicobra" node_id="1" index="true"
    novo="false">
      <event name="onLoad" node_id="1" item_id="null" event_id="1" novo="false"/>
      <state name="onLoad" node_id="1" item_id="null" state_id="1" novo="false"/>
      <state name="Load" node_id="1" item_id="null" state_id="2" novo="false"/>
      <component type="link" dom_id="13" node_id="1" item_id="1" name="Ouvidoria" externo="false"
      novo="false">
        <event name="click" node_id="1" item_id="1" event_id="2" novo="false"><![CDATA[]]></event>
        <event name="enter" node_id="1" item_id="1" event_id="3" novo="false"><![CDATA[]]></event>
        <state name="not␣visited" node_id="1" item_id="1" state_id="3" novo="false"/>
        <state name="visited" node_id="1" item_id="1" state_id="4" novo="false"/>
      </component>
  ...
      <component type="button" dom_id="388" node_id="10" item_id="655" name="" novo="false">
        <event name="click" node_id="10" item_id="655" event_id="1602" novo="false"/>
        <event name="enter" node_id="10" item_id="655" event_id="1603" novo="false"/>
        <state name="selected" node_id="10" item_id="655" state_id="1839" novo="false"/>
        <state name="notselected" node_id="10" item_id="655" state_id="1840" novo="false"/>
      </component>
    </page>
  </pages>
  <interactions>
    <interaction id_int="1" initialState="15" finalState="16" event_source="" event_target="14"
    source_id="7" type_source="component" target_id="3" type_target="page">291857</interaction>
    <interaction id_int="2" initialState="416" finalState="417" event_source="14" event_target="390"
    source_id="3" type_source="page" target_id="3" type_target="page">291907</interaction>
    <interaction id_int="3" initialState="436" finalState="437" event_source="" event_target="409"
    source_id="161" type_source="component" target_id="5" type_target="page">306309</interaction>
    <interaction id_int="4" initialState="798" finalState="799" event_source="409" event_target="712"
    source_id="5" type_source="page" target_id="5" type_target="page">306359</interaction>
  ...
  </interactions>
</site>
```

A.3.2 *Site* B

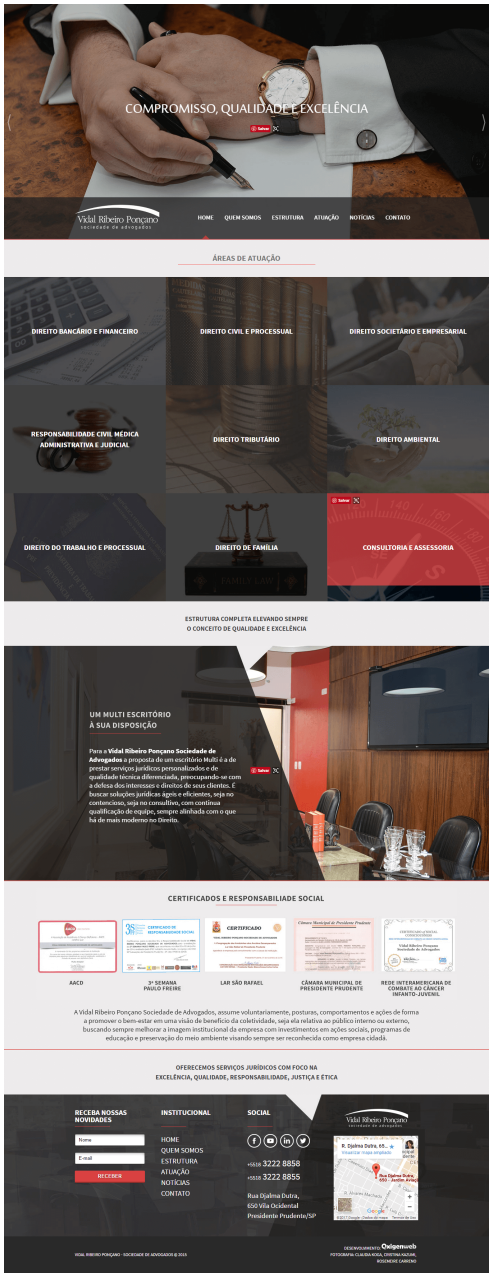

Figura A.4: Página inicial do *site* Vidal Ribeiro Ponçano utilizado no 2º experimento.

Listagem A.7: Estrutura do arquivo XML criado pelo *crawler* do *site* Vidal Ribeiro Ponçano

```
<?xml version="1.0" encoding="UTF-8"?>
<site url="http://www.vidalrp.adv.br/" titulo="Vidal Ribeiro Ponçano - Advogados Presidente
Prudente" tipo="crawler">
  <pages>
    <page url="http://www.vidalrp.adv.br/" titulo="Vidal Ribeiro Ponçano - Advogados
    Presidente Prudente" node_id="1" index="true">
      <event name="onLoad" node_id="1" item_id="null" event_id="1"/>
      <state name="onLoad" node_id="1" item_id="null" state_id="1"/>
      <state name="Load" node_id="1" item_id="null" state_id="2"/>
    <component type="div" dom_id="5" node_id="1" item_id="1" name=" MULTI
```

```
␣␣␣␣␣␣ESCRITÓRIO␣A␣SUA␣DISPOSIÇÃO␣PARCERIA␣DE␣SUCESSO␣␣COMPROMISSO,␣QUALIDADE␣E
␣␣␣␣␣␣EXCELÊNCIA␣">
        <event name="click" node_id="1" item_id="1" event_id="2"><![CDATA[Função Javascript]]>
        </event>
        <state name="selected" node_id="1" item_id="1" state_id="3"></state>
        <state name="notselected" node_id="1" item_id="1" state_id="4"></state>
      </component>
    ...
      <component type="link" dom_id="117" node_id="15" item_id="497" name="" externo="true">
        <event name="click" node_id="15" item_id="497" event_id="1154">
        <![CDATA[http://www.oxigenweb.com.br/]]></event>
        <event name="enter" node_id="15" item_id="497" event_id="1155">
        <![CDATA[http://www.oxigenweb.com.br/]]></event>
        <state name="not␣visited" node_id="15" item_id="497" state_id="1315"></state>
        <state name="visited" node_id="15" item_id="497" state_id="1316"></state>
      </component>
    </page>
  </pages>
  <edges>
    <edge ed_id="1" source="1" target="1" ref_item_id="2">
      <data event_id="3">click</data>
      <data event_id="4">enter</data>
    </edge>
    <edge ed_id="2" source="1" target="1" ref_item_id="3">
      <data event_id="5">click</data>
      <data event_id="6">enter</data>
    </edge>
  ...
  </edges>
  <structure>
    <page url="http://www.vidalrp.adv.br/" titulo="Vidal␣Ribeiro␣Ponçano␣–␣Advogados␣Presidente
␣␣␣␣Prudente" node_id="1" index="true">
      <node name="div" class="slide-home" id="" node_id="1" type="" item_id="" str_id="1">
        <node name="div" class="ls-wp-fullwidth-container" id="" node_id="1" type="" item_id=""
         str_id="2">
          <node name="div" class="ls-wp-fullwidth-helper" id="" node_id="1" type="" item_id=""
          str_id="3">
            <node name="component" id="layerslider_1" class="ls-wp-container␣ls-container
␣␣␣␣␣␣␣␣␣␣␣␣ls-fullwidth" type="div" dom_id="5" node_id="1" item_id="1"  externo="undefined"
              str_id="4">
            <node name="component" id="" class="ls-nav-prev" type="link" dom_id="21" node_id="1"
             item_id="2" externo="false"  str_id="6"></node>
            <node name="component" id="" class="ls-nav-next" type="link" dom_id="22" node_id="1"
             item_id="3" externo="false"  str_id="7"></node>
          </node>
        </node>
      </node>
    ...
    </page>
  </structure>
</site>
```

Listagem A.8: Estrutura do arquivo XML criado pelo *tracer* do *site* Vidal Ribeiro Ponçano

```
<?xml version="1.0" encoding="UTF-8"?>
<site url="http://www.vidalrp.adv.br/" titulo="Vidal␣Ribeiro␣Ponçano␣–␣Advogados␣Presidente␣Prudente"
tipo="tracer">
  <pages>
    <page url="http://www.vidalrp.adv.br/" titulo="Vidal␣Ribeiro␣Ponçano␣–␣Advogados␣Presidente
␣␣␣␣Prudente"
```

```
      node_id="1" index="true" novo="false">
       <event name="onLoad" node_id="1" item_id="null" event_id="1" novo="false"/>
       <state name="onLoad" node_id="1" item_id="null" state_id="1" novo="false"/>
       <state name="Load" node_id="1" item_id="null" state_id="2" novo="false"/>
       <component type="div" dom_id="5" node_id="1" item_id="1" name="␣MULTI␣ESCRITÓRIO
␣␣␣␣␣A␣SUA␣DISPOSIÇÃO
␣␣␣␣␣␣␣␣PARCERIA␣DE␣SUCESSO␣␣COMPROMISSO,␣QUALIDADE␣E␣EXCELÊNCIA␣" novo="false">
         <event name="click" node_id="1" item_id="1" event_id="2" novo="false">
         <![CDATA[Função Javascript]]></event>
         <state name="selected" node_id="1" item_id="1" state_id="3" novo="false"/>
         <state name="notselected" node_id="1" item_id="1" state_id="4" novo="false"/>
       </component>
   ...
       <component type="link" dom_id="117" node_id="15" item_id="497" name="" externo="true"
       novo="false">
         <event name="click" node_id="15" item_id="497" event_id="1154" novo="false">
         <![CDATA[http://www.oxigenweb.com.br/]]></event>
         <event name="enter" node_id="15" item_id="497" event_id="1155" novo="false">
         <![CDATA[http://www.oxigenweb.com.br/]]></event>
         <state name="not␣visited" node_id="15" item_id="497" state_id="1315" novo="false"/>
         <state name="visited" node_id="15" item_id="497" state_id="1316" novo="false"/>
       </component>
     </page>
   </pages>
   <interactions>
     <interaction id_int="1" initialState="13" finalState="14" event_source="" event_target="11"
     source_id="6" type_source="component" target_id="2" type_target="page">20635</interaction>
     <interaction id_int="2" initialState="107" finalState="108" event_source="11" event_target="95"
     source_id="2" type_source="page" target_id="2" type_target="page">20685</interaction>
     <interaction id_int="3" initialState="119" finalState="120" event_source="" event_target="106"
     source_id="49" type_source="component" target_id="5" type_target="page">29272</interaction>
     <interaction id_int="4" initialState="371" finalState="372" event_source="106" event_target="330"
     source_id="5" type_source="page" target_id="5" type_target="page">29322</interaction>
     ...
   </interactions>
</site>
```

A.4 3ª Avaliação

A.4.1 *Site* A

Figura A.5: Página inicial do *site* Malta Cobranças utilizado no 3º experimento.

Listagem A.9: Estrutura do arquivo XML criado pelo *crawler* do *site* Malta Cobranças

```
<?xml version="1.0" encoding="UTF-8"?>
<site url="http://www.maltacobrancas.com.br/" titulo="Malta Cobranças" tipo="crawler">
  <pages>
    <page url="http://www.maltacobrancas.com.br/" titulo="Malta Cobranças" node_id="1" index="true">
      <event name="onLoad" node_id="1" item_id="null" event_id="1"/>
      <state name="onLoad" node_id="1" item_id="null" state_id="1"/>
      <state name="Load" node_id="1" item_id="null" state_id="2"/>
      <component type="link" dom_id="5" node_id="1" item_id="1" name="" externo="false">
        <event name="click" node_id="1" item_id="1" event_id="2">
        <![CDATA[http://www.maltacobrancas.com.br/]]></event>
        <event name="enter" node_id="1" item_id="1" event_id="3">
        <![CDATA[http://www.maltacobrancas.com.br/]]></event>
        <state name="not visited" node_id="1" item_id="1" state_id="3"></state>
        <state name="visited" node_id="1" item_id="1" state_id="4"></state>
      </component>
    ...
    </page>
  </pages>
  <edges>
    <edge ed_id="1" source="1" target="1" ref_item_id="1">
```

```
      <data event_id="2">click</data>
      <data event_id="3">enter</data>
    </edge>
    <edge ed_id="2" source="1" target="1" ref_item_id="6">
      <data event_id="11">click</data>
      <data event_id="12">enter</data>
    </edge>
  ...
  </edges>
  <structure>
    <page url="http://www.maltacobrancas.com.br/" titulo="Malta␣Cobranças" node_id="1" index="true">
      <node name="div" class="site" id="page" node_id="1" type="" item_id="" str_id="1">
        <node name="header" class="site-header" id="masthead" node_id="1" type="" item_id=""
        str_id="2">
        <node name="div" class="" id="top-header" node_id="1" type="" item_id="" str_id="3">
  ...
    </page>
    <page url="http://www.maltacobrancas.com.br/portfolio.html" titulo="Malta␣Portifólio␣Online"
    node_id="16" index="false">
    </page>
  </structure>
</site>
```

Listagem A.10: Estrutura do arquivo XML criado pelo *tracer* do *site* Malta Cobranças

```
<?xml version="1.0" encoding="UTF-8"?>
<site url="http://www.maltacobrancas.com.br/" titulo="Malta␣Cobranças" tipo="tracer">
  <pages>
    <page url="http://www.maltacobrancas.com.br/" titulo="Malta␣Cobranças" node_id="1" index="true"
    novo="false">
      <event name="onLoad" node_id="1" item_id="null" event_id="1" novo="false"/>
      <state name="onLoad" node_id="1" item_id="null" state_id="1" novo="false"/>
      <state name="Load" node_id="1" item_id="null" state_id="2" novo="false"/>
    <component type="link" dom_id="5" node_id="1" item_id="1" name="" externo="false" novo="false">
      <event name="click" node_id="1" item_id="1" event_id="2" novo="false">
      <![CDATA[http://www.maltacobrancas.com.br/]]></event>
      <event name="enter" node_id="1" item_id="1" event_id="3" novo="false">
      <![CDATA[http://www.maltacobrancas.com.br/]]></event>
      <state name="not␣visited" node_id="1" item_id="1" state_id="3" novo="false"/>
      <state name="visited" node_id="1" item_id="1" state_id="4" novo="false"/>
    </component>
...
  </pages>
  <interactions>
    <interaction id_int="3" initialState="17" finalState="18" event_source="" event_target="15"
    source_id="8" type_source="component" target_id="3" type_target="page">59377</interaction>
    <interaction id_int="4" initialState="113" finalState="114" event_source="15" event_target="104"
    source_id="3" type_source="page" target_id="3" type_target="page">59427</interaction>
    <interaction id_int="5" initialState="139" finalState="140" event_source="" event_target="128"
    source_id="67" type_source="component" target_id="8" type_target="page">18772</interaction>
    <interaction id_int="6" initialState="325" finalState="326" event_source="128" event_target="296"
    source_id="8" type_source="page" target_id="8" type_target="page">18822</interaction>
  ...
  </interactions>
</site>
```

A.4.2 *Site* B

Figura A.6: Página inicial do *site* NW Cobranças utilizado no 3º experimento.

Listagem A.11: Estrutura do arquivo XML criado pelo *crawler* do *site* NW Cobranças

```
<?xml version="1.0" encoding="UTF-8"?>
<site url="http://www.nwcobrancas.com.br/" titulo="
Bem␣vindo␣-␣NW␣-␣Soluções␣ágeis␣e␣personalizadas␣-␣Equipes␣de␣alta␣performance␣prontos␣a␣maximizar␣o
caixa␣de␣sua␣empresa
" tipo="crawler">
  <pages>
    <page url="http://www.nwcobrancas.com.br/" titulo="
␣␣␣␣Bem␣vindo␣-␣NW␣-␣Soluções␣ágeis␣e␣personalizadas␣-␣Equipes␣de␣alta␣performance␣prontos␣a␣maximizar
␣␣␣␣o␣caixa␣de␣sua␣empresa
```

```
␣␣␣␣" node_id="1" index="true">
        <event name="onLoad" node_id="1" item_id="null" event_id="1"/>
        <state name="onLoad" node_id="1" item_id="null" state_id="1"/>
        <state name="Load" node_id="1" item_id="null" state_id="2"/>
      <component type="link" dom_id="3" node_id="1" item_id="1" name="0800␣940␣1243" externo="false">
        <event name="click" node_id="1" item_id="1" event_id="2">
        <![CDATA[http://www.nwcobrancas.com.br/]]></event>
        <event name="enter" node_id="1" item_id="1" event_id="3">
        <![CDATA[http://www.nwcobrancas.com.br/]]></event>
        <state name="not␣visited" node_id="1" item_id="1" state_id="3"></state>
        <state name="visited" node_id="1" item_id="1" state_id="4"></state>
      </component>
      <component type="link" dom_id="6" node_id="1" item_id="2" name="" externo="true">
        <event name="click" node_id="1" item_id="2" event_id="4">
        <![CDATA[https://www.facebook.com/NW-Administradora-
        Solu%C3%A7%C3%B5es-Integradas-em-Cobran%C3%A7as-934889299935447/?fref=ts]]></event>
        <event name="enter" node_id="1" item_id="2" event_id="5">
        <![CDATA[https://www.facebook.com/NW-Administradora-
        Solu%C3%A7%C3%B5es-Integradas-em-Cobran%C3%A7as-934889299935447/?fref=ts]]></event>
        <state name="not␣visited" node_id="1" item_id="2" state_id="5"></state>
        <state name="visited" node_id="1" item_id="2" state_id="6"></state>
      </component>
    ...
    </page>
  </pages>
  <edges>
    <edge ed_id="1" source="1" target="1" ref_item_id="1">
      <data event_id="2">click</data>
      <data event_id="3">enter</data>
    </edge>
    <edge ed_id="2" source="1" target="1" ref_item_id="5">
      <data event_id="10">click</data>
      <data event_id="11">enter</data>
    </edge>
  ...
  </edges>
  <structure>
    <page url="http://www.nwcobrancas.com.br/" titulo="
␣␣␣␣Bem␣vindo␣-␣NW␣-␣Soluções␣ágeis␣e␣personalizadas␣-␣Equipes␣de␣alta␣performance␣prontos
␣␣␣␣a␣maximizar␣o␣caixa␣de␣sua␣empresa
␣␣␣␣" node_id="1" index="true">
      <node name="div" class="bar-top-icons" id="" node_id="1" type="" item_id="" str_id="1">
        <node name="div" class="container" id="" node_id="1" type="" item_id="" str_id="2">
          <node name="span" class="" id="" node_id="1" type="" item_id="" str_id="3">
            <node name="component" id="" class="" type="link" dom_id="3" node_id="1"
            item_id="1" externo="false"  str_id="4"></node>
            <node name="component" id="" class="" type="link" dom_id="6" node_id="1"
            item_id="2" externo="true"  str_id="5"></node>
            <node name="component" id="" class="" type="link" dom_id="8" node_id="1"
            item_id="3" externo="true"  str_id="6"></node>
          </node>
        </node>
      </node>
    ...
    </page>
  </structure>
</site>
```

Listagem A.12: Estrutura do arquivo XML criado pelo *tracer* do *site* NW Cobranças

```
<?xml version="1.0" encoding="UTF-8"?>
<site url="http://www.nwcobrancas.com.br/" titulo="
```

```
Bem␣vindo␣–␣NW␣–␣Soluções␣ágeis␣e␣personalizadas␣–␣Equipes␣de␣alta␣performance␣prontos␣a␣maximizar
o␣caixa␣de␣sua␣empresa
" tipo="tracer">
  <pages>
    <page url="http://www.nwcobrancas.com.br/" titulo="␣␣Bem␣vindo␣–␣NW␣–␣Soluções␣ágeis␣e
␣␣␣␣personalizadas␣–␣Equipes␣de␣alta␣performance␣prontos␣a␣maximizar␣o␣caixa␣de␣sua␣empresa␣"
    node_id="1" index="true"  novo="false">
      <event name="onLoad" node_id="1" item_id="null" event_id="1" novo="false"/>
      <state name="onLoad" node_id="1" item_id="null" state_id="1" novo="false"/>
      <state name="Load" node_id="1" item_id="null" state_id="2" novo="false"/>
      <component type="link" dom_id="3" node_id="1" item_id="1" name="0800␣940␣1243" externo="false"
      novo="false">
        <event name="click" node_id="1" item_id="1" event_id="2" novo="false">
        <![CDATA[http://www.nwcobrancas.com.br/]]></event>
        <event name="enter" node_id="1" item_id="1" event_id="3" novo="false">
        <![CDATA[http://www.nwcobrancas.com.br/]]></event>
        <state name="not␣visited" node_id="1" item_id="1" state_id="3" novo="false"/>
        <state name="visited" node_id="1" item_id="1" state_id="4" novo="false"/>
      </component>
  ...
    </page>
  </pages>
  <interactions>
    <interaction id_int="3" initialState="21" finalState="22" event_source="" event_target="20"
    source_id="10" type_source="component" target_id="4" type_target="page">22814</interaction>
    <interaction id_int="4" initialState="129" finalState="130" event_source="20" event_target="120"
    source_id="4" type_source="page" target_id="4" type_target="page">22864</interaction>
    <interaction id_int="5" initialState="151" finalState="152" event_source="" event_target="141"
    source_id="66" type_source="component" target_id="5" type_target="page">18005</interaction>
    <interaction id_int="6" initialState="167" finalState="168" event_source="141" event_target="155"
    source_id="5" type_source="page" target_id="5" type_target="page">18055</interaction>
  ...
  </interactions>
</site>
```