Utilização de Técnicas de Integração de Software para Aplicações Web no Contexto de Ferramentas de Acessibilidade

Wesley Tessaro Andrade

Utilização de Técnicas de Integração de Software para Aplicações Web no Contexto de Ferramentas de Acessibilidade

Wesley Tessaro Andrade

Orientadora: *Profª Drª Débora Maria Barroso Paiva*

Versão final apresentada ao programa de Pós-graduação em Ciência da Computação, da Universidade Federal de Mato Grosso do Sul como requisito para obtenção do título de Mestrado em Ciência da Computação.

UFMS - Campo Grande
Julho/2017

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ANSI *American National Standards Institute*

ATAG *Authoring Tool Accessibility Guidelines*

CASE *Computer-Aided Software Engineering*

CBD *Component-Based Development*

CE Critérios de Exclusão

CI Critérios de Inclusão

CORBA *Common Object Request Broker Architecture*

CSA *Central Service Application*

CBD *Component-Based Development*

DSL Linguagem de Definição de Interfaces

EAI *Enterprise Application Integration*

ebXML *Electronic Business using eXtensible Markup Language*

EDI *Eletronic Data Interchang*

eMAG Modelo de Acessibilidade em Governo Eletrônico

ERP *Enterprise Resource Planning*

HREOC *Human Rights and Equal Opportunity Commission*

IBGE Instituto Brasileiro de Geografia e Estatística

ICA *O Intelligent Core Adapters*

IDE Ambiente de Desenvolvimento Integrado

IEEE *Institute of Electrical and Electronics Engineers*

IETF *Internet Engineering Task Force*

ISO *International Organization for Standardization*

J2EE *Java Platform, Enterprise Edition*

MS Mapeamento Sistemático

OMG *Object Management Group*

OMS Organização Mundial da Saúde

ONG Organização Não-Governamental

OSLC *Open Services for Life-cycle Collaboration*

QP Questões de Pesquisa

RS Revisão Sistemática

TA Tecnologias Assistivas

UAAG *User Agent Accessibility Guidelines*

UFMS Universidade Federal de Mato Grosso do Sul

UIT União Internacional de Telecomunicações

UML *Unified Modeling Language*

W3C *World Wide Web Consortium*

WAI *Web Accessibility Initiative*

WCAG Web Content Accessibility Guidelines

WEB *World Wide Web*

XML *Extensible Markup Language*

Abstract

Resumo

A crescente expansão da Web no mundo é um fenômeno que traz consigo vários desafios. A acessibilidade está diretamente relacionada à inclusão digital e ao bem estar social de uma grande parcela da população. Fornecer ferramentas e metodologias de apoio à acessibilidade podem aportar melhorias e minimizar problemas que hoje existem no cenário mundial. O propósito deste trabalho de mestrado é estudar técnicas de integração de software apresentadas na literatura e aplicá-las no contexto de ferramentas que concebem produtos acessíveis. Como resultado deste estudo é concebida uma ferramenta denominada *Acero*, que engloba as etapas do desenvolvimento de software com o objetivo de entregar um código acessível aos desenvolvedores. É conduzido um estudo de caso que indicou a viabilidade de sua utilização de acordo com os critérios analisados.x

The increasing expansion of the Web in the world is a phenomenon that brings multiple challenges in various segments of society. The accessibility is directly related to digital inclusion and social welfare of a large portion of the population. Despite several laws and recommendations there is still much to do. Provide tools and methodologies to support accessibility can contribute improvements and solve problems that exist today in the world scenario. The purpose of this work was to study the master's software integration techniques applied in the context of product design tools accessible. As a result of this study was designed a tool called de Acero comprising the steps of software development with the goal of delivering a code accessible to developers.

CAPÍTULO

1

Introdução

1.1 Contextualização

A crescente disponibilidade de computadores e equipamentos eletrônicos e a viabilidade do acesso à Internet trouxeram uma revolução em todos os âmbitos da sociedade moderna. A influência e a organização de informações contribuem para a multiplicação de conhecimentos e do alcance intelectual do ser humano. Em todos os setores da sociedade está presente algum tipo de tecnologia envolvendo computadores e redes de interconexão, o que contribui na troca e integração de informações [41].

Segundo a União Internacional de Telecomunicações (UIT[1]), órgão subordinado à Organização das Nações Unidas (ONU[2]), no ano de 2016 aproximadamente 3 bilhões de pessoas no mundo possuíam acesso a Internet, correspondendo cerca de 40 % da população mundial [17]. A inclusão digital implica em estar preocupado com as necessidades e demandas de todos os usuários, de um serviço ou sistema, incluindo aqueles com necessidades especiais de acesso.

A Organização Mundial da Saúde (OMS[3]) estimou em 2014, que aproximadamente 1 bilhão de pessoas no mundo tinham algum tipo de deficiência, variando de deficiências visuais e auditivas à dificuldades cognitivas e motoras [15]. No Brasil, segundo o último censo realizado desde então, pelo Instituto Brasileiro de Geografia e Estatística (IBGE) em 2010, existiam aproximadamente 45,6 milhões de pessoas enfrentando algum tipo de deficiência [42].

[1] http://www.itu.int/
[2] http://www.un.org/
[3] http://www.who.int/en/

É relativamente fácil para uma pessoa sem quaisquer dificuldades físicas, mentais e cognitivas, acessar a Internet e interagir com recursos eletrônicos, tais como, navegar, posicionar o *mouse* sobre palavras e selecioná-las, ler uma página Web, compreender uma imagem, mesmo que esta não contenha informações adicionais, como rótulos e legendas [59]. Entretanto, para pessoas com dificuldades motoras, por exemplo, doença de *Parkinson* e Esclerose Múltipla, ou outras dificuldades como deficiência visual, faz-se necessário que os recursos de mídia, como computadores, celulares, *tablets*, softwares e *websites* ofereçam suporte às necessidades de seus usuários, ou seja, que estes recursos e ferramentas sejam completamente acessíveis [59].

Existem muitas definições a respeito de acessibilidade. De forma geral, acessibilidade na Web significa que pessoas com deficiência consigam acessar, navegar, interagir e contribuir com informações na Web. Ainda nesse escopo, a acessibilidade engloba auxiliar pessoas que disponham de capacidades reduzidas, como idosos [6].

Apesar da grande necessidade e demanda em oferecer recursos que possibilitem a inclusão digital, sob diversos âmbitos e perfis de usuários, a acessibilidade na Web nem sempre foi prioridade [28]. Contudo, atualmente as empresas e desenvolvedores estão notando que aqueles que negligenciam o desenvolvimento de produtos acessíveis podem perder uma expressiva quantidade de clientes [47].

Com o objetivo de contribuir na inclusão digital de forma igualitária, o órgão regulamentador da Web, o *World Wide Web Consortium* (W3C[4]), criou um grupo internacionalmente renomado no âmbito de acessibilidade denominado *Web Accessibility Initiative* (WAI[5]), cuja principal missão é desenvolver estratégias, diretrizes e recursos que auxiliem na implementação de *websites* e conteúdos acessíveis. Para promoção da inclusão e no intuito de fornecer recursos acessíveis na Web, este grupo estabeleceu um conjunto de diretrizes, encontrando-se atualmente na versão 2.0, a *Web Content Accessibility Guidelines* (WCAG[6]), que apoia o desenvolvimento de conteúdos acessíveis [6]. Contudo, essas diretrizes não impedem a criação de outras diretrizes e recomendações de acessibilidade por outros órgãos e países.

No dia 7 de Julho de 2015, foi publicado no Diário Oficial da União, o Estatuto da Pessoa com Deficiência[7] que tornou obrigatória a acessibilidade digital nos sítios da internet mantidos por empresas com sede ou representação comercial no País ou por órgãos de governo. Apesar dos esforços empreendidos para a regulamentação de acessibilidade Web, a maioria das páginas Web não

[4]https://www.w3.org/
[5]https://www.w3.org/WAI/
[6]https://www.w3.org/WAI/intro/wcag
[7]http://www.planalto.gov.br/ccivil_03/_Ato2015−2018/2015/Lei/L13146.htm

é acessível, inclusive no âmbito nacional [62]. Pesquisadores indicam que esse problema não ocorre somente por falta de conhecimento técnico, mas também pela falta de conscientização dos desenvolvedores e *designers* envolvidos na construção de conteúdos Web [56]. Outro problema considerável é que as ferramentas de software que ajudam os profissionais a aplicarem acessibilidade em seus produtos ainda encontram-se em estágios iniciais, havendo a necessidade de serem aprimoradas para que seja possível incorporar a acessibilidade durante todo o processo de desenvolvimento [46] [55].

A Engenharia de Software tem papel crucial na concepção de produtos acessíveis de qualidade, pois, a ausência de metodologias e processos bem definidos podem resultar em produtos finais não acessíveis. Além disso, os custos do desenvolvimento de produtos acessíveis são menores se a acessibilidade for considerada desde as etapas iniciais que, por conseguinte, resulta em um produto com valor agregado maior [39] [61] [49].

Durante o desenvolvimento de um produto de software é necessário realizar diversas tarefas e ações. Dentre estas necessidades estão, por exemplo, a especificação de requisitos, codificação e a realização de testes. Contudo, não é possível automatizar totalmente as tarefas e os processos necessários à concepção de um software, sendo primordial a intervenção humana. No entanto, é possível utilizar os computadores para auxiliar todo o processo de desenvolvimento de software. Neste contexto, o termo CASE (*Computer-Aided Software Engineering*) refere-se a uma classificação de ferramentas de software que auxiliam no processo de desenvolvimento de produtos. Os computadores podem aportar auxílio em tarefas repetitivas e de apoio ao desenvolvedor, com o objetivo de diminuir a complexidade do desenvolvimento e contribuir na produtividade e qualidade.

A integração de ferramentas no contexto do desenvolvimento de soluções acessíveis pode aportar distintos benefícios, além de preencher lacunas existentes. Como supracitado, um dos principais motivos para a falta de acessibilidade em *websites* é a limitação das ferramentas que apoiam a acessibilidade no processo de desenvolvimento.

Por meio de estudos direcionados à integração e acessibilidade, é possível oferecer aos desenvolvedores um ambiente de desenvolvimento mais amplo e completo, integrando distintas ferramentas de apoio a acessibilidade [46]. Este trabalho apresenta a utilização de técnicas de integração aplicadas no contexto do desenvolvimento de soluções acessíveis. Além disso, são propostas algumas soluções para integração de ferramentas já existentes.

1.2 Justificativa

Diante deste tema relevante e essencial à inclusão digital, o grupo de Engenharia de Software da Universidade Federal de Mato Grosso do Sul (UFMS) está trabalhando há alguns anos com o objetivo de produzir ferramentas e conhecimentos que promovam a acessibilidade.

Durante este período de pesquisa foram construídas ferramentas de apoio ao desenvolvedor de produtos acessíveis, por exemplo o trabalho desenvolvido por *Branco* [30] gerou uma ferramenta denominada *AccTrace* que permite acompanhar a evolução dos requisitos de acessibilidade até a fase de codificação, fornecendo ao desenvolvedor informações relevantes para a construção de um produto acessível. Outro recurso de apoio à acessibilidade desenvolvido no grupo foi um *framework*[8], denominado *Homero* [32], desenvolvido utilizando a linguagem PHP. O trabalho considerou as diretrizes automatizáveis da WCAG e seu principal objetivo foi facilitar a criação de interfaces web acessíveis.

Por meio da revisão bibliográfica realizada neste trabalho foi percebido que a rastreabilidade dos requisitos em conjunto com o processo de desenvolvimento de *software* ainda é um grande desafio. Além da ferramenta *AccTrace*, não foi encontrada outra ferramenta de apoio que fornecesse auxílio tão amplo no processo de desenvolvimento de produtos acessíveis. Em conjunto com o *framework Homero* foi possível ampliar a abrangência da ferramenta *AccTrace* fornecendo suporte não só à linguagem de programação *Java* como também à linguagem *PHP* e *HTML*.

Além disso, foi vislumbrada a possibilidade de integrar as duas ferramentas *CASE* sob o mesmo ambiente de desenvolvimento, com o objetivo de obter uma solução mais robusta, ampla e única, que possa ajudar o desenvolvedor na criação de aplicações acessíveis.

1.3 Objetivos

O objetivo geral deste trabalho foi fornecer um arcabouço computacional para apoiar o desenvolvimento de aplicações *web* acessíveis.

Os objetivos específicos foram:

- Realizar estudos sobre integração em ferramentas CASE.
- Identificar técnicas e propor mecanismos de integração para as ferramentas desenvolvidas pelo grupo de pesquisa em acessibilidade da UFMS que tem por objetivo promover a acessibilidade.

[8] Um *framework* é um esquema ou padrão, que tem o objetivo de facilitar a implementação de um produto [32].

- Integrar as ferramentas *Homero* e *AccTrace*, com o objetivo de ofertar um ambiente de desenvolvimento mais amplo e robusto.
- Realizar experimento com usuários visando avaliar a funcionalidade da solução desenvolvida.

Buscou-se neste trabalho utilizar a integração de ferramentas CASE de apoio à acessibilidade visando ofertar aos desenvolvedores de conteúdos acessíveis um ambiente mais robusto, amplo e integrado. Como resultados, espera-se que os recursos desenvolvidos e integrados apoiem de fato o desenvolvimento de aplicações acessíveis.

1.4 Organização do Texto

Este trabalho está organizado em seis capítulos. O primeiro capítulo apresentou o objetivo do trabalho, desafios presentes e motivações para melhoria do estado da arte.

O segundo capítulo apresenta os principais conceitos referentes a acessibilidade no contexto da Web. Esse capítulo apresenta definições sobre acessibilidade, deficiência, tecnologias de apoio, legislações e diretrizes que promovem a acessibilidade de produtos Web. Além disso, são apresentados métodos e técnicas de avaliação de acessibilidade em produtos.

O terceiro capítulo expõe o levantamento bibliográfico a respeito de integração de ferramentas, no contexto de ferramentas CASE. Nesse capítulo são levantadas técnicas, modelos e considerações a respeito de integração.

O quarto capítulo apresenta a proposta e o desenvolvimento de uma ferramenta de integração entre as ferramentas *AccTrace* e o *framework* Homero, denominada de Acero. Além disso, apresenta uma prova de conceito, demonstrando a utilização da ferramenta integrada, bem como sua efetividade e limitações.

O quinto capítulo apresenta o estudo de caso realizado com usuários reais, com o objetivo de analisar a solução proposta quanto a funcionalidade e eficiência no desenvolvimento de aplicações web acessíveis.

O sexto e último capítulo apresenta as conclusões gerais e desafios futuros a serem trabalhados.

CAPÍTULO

2

Acessibilidade na Web: Fundamentação Teórica e Trabalhos Relacionados

2.1 Considerações Iniciais

A Web constantemente está oferecendo acesso à informações e interações a conteúdos dinâmicos, utilizando por exemplo, as redes sociais. Além disso, oferece oportunidades de comunicação relevantes, de modo que não seriam possíveis por outras formas. [59].

Infelizmente, a acessibilidade não está presente em todas as partes da Web. A maioria dos sítios da Web não oferece seus recursos de forma acessível, o que torna difícil ou praticamente impossível o acesso e a contribuição de informações de pessoas com deficiência [59] [62].

Neste capítulo são apresentadas as principais concepções e conceitos pertinentes à acessibilidade Web. São apresentadas tecnologias adaptativas, que são ferramentas ou dispositivos que auxiliam pessoas com deficiência e também são apresentadas diretrizes e recomendações que permeiam a acessibilidade no universo digital a fim fornecer conteúdos acessíveis.

2.2 Deficiência e Incapacidade Funcional

Segundo a Organização Mundial de Saúde (OMS), deficiência é o termo utilizado para se referir a indivíduos que tenham o funcionamento limitado ou

ausência da estrutura anatômica, fisiológica e/ou intelectual [15].

Segundo o Decreto Nacional Nº 3298, de 20 de Dezembro de 1999 [1] e a Lei Brasileira de Inclusão - Lei Nº 13.146, De 6 DE Julho de 2015 [2], são consideradas pessoas com deficiência aquelas pessoas que se enquadram em pelo menos uma das seguintes categorias:

- Deficiência Física: Alteração parcial ou completa de uma ou mais partes do corpo humano, conduzindo a perda parcial ou total das funções físicas.
- Deficiência Sensorial: A deficiência sensorial é caracterizada pela alteração parcial ou completa de algum dos cinco sentidos (visão, audição, paladar, tato e olfato).
- Deficiência Intelectual: Funcionamento intelectual inferior à média, afetando a capacidade adaptativa do indivíduo.

Sendo assim, a incapacidade funcional engloba diversos aspectos inerentes à natureza do indivíduo, tais como condições cognitivas, emocionais, físicas e mentais. Refere-se também, a incapacidade parcial ou total para desempenhar tarefas cotidianas, básicas ou necessárias à vida independente, como a comunicação e mobilidade [21]. Ademais disso, a acessibilidade percorre um viés ainda mais amplo, considerando também atender indivíduos com deficiências situacionais ou temporários, por exemplo, quando um usuário utiliza o celular no inverno e a mobilidade dos dedos é prejudicada.

2.2.1 Tecnologias Assistivas

Segundo o Comitê de Ajudas Técnicas (CAT), tecnologia Assistiva (TA) é uma área de conhecimento, com característica interdisciplinar, que engloba distintos produtos, recursos, métodos, estratégias e serviços que promovem e auxiliam as pessoas com deficiência ou mobilidade reduzida visando sua autonomia, qualidade de vida e inclusão digital [29]. Refere-se também a qualquer produto (incluindo dispositivos, instrumentos, tecnologias e softwares) disponibilizado com o objetivo de prevenir, compensar, controlar, mitigar ou neutralizar deficiências e limitações [50].

Existem diversos objetos que podem auxiliar pessoas com necessidades especiais a realizarem atividades do cotidiano, como a locomoção, comunicação, estudo, trabalho e lazer. Cadeiras de rodas, bengalas, software preditor de palavras e *display Braile* são apenas alguns exemplos de recursos de Tecnologia Assistiva [50] [29].

[1] http://www.planalto.gov.br/ccivil_03/decreto/d3298.htm
[2] http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2015/Lei/L13146.htm

A Tabela 2.1 apresenta quatro classes de deficiências e dificuldades, principais obstáculos e alguns exemplos de recursos de Tecnologia Assistiva que podem auxiliar seus usuários na utilização do computador [38].

Tabela 2.1: Exemplos de TA para auxiliar pessoas com limitações e deficiências na utilização do computador. *Adaptado de [38].*

Tipo de limitação	**Experiência do Usuário Sem TA**	**Possíveis Soluções de TA**
Visual		
Leve (baixa visão, daltonismo)	Dificuldade na leitura de conteúdos	* Alterar tamanho de fontes e cores *Alterar estilo de fontes *Utilizar telas maiores
Grave (cegueira)	Incapacidade de utilizar o monitor do computador, necessita receber as informações por meio da audição.	*Leitor de tela (síntese de voz e sinais sonoros) *Audiodescrição de vídeos *Display Braile *Navegação pelo teclado
Motora		
Leve (dor temporária, destreza motora reduzida como um braço fraturado) a grave (paralisia)	A utilização do mouse padrão e do teclado é dolorosa e difícil.	*Ajuste fino do mouse e teclado, para torná-los mais sensíveis ao usuário *Teclado e mouse virtual *Software de reconhecimento de fala *Dispositivos de entrada alternativos, como Joystick e mouse Head Tracking
Auditiva		
Leve (dificuldade de ouvir) a grave (surdez)	Dificuldade ou impossibilidade para distinguir palavras e sons, necessita receber informações de forma visual	*Ajuste de volume *Sons integrados a avisos visuais *Utilizar legendas em conteúdos de multimídia *Utilizar línguas de sinais
Mental ou Cognitiva		
Leve (dificuldade de aprendizado) a grave (Alzheimer's, demência)	Dificuldade em reconhecer e memorizar palavras, dificuldade de atenção e de concentração.	*Softwares preditores de Palavras *Fala combinada com recursos visuais *Interface de Usuário Simplificada *Lembrete de eventos

Diante das dificuldades enfrentadas por esses usuários, torna-se importante fornecer meios de acesso e tecnologias intermediárias à interação com os conteúdos digitais, amparando assim, na equidade da inclusão digital [38].

Na próxima seção é apresentado um panorama geral sobre legislações referentes à acessibilidade no âmbito de tecnologia da informação, tanto no cenário nacional quanto internacional.

2.3 Legislação sobre acessibilidade na Web

A utilização da Web está presente na maioria dos setores da sociedade e na vida pessoal dos indivíduos. Em muitos países, a Web tem se tornado a principal fonte para obtenção de informações e acesso a serviços governamentais, educacionais, noticiários, de lazer e de muitos outros elementos. Consequentemente, sua utilização está substituindo, ou diminuindo, a utilização de recursos que outrora eram fortemente utilizados, como jornais e revistas em versões impressas. Portanto, torna-se necessário que a Web seja acessível

a fim de propiciar igualdade, oportunidade de acesso e interação de todas as pessoas que necessitem utilizá-la [6] [50].

Diante desta grande necessidade, de fornecer conteúdos acessíveis na Web, diversos governos instituíram leis que estabelecem a disponibilização de informações acessíveis, mesmo que somente no escopo de sítios governamentais, com o intuito de garantir a igualdade de oportunidade no acesso. É comum essas nações adotarem diretrizes, internacionais ou nacionais, a fim de padronizar o desenvolvimento de produtos e conteúdos no país. Alguns exemplos de países que definem ser obrigatório a acessibilidade são:

- Austrália: O foco na acessibilidade Web adveio como resultado da lei *Disability Discrimination Act de 1992*, sob a seção 24, que estabelece ser ilegal a discriminação às pessoas com deficiência na oferta de serviços, instalações e bens. Entretanto, somente após o documento *World Wide Web Access: Disability Discrimination Act Notes consultivos*, criado em 2002 pela Comissão de Direitos Humanos e Igualdade de Oportunidades (HREOC), foi assegurado a acessibilidade Web em domínios públicos [1] [10].

- Brasil: Por meio do Decreto 5.296 de 2 de dezembro de 2004, as Leis 10.048/2000 e 10.098/2000, que preveem acessibilidade, foram regulamentadas, tornando obrigatória a disponibilização de Web Sites da administração pública acessíveis [4] [2] [3]. Além disso, a Lei 13.146 de 6 de Julho de 2015 instituiu a obrigatoriedade de que sítios da internet, mantidos por empresas com sede ou representação comercial no país, devem ofertar sítios web acessíveis conforme práticas e diretrizes de acessibilidade adotadas internacionalmente [16].

- União Europeia: No dia 26 de fevereiro de 2014 foi aprovada pelos deputados representantes da União Europeia a exigência de que seus estados membros garantam a acessibilidade em todos seus Web Sites públicos, contribuindo para a melhoria de cerca de 761 mil Web Sites dos setores públicos [14].

- Israel: O Ministério de Justiça Israelense regulamentou a norma 5568, em março de 2013, que estabelece que os Web Sites públicos israelenses sejam acessíveis. Entretanto os padrões israelenses estabelecidos são mais brandos, se comparados com padrões de outros países, devido as dificuldades técnicas, como a oferta de legendas e textos, em razão do idioma hebraico [12].

2.4 Documentos e padrões

Pelo fato da Internet ter extensão global, é necessário definir padrões e recomendações, com o objetivo de não a fragmentar, garantindo assim sua escalabilidade a longo prazo e evitando o monopólio de tecnologias. Neste contexto, duas expressivas organizações, ou grupos, destacam-se. A organização *The Internet Engineering Task Force* (IETF) tem caráter colaborativo em assuntos relacionados à infraestrutura e definição de protocolos de comunicação da Internet. O outro grupo, o W3C, é responsável por definir padrões para linguagens e protocolos a serem utilizados em aplicações [50] [52] .

Com o objetivo de melhorar a acessibilidade no contexto da Web, a organização W3C lançou uma iniciativa cujo principal esforço é coordenar os esforços internacionais, técnicos e humanos, na melhoria da acessibilidade Web. Esta iniciativa é denominada WAI [52] [50] [6].

A WAI elabora uma série de normas e diretrizes de acessibilidade, conhecidas como Componentes Essenciais de Acessibilidade na Web [6]. As principais normas e diretrizes serão descritas a seguir.

2.4.1 Diretrizes de Acessibilidade para Conteúdo Web

As Diretrizes de Acessibilidade para Conteúdo Web, ou *Web Content Accessibility Guidelines (WCAG)*, é um conjunto de documentos que auxilia e explica, através de diretrizes e recomendações, como tornar os conteúdos Web acessíveis à pessoas com deficiências. O WCAG é voltado para desenvolvedores de conteúdos, mas serve também para auxiliar desenvolvedores de ferramentas de avaliação, desenvolvedores de ferramentas de auditoria e os desenvolvedores de ferramentas de garantia de qualidade e validação.

O WCAG 1.0 foi publicado como o padrão vigente do W3C em Maio de 1999. Este documento é composto por 12 diretrizes, divididas em *checkpoints*, ou pontos de verificação e propriedades associados a cada um destes.

As prioridades são divididas em três grandes grupos:

- Nível 1: Os desenvolvedores **devem** satisfazer esses *checkpoints*, pois um ou mais grupos de pessoas poderão não conseguir acessar o conteúdo Web. Se todos os *checkpoints* associados a esta prioridade forem atendidos, o site receberá, nível de conformidade “A”.
- Nível 2: Os desenvolvedores **deveriam** satisfazer os *checkpoints*, de outro modo, um ou mais grupos terão dificuldades para acessar as informações. A conformidade com este nível é descrita como “AA”.
- Nível 3: Os desenvolvedores **poderiam** satisfazer esses requisitos para facilitar o acesso de alguns grupos. A conformidade com este nível é

referenciada como “AAA”.

Devido a evolução e criação de novas tecnologias Web, o W3C necessitava fornecer melhorias para atender estas novas ferramentas e possibilitar a escalabilidade do padrão WCAG 1.0. Esta foi a motivação para o desenvolvimento do padrão WCAG 2.0 [8].

A construção do padrão WCAG 2.0 foi baseada no WCAG 1.0 e conta com novas recomendações. Não obstante, algumas alterações também foram realizadas: um dos principais pontos de mudança foi que ao invés de cada diretriz possuir pontos de verificação, ou *checklists*, são associados 61 critérios de sucesso, que são declarações que podem ser testadas automaticamente, ou manualmente, a fim de verificar se o conteúdo Web é acessível ou não. É por meio destes critérios de sucessos que são estabelecidos os níveis de conformidade, “A”, “AA” ou “AAA” [45].

O WCAG 2.0 foi publicado em 11 de Dezembro de 2008 como o padrão atual do grupo W3C. É composto por 12 diretrizes organizadas sob quatro princípios fundamentais [7]. A seguir estes fundamentos são sucintamente apresentados:

- Perceptível: Os dados e componentes das interfaces devem ser apresentados aos seus usuários de maneira perceptível.
- Operável: Os componentes de interfaces de usuário devem ser operáveis, independente da necessidade que tenha o usuário.
- Compreensível: Os conteúdos e as operações sobre as interfaces devem ser compreensíveis.
- Robusta: Os conteúdos e as informações devem ser interpretadas de forma confiável pelas ferramentas que usuários utilizam, incluindo neste escopo recursos de Tecnologia Assistiva. Além disso, os usuários não devem ser prejudicados caso a tecnologia necessária à acessibilidade mude.

Há muitas razões que justificam o sucesso do padrão de acessibilidade WCAG. Uma das razões é que ele é fruto de um esforço multilateral, recebendo contribuições do público em todas as suas fases, favorecendo assim, a sua qualidade. Outro pressuposto importante é que este padrão promove a simplicidade em sua aplicabilidade, através de seus princípios supracitados. Os critérios de acessibilidade do WCAG vão além de conteúdos HTML, auxiliando também os desenvolvedores a utilizá-lo para projetar por exemplo, aquivos PDF acessíveis [45].

Embora o WCAG seja uma orientação eminentemente técnica, não obrigatória, alguns países utilizam, ou se inspiram neste padrão com o objetivo de

fornecer conteúdos acessíveis em seus portais governamentais. Alguns destes exemplos são: Israel [12], Canadá [11], Japão [9], Irlanda [5], Itália [13] e Brasil [34].

2.4.2 Diretrizes de Acessibilidade para as Ferramentas de Autoria

As ferramentas de autoria ou construção são softwares ou serviços que permitem a elaboração de sites e de conteúdos Web. Existem diversas ferramentas e serviços no escopo de autoria, cada qual com sua singularidade, para a criação, por exemplo, de *blogs* e páginas *wiki*. Não obstante, empresas podem criar suas próprias ferramentas de autoria, a fim de satisfazer suas próprias necessidades [18] [45]. As Diretrizes de Acessibilidade para as Ferramentas de Autoria, ou a *Authoring Tool Accessibility Guidelines (ATAG)*, são diretrizes e recomendações que orientam o desenvolvimento de ferramentas de autoria.

Existem dois objetivos prevalentes em se utilizar o ATAG. Em primeiro lugar o ATAG fornece recomendações sobre como desenvolver ferramentas de autoria acessíveis, com o objetivo de que os criadores de conteúdos, que possuem deficiência possam utilizar a ferramenta de forma independente. O segundo objetivo principal é auxiliar os desenvolvedores a criarem conteúdos acessíveis, em conformidade com o padrão WCAG [18] [45] [6].

Assim como o padrão WCAG, a recomendação ATAG possui diretrizes e pontos de verificação para auxiliar no desenvolvimento de ferramentas de autoria. Sua primeira versão, o documento ATAG 1.0, foi aprovado em fevereiro de 2000. A segunda versão do documento, a ATAG 2.0, foi proposta em Julho de 2015 e é compatível com o último padrão de acessibilidade vigente do W3C, o WCAG 2.0 [18] [45] [6].

A estrutura, conforme apresentado anteriormente, segue o padrão WCAG, definindo de igual forma, níveis de conformidades, “A”, “AA” ou “AAA”. O documento ATAG é dividido em duas partes, parte *A* e parte *B*. A parte *A* trata de tornar a própria ferramenta acessível. A parte *B* aborda a criação de conteúdo acessível. Lamentavelmente, o documento ATAG não é utilizado amplamente pela indústria de TI, como necessitava ser, tornando-se então uma necessidade atual inegável, visto a demanda de usuários e desenvolvedores [45].

Por exemplo, a Diretriz A.3.1, referente a interface de usuário da ferramenta de autoria, indica a necessidade de fornecer acesso aos recursos de criação por meio de atalhos de teclado. Esta diretriz tem o objetivo de diminuir o esforço empreendido por alguns usuários deficientes, que possuem alguma mobilidade reduzida.

2.4.3 Diretrizes de Acessibilidade para Agentes de Usuários

As Diretrizes de Acessibilidade para Agentes de Usuários, ou *User Agent Accessibility Guidelines (UAAG)*, é um conjunto de diretrizes projetadas para avaliar a acessibilidade das ferramentas que permitem seus usuários acessarem conteúdo Web, como *Media Players*, navegadores Web e tecnologias assistivas. Essas ferramentas são comumente conhecidas por *User Agents* ou Agentes de Usuário [6].

Voltada a desenvolvedores de ferramentas para o usuário final, a diretriz UAAG conta com duas versões. Uma aprovada em dezembro de 2002, a 1.0, é considerada atualmente uma versão estável e madura. Outra versão, a UAAG 2.0, encontra-se em desenvolvimento, entretanto, já é considerada como um esboço maduro e seu uso é fortemente recomendado pela W3C, pois, alinha-se com as diretrizes de acessibilidade WCAG 2.0. O UAAG segue a mesma estrutura e os níveis de conformidade das diretrizes WCAG [6]. Em sua versão mais recente, a UAAG 2.0, conta com vinte e cinco diretrizes, três orientações e considera cinco princípios: fazer o conteúdo Perceptível, Operável, Compreensível, permitir Acesso Programático e contribuir para Especificações e Convenções.

Os três primeiros princípios são coerentes aos que a diretriz WCAG 2.0 fornece. No entanto, os dois últimos princípios são específicos à diretriz UAAG 2.0. O princípio de Acesso Programático se refere a como o Agente de Usuário apoia a acessibilidade, amparando seus usuários em assuntos referentes a programação de conteúdos, como a modificação e gravação de informações em elementos, por exemplo, na edição de um campo. O quinto princípio, Especificações e Convenções, em suma, exige o cumprimento de normas de acessibilidade, destacando-se o padrão WCAG 2.0 [6].

2.5 Métodos de Avaliação e Medição

As legislações e os padrões discutidos nas seções anteriores fornecem unicamente orientações técnicas do que deve ser oferecido no projeto e alcançam apenas objetivos de alto nível [45].

Os métodos de avaliação encontram-se em um nível técnico mais detalhado e correspondem a identificação de falhas e erros específicos na codificação e projeto. Para uma maior eficácia, os métodos de avaliação, que serão explanados, devem ser utilizados em conjunto com diretrizes e padrões de acessibilidade existentes. Muitos métodos para a avaliação de acessibilidade podem ser utilizados, tais como testes de usuário, revisões de especialistas e avaliações automatizadas. Torna-se importante notar que os métodos de avaliação pode ser utilizados em conjunto ou de forma individual em uma ordem dese-

jada [45].

Existem três principais métodos, que auxiliam na avaliação de acessibilidade:

1. Testes de usuário: envolvendo usuários com deficiência;

2. Revisões de especialistas ou peritos: envolvendo opiniões e testes por especialistas na área de acessibilidade;

3. Avaliações automatizadas: envolvendo a utilização de ferramentas de *software* automatizadas;

Cada um desses métodos supracitados serão descritos brevemente nas próximas subseções.

2.5.1 Testes de Usuários

Os usuários, como utilizadores do sistema ou produto, podem fornecer informações relevantes, permitindo identificar falhas na sua interação com o sistema ou produto. No contexto de usuários com deficiência e dificuldades, estes podem identificar a lacuna existente entre a acessibilidade técnica do sistema ou do dispositivo com a realidade de uso e interação, auxiliando assim a correção destas falhas e contribuindo para obtenção da qualidade [45].

Testes de usuários envolvem oferecer aos usuários, ou àqueles que representam múltiplos usuários, uma lista de tarefas a serem executadas no sistema ou produto sob avaliação sem assistência de outras pessoas. Os observadores, especialistas em acessibilidade e *designers*, tentam identificar onde os usuários enfrentam dificuldades. O desempenho da tarefa, ou quantidade de tarefas concluídas com êxito, e desempenho de tempo, ou tempo decorrido para executar as tarefas requeridas, são registradas como as duas métricas mais comuns utilizadas nos testes de usuários [45].

2.5.2 Revisões de Especialistas

Revisões de especialistas envolvem realizar diversas avaliações, oferecer opiniões e testar o produto ou serviço, com o objetivo verificar a conformidade técnica com os requisitos legais [45]. Dentre as principais inspeções estão:

- Avaliação Heurística: Utilização de um conjunto de heurísticas, ou métodos não empíricos, aplicados às interfaces com a finalidade de encontrar problemas relacionados à acessibilidade. Entretanto, tal abordagem abrange apenas problemas mais recorrentes.

- Avaliação de Diretrizes: Utilização de uma inspeção abrangente, aborda a plena conformidade com os requisitos de acessibilidade e diretrizes legais, abrangendo vários tipos de deficiências. Entretanto, esse tipo de avaliação é onerosa em relação ao tempo.

- Avaliação de Consistência: Utilização de uma inspeção envolvendo especialistas que visa garantir a consistência de um conjunto de fatores, referentes às interfaces, cores, linguagem, formatos de entrada e saídas, entre outros. Sua utilização é comum, entretanto, não atendem de forma específica a acessibilidade.

Os especialistas inspecionam interfaces de sistemas ou produtos a fim de encontrar falhas técnicas, auxiliando no cumprimento rigoroso dos requisitos legais e fornecendo orientações técnicas relevantes. Entretanto, pelo fato dos especialistas não serem usuários representativos, estes podem possuir poucos conhecimentos no domínio.

2.5.3 Revisões Automatizadas

A revisão automatizada utiliza o viés da utilização de ferramentas de *software*, os quais verificam o desenho de tela, geralmente de páginas Web, com o objetivo de observar o cumprimento de um conjunto de orientações técnicas. Sua principal vantagem, frente a outras abordagens, é que esta pode rapidamente e repetidamente verificar acessibilidade em um grande número de páginas Web [45].

Apesar das vantagens trazidas pela abordagem automatizada, na avaliação de acessibilidade, há uma série de preocupações decorrentes da dependência exclusiva destas ferramentas de *software*. Devido às ferramentas automatizadas não serem perfeitas, estas podem fornecer resultados confusos e equivocados. Por conta disso, a WCAG 2.0 preconiza, ou recomenda, a divisão de itens que podem ser testados com segurança e confiança em ferramentas automáticas, com aqueles que podem ser testados de forma confiável por seres humanos. Por exemplo, as ferramentas automatizadas podem avaliar a presença ou não de componentes, tais como a utilização de cores, entretanto, a ferramenta automatizada não é capaz de avaliar com eficácia a semântica de conteúdos, por exemplo, se o contraste entre o texto e a cor de fundo é suficiente à compreensão das informações [45].

Para os desenvolvedores, ferramentas que auxiliam a avaliação são instrumentos importantes que os ajudam encontrar falhas de acessibilidade, fornecendo avaliações mais rápidas, se comparado à avaliações manuais. Além disso, ferramentas de apoio à acessibilidade têm boa aceitação pelos desenvolvedores de conteúdo, auxiliando-os em tarefas repetitivas e ajudando os

desenvolvedores iniciantes, que geralmente possuem baixo nível de conhecimento técnico [45].

De maneira geral, revisões automáticas, tem como objetivo principal, auxiliar desenvolvedores na identificação de problemas relacionados a aplicação dos pontos de verificação das diretrizes de acessibilidade, sejam elas a WCAG 2.0, eMAG [34], entre outras. As ferramentas de verificação analisam o código fonte do conteúdo e indicam onde existem falhas, que prejudicam a acessibilidade do conteúdo [6].

A Figura 2.1 ilustra um exemplo de análise automática de conformidade utilizando a ferramenta *WAVE* (http://wave.webaim.org/) sobre o sítio da prefeitura de Campo Grande (http://www.pmcg.ms.gov.br/). Como apresentado na imagem, a ferramenta *WAVE* aponta erros relacionados a falta de textos alternativos em 40 elementos, 15 erros referentes a falta de rótulos em imagens de *link*, além da não definição da linguagem da página, entre outros erros que afetam na acessibilidade do sítio Web.

Figura 2.1: Exemplo de utilização da ferramenta de avaliação de acessibilidade *WAVE*

Considerando a diversidade de ferramentas que realiza a avaliação automática de conformidade à diretrizes de acessibilidade, a Tabela 2.2 apresenta uma sequência de ferramentas relevantes neste âmbito, além de informações referentes às suas características. Elas foram enumeradas a partir da listagem oferecida pelo *Web Accessibility Evaluation Tools List (WAI)*. A coluna descrita como “Ferramenta”, refere-se ao nome e ao endereço da ferramenta. A coluna “Fornecedor”, informa seu responsável. A coluna “Plataforma” apresenta se a ferramenta é executada sobre um servidor Web, plataforma *Desktop* e/ou se estende sobre um navegador Web. A coluna “Linguagem” apresenta, se dis-

ponível, a linguagem de programação em que foi desenvolvida a ferramenta. Quanto à licença, os softwares que disponham de livre acesso ao código fonte, foram classificados como ferramentas de "Código Aberto". Ferramentas que tenham sua utilização livre mas sem acesso ao código, foram classificadas como "Softwares Livres" e, por fim, as ferramentas pagas, foram classificadas como "Comercial". Além disso é possível que existam ferramentas mescladas, com algumas funcionalidades pagas e outras gratuitas. Na coluna subsequente, a "Conformidade", diz respeito a quais diretrizes a ferramenta aborda. Por fim, a tabela apresenta a coluna "Formatos Suportados" indicando quais formatos de conteúdo a ferramenta abrange para realização da avaliação automática.

Tabela 2.2: Exemplos de ferramentas que auxiliam na avaliação automática de Web Sites, no âmbito de acessibilidade [43].

Ferramenta	**Fornecedor**	**Plataforma**	**Linguagem**	**Licença**	**Conformidade**	**Formatos Suportados**
ASES http://www.governoeletronico.gov.br	Programa de Governo Eletrônico Brasileiro	Desktop: Windows / Linux	Java	Software Livre	WCAG e eMAG	CSS,HTML e XHTML
A-Tester http://www.evaluera.co.uk/	Evaluera Ltda	Servidor / Extensão Para: Google Chrome	Não informada	Software Livre	WCAG	HTML
A11Y Compliance Platform http://www.boia.org/?wc3	Bureau of Internet Accessibility	Servidor	Não Informada	Comercial	WCAG, Section 508, French government standard	CSS, HTML, XHTML, SVG, PDF, Imagens e SMIL
Accessibility Developer Tools https://chrome.google.com/ webstore/ detail/accessibility-developer - t/fpkknkljclfencb dbgkenhalefipecmb?hl=en	Google Accessibility	Extensão para: Google Chrome	JavaScript	Software Livre	WCAG	CSS, HTML
Accessibility Management Platform (AMP) http://amp.ssbbartgroup.com/	SSB BART Group	Servidor / Extensão para: Firefox e Internet Explorer	Não Informada	Comercial	WCAG, Section 508, Japanese industry standard, Irish National IT Accessibility Guidelines	CSS, HTML, XHTML, PDF, XML, Javascript, AJAX, ARIA
AccessIn http://www.accessin.org/	Digital Accessibility Centre	Servidor	Não Informada	Comercial	WCAG	CSS, HTML, XHTML, SVG,PDF, Imagens e SMIL
AccessLint http://accesslint.com/	Cameron Cundiff	Servidor / Desktop	Ruby	Software Livre	WCAG	CSS, HTML
AccessMonitor http://www.acessibilidade.gov.pt/ accessmonitor/	Fundação para a Ciência e a Tecnologia, IP	Servidor	PHP	Software Livre	WCAG	CSS, HTML e XHTML
AChecker http://achecker.ca	Inclusive Design Research Centre	Servidor	PHP	Software Livre / Código Aberto	WCAG, Section 508, Stanca Act e BITV	CSS, HTML e XHTML
COMPLYFirst Professional http://www.odellus.com/	Odellus Corporation	Desktop	Não Informada	Comercial	WCAG e Section 508	CSS, HTML, XHTML, PDF e Imagens
DaSilva http://www.dasilva.org.br/	Acessibilidade Brasil	Servidor	Java	Software Livre	WCAG	CSS, HTML e XHTML
Examinator http://examinator.ws/	Carlos Benavidez	Servidor	PHP	Software Livre	WCAG	CSS, HTML e XHTML
HERA http://www.sidar.org/hera/	Fundación Sidar	Servidor	PHP	Software Livre / Código Aberto	WCAG	CSS e HTML
HiSoftware Compliance Sheriff Web http://www.hisoftware.com	HiSoftware Inc.	Servidor / Extensão para: Firefox e Internet Explorer	Não Informada	Comercial	WCAG, Section 508, RGAA	Documentos Microsoft Office, PDF , CSS, HTML, XHTML e Imagens
MAUVE http://hiis.isti.cnr.it:8080/ MauveWeb/	Human Interfaces in Information Systems Laboratory - ISTI-CNR	Servidor	Java	Software Livre	WCAG, Stanca Act	CSS, HTML e XHTML
Total Validator http://www.totalvalidator.com/	Total Validator	Desktop / Extensão para: Google Chrome e Firefox	Java	Software Livre / Comercial	WCAG, Section 508	HTML e XHTML
Vamolà http://www.validatore.it/	Regione Emilia - Romagna	Servidor	PHP	Software Livre / Código Livre	WCAG e Stanca Act	CSS, HTML e XHTML
WAVE http://wave.webaim.org/	WebAIM	Servidor / Extensão para: Google Chrome e Firefox	Não Informada	Software Livre / Comercial	WCAG e Section 508	CSS, HTML, XHTML e Imagem

2.6 Trabalhos Relacionados

Nesta seção são elencados os principais trabalhos no âmbito da acessibilidade que mostram o impacto dos ambientes de desenvolvimento e da evolução da tecnologia na melhoria da acessibilidade web.

O trabalho de *Richards* [57] investigou as tendências da acessibilidade web ao longo dos últimos 14 anos e analisou a relação entre o desenvolvimento de novas tecnologias e recursos desenvolvidos com o aumento da acessibilidade. Uma de suas conclusões é que ferramentas de apoio ao desenvolvimento de conteúdo acessível contribui diretamente na melhoria da acessibilidade, no entanto também alerta que a análise de acessibilidade exige um controle humano, sendo necessário combinar verificadores automáticos de acessibilidade automatizados com a verificação humana [57] [40].

Outro estudo relevante é o do *Fernandes* [36] que analisou o impacto das tecnologias de desenvolvimento na acessibilidade do conteúdo desenvolvido. Apesar da amostra utilizada, este trabalho apresenta que a tecnologia usada pelos desenvolvedores têm influência direta na acessibilidade do conteúdo criados. Segundo o mesmo, os conteúdos desenvolvidos nas linguagens *PHP* e *Rubi* alcançavam maiores níveis de acessibilidade [36].

Outro estudo relevante foi conduzido por *Freire* [37] que analisou grupos específicos com deficiência e identificou as características principais dos problemas de acessibilidade. Como conclusão o autor verificou que os sites possuem variados níveis de barreira, que dependem da deficiência que o usuário possui e que não há correlação significativa entre as classificações das gravidades de problema de usuários e os níveis de prioridades associadas às diretrizes e critérios de sucesso do *WCAG* [37].

Como pode ser percebido nos estudos relacionados, a acessibilidade é influenciada por diversas variáveis, tais como conhecimento técnico dos desenvolvedores, diretriz de acessibilidade e tecnologia de desenvolvimento utilizada [67]. Além disso, diretrizes como a WCAG não fornece todas as orientações para desenvolver uma solução de acessibilidade universal que atenda qualquer usuário deficiência [64][37].

2.7 Considerações Finais

Este capítulo apresentou a legislação, documentos e tecnologias de apoio a acessibilidade Web. Neste trabalho, serão investigadas, integradas e desenvolvidas ferramentas que promovam o desenvolvimento de produtos em conformidade com normas e padrões descritos nesta seção, mais especificadamente com respeito as diretrizes *WCAG*. Também serão utilizados avaliadores au-

tomáticos de acessibilidade de licença livre. O próximo capítulo apresenta a pesquisa bibliográfica referente a integração de ferramentas de software.

CAPÍTULO

3

Integração De Ferramentas CASE

3.1 Considerações Iniciais

Frequentemente o ambiente de desenvolvimento é composto por várias ferramentas de desenvolvimento especializadas a fim de facilitar e agilizar o processo de desenvolvimento de software. Com objetivo de incrementar o ambiente, torna-se relevante integrar outras ferramentas, que juntas proporcionarão uma solução mais ampla.

Neste capítulo são apresentados os principais conceitos relacionados à integração de ferramentas de software. Além disso, são elencados os modelos e os tipos de testes de integração, os quais serão úteis para garantir que cada uma das ferramentas CASE mantenham suas funcionalidades sem erros.

3.2 Ferramentas CASE

Computer-Aided Software Engineering (CASE) é o domínio de ferramentas de software que auxilia as atividades de Engenharia de Software, utilizadas para projetar, implementar e testar aplicativos. Em geral, uma ferramenta CASE engloba um conjunto de outras ferramentas automatizadas e integradas, que apoiam o processo de desenvolvimento de software em um escopo mais amplo. As ferramentas CASE podem ser classificadas sob diversos aspectos, de forma geral, podendo ser classificadas em três categorias [25] [60]:

- Ferramenta de Apoio: Apoiam somente tarefas específicas no processo de produção de software.

- *Workbenches* ou plataforma: Apoiam uma ou algumas atividades do processo de desenvolvimento de software, para isso, integram diversas ferramentas em uma única aplicação.
- Ambientes de Integração: Englobam todo, ou grande parte, do processo de desenvolvimento de software, integrando um conjunto de ferramentas e *Workbenches*.

As ferramentas CASE apresentam vantagens em relação a outros tipos de ferramentas e recursos. Como o desenvolvimento na ferramenta CASE é centralizado, a sua utilização pode reduzir gastos, diminuir tempo de desenvolvimento e melhorar a qualidade do software desenvolvido [33]. Entretanto, por haver diversos recursos integrados, a complexidade da ferramenta CASE pode ser superior a de outros tipos de ferramentas, tornando-se importante, desenvolver ferramentas CASE com facilidade de uso.

3.3 Integração de Ferramentas

A necessidade de desenvolver produtos de software fez com que atualmente exista uma grande quantidade de soluções, nos mais variados domínios. A procura pelo incremento de produtividade e qualidade levou à criação de soluções esparsas que atendem à necessidade de indivíduos e empresas separadamente. Entretanto, pouca atenção é dada à integração de ferramentas existentes.

A integração de ferramentas tem sido um desafio presente desde os anos 80, permeando o desenvolvimento de software. Segundo especialistas, esta lacuna talvez esteja presente atualmente por causa de fatores externos e internos, tais como pela falta de padronização de interfaces e pela falta de interesse por parte dos consumidores e desenvolvedores [44].

A integração de ferramentas de software pode ser definida em alto nível como uma suíte de ferramentas, combinadas entre si com o objetivo de fornecer uma solução mais robusta e abrangente. Essas ferramentas devem realizar de forma sincronizada o intercâmbio de informações, dados, artefatos e serviços, abarcando assim, uma ou mais etapas do desenvolvimento de software [60].

A utilização de ferramentas integradas outorga diversos benefícios no processo de desenvolvimento de software, tais como maior controle do projeto, redução de esforços e atividades e diminuição de erros, contribuindo assim na qualidade do produto final [44].

Por se tratar da união de ferramentas para solucionar ou auxiliar um processo do desenvolvimento, a integração de software exige que essas ferramentas sejam de alguma forma compatíveis para que, combinadas, promovam o

êxito da solução. Diante disso, a integração dessas ferramentas pode ser realizada sob distintos tipos de integração, aplicadas a ferramentas contidas em *workbenches* ou em um ambiente de integração [19]:

- Integração de Plataforma;
- Integração de Dados;
- Integração de Apresentação;
- Integração de Controle; e
- Integração de Processo.

Os tipos de integração supracitados são descritos em mais detalhes nas seções a seguir.

3.3.1 Integração de Plataforma

A integração de plataforma é baseada na cooperação e execução orquestrada de processos distribuídos, executados sobre plataformas de *Hardware/ Software* heterogêneas, através da comunicação utilizando Web *Services* [26].

A Figura 3.1 apresenta um exemplo de uma arquitetura de integração com quatro ferramentas localizadas em diferentes plataformas. Em geral, a arquitetura contém três classes de elementos. A aplicação central de serviços, que provê a comunicação e integração entre os serviços, além de fornecer as descrições de interfaces necessárias para a troca de informações entre as aplicações e plataformas. A interface com o núcleo que fornece uma interface de transformação padronizada para a comunicação das ferramentas com a aplicação central de serviços. Por fim, as ferramentas implementam as funções não fornecidas pela aplicação central de serviços [26].

3.3.2 Integração de Dados

A capacidade de compartilhar informações e dados entre ferramentas é fundamental [27]. Os dados oriundos de uma ferramenta devem ser compreendidos por outras ferramentas, que deles fazem uso [35]. No âmbito de integração de dados, a integração pode estar contida entre distintos níveis de integração:

- Arquivos Compartilhados: Cada ferramenta foi projetada para reconhecer determinados formatos de arquivos. A integração neste tipo de abordagem força as ferramentas a conhecerem a estrutura lógica do arquivo.

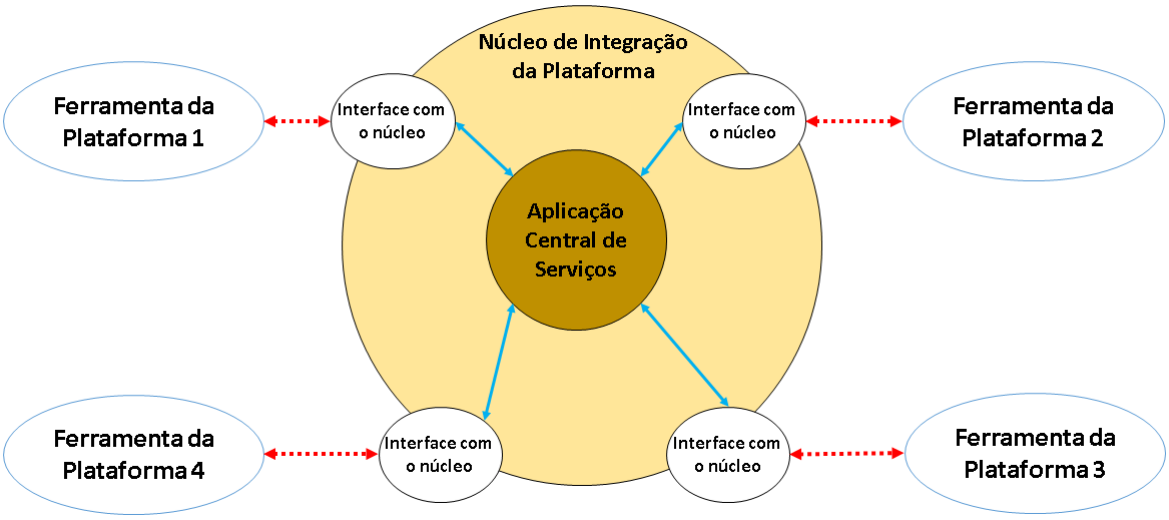

Figura 3.1: Visão geral de uma arquitetura de integração de plataforma. Adaptado de *Bangemann* [26].

A implementação deste comportamento pode representar custos consideráveis, visto que cada ferramenta dispõe de formatos singulares, sendo necessário a implementação de filtros e tradutores de formatos.

- Estrutura de dados e variáveis compartilhadas: As ferramentas utilizam estrutura de dados e variáveis que contribuem no correto funcionamento do Software. Essas estruturas e variáveis podem ser reconhecidas por outras ferramentas que utilizam da mesma linguagem de programação.

- Repositório Compartilhado: As ferramentas podem ser integradas por meio de um sistema de gerenciamento de objetos. Este modelo de integração é acessível por todas as ferramentas. A utilização de repositório compartilhado é a forma mais flexível de integração de dados, entretanto as ferramentas devem acessar constantemente o repositório, o que gera perda de desempenho. Outro fator negativo é que deve-se implementar um sistema de gerenciamento de objeto, incrementando-se um custo adicional.

Diante das demandas de produtos de software, a integração de ferramentas surge como uma ferramenta interessante, possibilitando aos desenvolvedores, nas etapas de construção de software, o provimento de soluções mais completas e robustas. Fatores internos e externos, tais como a padronização de dados e compatibilidade entre as ferramentas, devem ser levadas em consideração, visto que influenciam diretamente na simplificação e eficiência da integração.

3.3.3 Integração de Apresentação

Os serviços de interface de usuário permitem que as ferramentas CASE interajam constantemente com o usuário, possibilitando a confecção de novas ferramentas de maneira simples [54]. A integração de apresentação é o grau em que as ferramentas, em diferentes contextos, podem interagir com o usuário, através de um estilo comum e um conjunto de padrões com o objetivo de reduzir as dificuldades de utilização da ferramenta CASE [35]. A integração de apresentação, de maneira geral, pode ser separada em três níveis:

- Sistema de Janela: As ferramentas integradas devem apresentar o mesmo sistema de interface gráfica (janelas), possibilitando assim, uniformidade entre as janelas.

- Comandos: As ferramentas integradas devem possuir os mesmos atributos de comandos e comportamentos.

- Interação: Aplicado sobre sistemas em que o usuário interage diretamente, as ferramentas integradas devem possuir um padrão de interação comum sobre objetos gráficos e textuais.

3.3.4 Integração de Controle

O mecanismo de integração de controle, enfatiza uma abordagem de comunicação comum entre ferramentas e um entendimento de eventos compartilhados [27]. Essa abordagem é baseada no compartilhamento de informações de controle, através da oferta de serviços por diferentes ferramentas, são enviados sinais de controle que auxiliam a tomada de decisão e mudança de fluxo destas ferramentas [35]. Para alcançar a integração de controle, as ferramentas recorrem a serviços de mensagens para fornecer três tipos de comunicação: Ferramenta-a-Ferramenta, Ferramenta-a-Serviço e Serviço-a-Serviço [54].

Ativar, desativar e utilizar serviços de outras ferramentas são exemplos de integração de controle, concedidos através de interfaces públicas de interação. Outro exemplo pode ser a troca de mensagens entre ferramentas e servidores [35]. A Figura 3.2 apresenta um exemplo do mecanismo de integração de controle por meio da troca de mensagens entre as ferramentas [27]. Neste exemplo, a Ferramenta 1 requisita da Ferramenta 3 uma informação, após receber a requisição, a Ferramenta 3 envia a informação para a ferramenta 1. Por fim, quando a Ferramenta 1 recebe a informação, notifica as Ferramentas 3 e 4, por exemplo, de algo relacionado a informação.

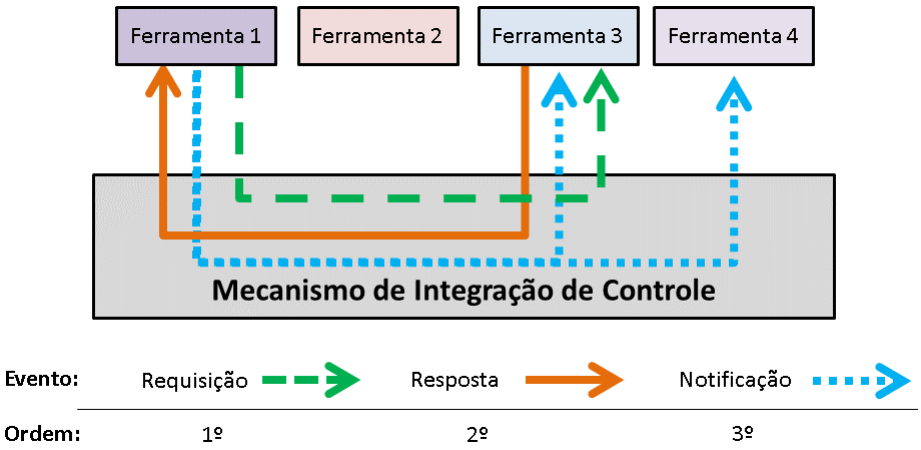

Figura 3.2: Exemplo de um mecanismo de integração de controle, através da troca de mensagem. (Adaptado de [27])

3.3.5 Integração de Processos

Com o objetivo de apoiar várias fases do processo de desenvolvimento, torna-se interessante integrar ferramentas que oferecem apoio a essas fases. A integração de processos pode ser auxiliada através de outros tipos de integração, tais como integração de dados, apresentação e controle, com a finalidade de abranger as fases do processo de desenvolvimento. A Figura 3.3 apresenta a integração das ferramentas CASE visando atender uma ou mais fases do processo de desenvolvimento de software, alçadas sob um ambiente de desenvolvimento. Dessa forma, é possível utilizar ferramentas de apoio ao desenvolvedor em uma ou mais fases do processo de desenvolvimento de software.

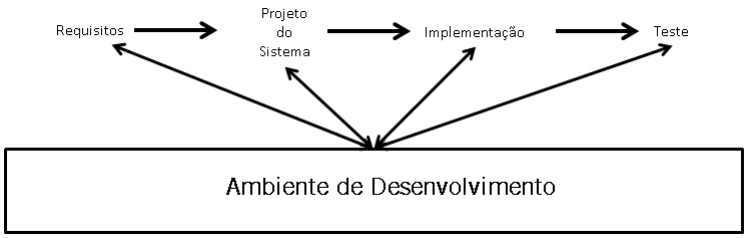

Figura 3.3: Integração dos processos de desenvolvimento. (Adaptado de [27])

Uma prática comum adotada é a utilização de *frameworks* com o objetivo de simplificar um processo e fornecer um nível maior de abstração [23].

3.4 Testes de Integração

Um sistema é composto por múltiplos componentes, ou módulos, que unidos compreendem a funcionalidade de um software ou *hardware*. A integração

é definida como um conjunto de interações entre esses componentes. Uma abordagem a integração corresponde a codificação de cada um destes artefatos de código separadamente, a união de todos os artefatos e por fim, o teste do sistema como um todo [44].

Enquanto o teste de unidade isola o módulo de suas dependências e o testa separadamente, o teste de integração tem o objetivo de verificar se após a integração dos módulos o sistema funciona como o esperado [22]. A Figura 3.4 apresenta a integração de dois módulos, módulo A e módulo B. Na interseção dos módulos encontra-se o teste de integração, que auxilia na busca e na correção de problemas, relacionados à interface entre os módulos ou componentes.

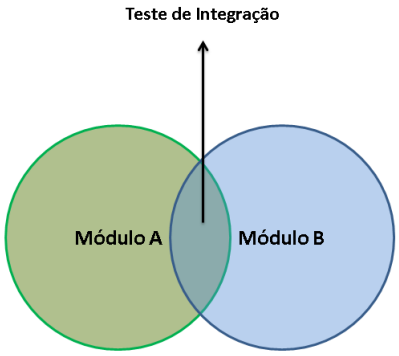

Figura 3.4: Exemplo de integração de dois módulos, módulo A e módulo B.

Os testes de integração de sistemas podem seguir várias abordagens, as mais comuns são: incremental, *Top–Down*, *Bottom-Up*, *Sandwich* e *Big Bang* [20].

No decorrer desta seção tais abordagens serão explanadas com maiores detalhes.

3.4.1 Incremental

Na abordagem incremental, o sistema é construído gradualmente. Em cada ciclo, mais módulos são integrados e testados para gerar um módulo maior. Por fim, o sistema é construído de forma incremental, ciclo após ciclo, até que todo o sistema esteja operável e pronto para receber os testes em nível de sistema.

O sistema é construído com uma sucessão de camadas, iniciando com a integração dos módulos principais, ou núcleo. Em cada ciclo, uma nova camada é adicionada ao módulo principal e testada, tornando-se este o novo módulo principal. O número de ciclos e a complexidade da integração se relacionam a diversos parâmetros, tais como o número de módulos e a interdependência

entre eles [20].

3.4.2 Top-Down

Sistemas com estruturas hierárquicas podem ser integrados e testados através das abordagens *Top-Down* e *Bottom-Up*. Em um sistema hierárquico, há um módulo de alto nível, que é decomposto em alguns módulos de segundo nível. Os módulos de segundo nível podem ser ainda decompostos em módulos de terceiro nível, e assim sucessivamente. Módulos terminais são aqueles, que em quaisquer níveis, não são mais decompostos. Módulos internos, ou não terminais, executam operações e invoca seus módulos subordinados, que retornam o controle e resultados ao módulo chamador.

Testes de integração na abordagem *Top-Down*, são realizados de cima para baixo, como apresentado na Figura 3.5, seguindo o fluxo da arquitetura desenvolvida, por exemplo, começando o teste na interface gráfica do utilizador (GUI) ou do menu principal [44].

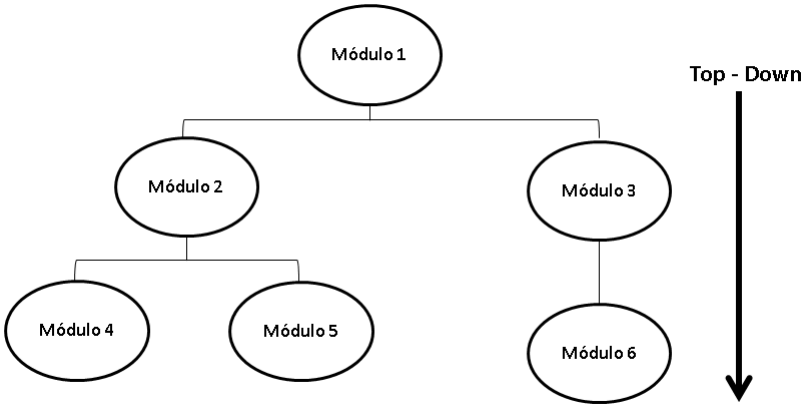

Figura 3.5: Exemplo de diagrama utilizando a abordagem *Top−Down*. (Adaptado de [27])

Embora a integração e os testes dos módulos ocorram de cima para baixo, é possível que em alguma etapa, algum módulo obrigatório esteja ausente. Em tal caso, é utilizado um programa denominado *Stub*, que retrata um esboço do módulo necessário [44]. As seguintes vantagens podem ser citadas para a abordagem *Top-Down*: o teste realizado é vantajoso pois é possível capturar as principais falhas do sistema. Outra vantagem é que os *Stubs*, ou esboços, podem ser construídos em menor tempo, se comparados à construção do módulo requerido, além de serem mais simples.

No entanto, existe a desvantagem da funcionalidade básica ser testada apenas ao final do ciclo.

3.4.3 Bottom-Up

Na abordagem *Bottom-Up* a integração e os testes dos módulos do sistema se iniciam com aqueles módulos que encontram-se no nível inferior. Um módulo é dito estar no nível inferior se ele não invoca outros módulos, ou seja, se são módulos finais [44].

A abordagem envolve o desenvolvimento de todos os módulos filhos, antes da devida integração com os módulos pais correspondentes [44]. Essa abordagem assume que todos os módulos foram individualmente testados anteriormente. A Figura 3.6 apresenta um exemplo de diagrama utilizando a abordagem *Bottom Up*.

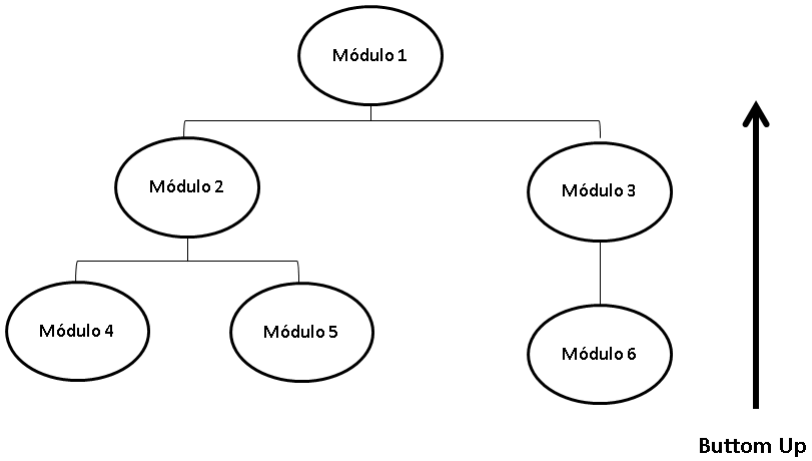

Figura 3.6: Exemplo de diagrama utilizando a abordagem *Bottom Up*. (Adaptado de [27])

Para integrar e testar um conjunto de módulos nesta abordagem, é necessário construir esboços de módulos, denominados módulos controladores de testes, que serão responsáveis por invocar os módulos que serão integrados. Quando a integração e testes de um grupo desejado de módulos de nível inferior são realizados e considerados satisfatórios, o módulo controlador é substituído pelo módulo real integrado, e um outro módulo controlador de teste é usado para integrar e testar os módulos já integrados com os ainda não integrados, e assim sucessivamente.

3.4.4 Big Bang e Sandwich

Na abordagem de teste de integração *Big Bang*, após cada módulo ter sido testado individualmente, todos os módulos do sistema são integrados conjuntamente, para então o sistema ser testado como um todo [58]. Como na abordagem *Big Bang* todos os componentes ou módulos são integrados e testados simultaneamente, por vezes os desenvolvedores utilizam esse tipo de

integração em sistemas pequenos.

A utilização da abordagem *Big Bang* não é recomendada para sistemas grandes devido a complexidade e pela grande quantidade de módulos e interfaces, podendo haver problemas em determinar sobre quais interfaces um certo problema ocorreu [53]. A Figura 3.7 apresenta um exemplo de diagrama utilizando a abordagem *Big Bang*.

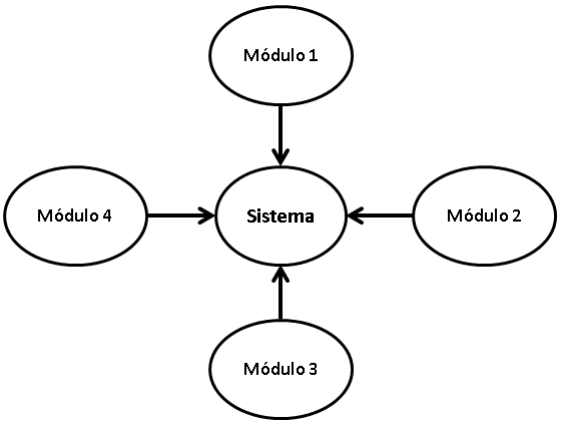

Figura 3.7: Exemplo de diagrama utilizando a abordagem *Big Bang*. Adaptado de [58]

A seguinte vantagem pode ser citada em relação a abordagem *Big Bang*: pelo fato de cada módulo não depender exclusivamente de outro, a integração torna-se rápida.

No entanto, a desvantagem desta abordagem é a dificuldade em encontrar e detectar a causa de falhas, pois o sistema testado encontra-se com todos os módulos já desenvolvidos.

Para suprir as limitações das abordagens existentes, é realizada a combinação de ambas abordagens, também conhecida como abordagem *Sandwich* ou híbrida [53]. Nesta abordagem, é possível utilizar a combinação entre as abordagens *Top-Down* e *Bottom-Up* [58].

A decisão final sobre a seleção de uma abordagem de integração e testes é diretamente relacionada a característica do sistema, bem como das expectativas dos clientes e requisitos do projeto. Estudos empíricos avaliaram a eficácia das abordagens *Top-Down*, *Bottom-Up*, *Sandwich* e *Big Bang* como estratégias na integração de sistemas de software. Os estudos indicaram que a abordagem *Top-Down* é mais efetiva em termos de correção de defeitos. As estratégias *Top-Down* e *Big Bang* indicaram produzir sistemas mais confiáveis. A abordagem *Bottom-Up* é geralmente menos eficaz na correção de problemas e produzem sistemas com maior risco de defeitos. Entretanto, sistemas integrados através da estratégia *Sandwich*, que utiliza as técnicas *Top-Down* e *Bottom-Up*, podem resultar em sistemas confiáveis [58].

3.5 Considerações Finais

Neste capítulo foi apresentada uma breve revisão bibliográfica sobre integração de ferramentas de software.

Os conhecimentos apresentados serão utilizados no desenvolvimento e na integração das ferramentas CASE de acessibilidade propostas para este trabalho. Pretende-se utilizar as técnicas de integração de plataforma, integração de dados, integração de apresentação, integração de controle e integração de processos. O teste de integração escolhido para este trabalho foi o incremental, pois, as ferramentas serão construídas e integradas gradativamente, ciclo a ciclo, até que ao final obtenha-se um sistema operável e integrado. Esta escolha foi realizada tomando como base a aplicabilidade e o conhecimento do autor em relação à abordagem. Os detalhes destas escolhas serão explanados no próximo capítulo.

Além disso, no próximo capítulo serão apresentadas as etapas de planejamento, construção e a prova de conceito da ferramenta proposta.

CAPÍTULO

4

Acero: Uma ferramenta CASE para o desenvolvimento de aplicações Web acessíveis

4.1 Considerações Iniciais

A construção de ferramentas de software não é trivial e isso inclui a construção de ferramentas que auxiliam a criação de conteúdos acessíveis. Apesar dos esforços dedicados para suprir as lacunas existentes no contexto de acessibilidade em Web sites, ainda há muito a ser feito [62].

Durante a investigação de estudos relacionados à acessibilidade foi notado que integrar ferramentas de software que já haviam sido desenvolvidas pelo grupo de Engenharia de Software da *Facom – UFMS* no contexto de acessibilidade poderia contribuir no desenvolvimento de conteúdo Web acessível. Além disso, utilizar ambientes de desenvolvimento é uma prática comum entre os desenvolvedores, pois, aportam benefícios como produtividade e qualidade das aplicações produzidas, visto que geralmente são integradas em um mesmo ambiente ferramentas especializadas e consolidadas.

Diante do exposto, foi estudada a integração de ferramentas e a criação de uma solução que estivesse no mesmo ambiente de desenvolvimento do usuário e que promovesse apoio na criação de conteúdos acessíveis. Assim, foi valorizada a possibilidade de integrar ferramentas com o objetivo de apoiar a evolução dos requisitos de acessibilidade durante todo o processo de desenvolvimento de Software.

Foram desenvolvidas pelo grupo de pesquisa em acessibilidade da UFMS duas ferramentas e um processo de desenvolvimento que auxiliam na concepção de soluções acessíveis, entretanto, elas atuam separadamente, em contextos e linguagens distintos. Estes trabalhos são apresentados nas próximas sub-seções.

4.2 Trabalhos Relacionados

Os seguintes trabalhos do grupo de pesquisa da UFMS foram importantes para o desenvolvimento deste projeto:

4.2.1 Um processo para o desenvolvimento de aplicações Web Acessíveis

O trabalho desenvolvido por *Maia* [49] propôs um processo para o desenvolvimento de aplicações Web acessíveis, denominado MTA. O processo é baseado na norma *ISO/IEC 12207* [31], que sugere a inserção tarefas de acessibilidade nos subprocessos de desenvolvimento para que, ao final dos processos, o desenvolvedor tenha um produto acessível.

O MTA foi idealizado com o objetivo de guiar o processo de desenvolvimento desde as etapas iniciais do projeto, com o intuito de evitar o retrabalho ocasionado pelas correções de acessibilidade que geralmente ocorrem na fase de testes. De forma resumida, aplicando tarefas de acessibilidade no processo de desenvolvimento, é possível influenciar positivamente na qualidade final do produto.

4.2.2 Acessibilidade nas Fases de Engenharia de Requisitos, Projeto e Codificação de Software: Uma Ferramenta de Apoio

Baseado no trabalho do *Maia* [49], o trabalho de *Branco* [30] teve por objetivo criar uma estrutura que considerasse os requisitos de acessibilidade em todas a fases do processo, isto é, da análise de requisitos até a fase de codificação, com o objetivo de fornecer um produto acessível de qualidade. Diante deste desafio, o autor desenvolveu uma ferramenta na forma de *plugin* para a *IDE Eclipse*, denominado de *AccTrace*, que associa os requisitos, modelos UML e técnicas de implementação de acessibilidade. Dessa forma, é possível realizar o rastreamento dos requisitos de acessibilidade, bem como a geração automática de classes *Java* com os comentários de implementação de acessibilidade.

Para o desenvolvimento das tarefas propostas pela *AccTrace*, a ferramenta integra outras soluções da seguinte forma: os requisitos são especificados

pelo *plugin Requirement Designer* [1], a definição dos artefatos UML ocorrem por meio do *plugin UML Designer* [2] e as classes são geradas automaticamente por meio do *plugin UML to Java Generator* [3].

A Figura 4.1 apresenta o fluxo percorrido pela ferramenta *AccTrace* para gerar o código com as indicações de acessibilidade e oferecer a rastreabilidade dos requisitos. Inicialmente, antes de utilizar propriamente a *AccTrace* são definidos os artefatos UML por meio do *plugin UML Designer*. Em seguida, são definidos os requisitos (incluindo os requisitos de acessibilidade) utilizando o *plugin Requirement Designer*. Nesta etapa também são referenciados quais artefatos UML possuem os requisitos definidos. Por fim, o *plugin AccTrace* está pronto para iniciar seu funcionamento.

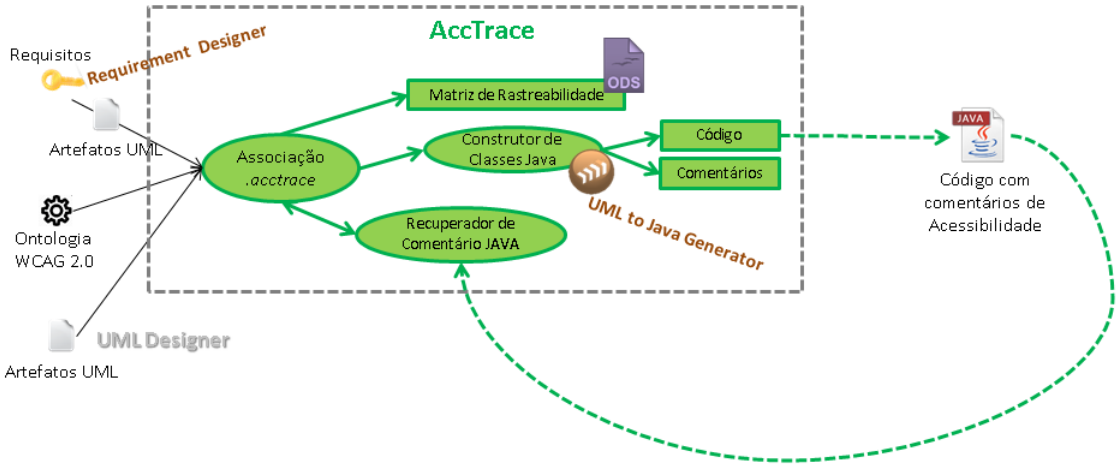

Figura 4.1: Fluxo da ferramenta *Acctrace*. Adaptado de [30]

Após a definição e referenciação dos requisitos e dos artefatos UML, utiliza-se a ferramenta *AccTrace* para realizar a associação com a Ontologia WCAG 2.0. A ontologia é um modelo de dados, representado na *AccTrace* como uma lista de técnicas de implementação, abordagens, critérios de sucesso e testes relacionados às diretrizes de acessibilidade WCAG 2.0. Dessa forma, a *AccTrace* permite criar um arquivo (extensão .actrace) que relaciona a ontologia com os artefatos e requisitos.

A partir do arquivo de associação *.acctrace*, é possível gerar uma matriz de rastreabilidade (extensão *.ods*) que apresenta em forma de três tabelas as relações entre: *Requisitos x Modelos UML*, *Requisitos x Ontologia* e *Modelos UML x Ontologia*.

A *AccTrace* também oferece suporte à criação de classes *Java* com comentários de apoio à acessibilidade de forma automática. A *AccTrace* utiliza como

[1] https://marketplace.eclipse.org/content/requirement-designer
[2] http://www.umldesigner.org/
[3] https://marketplace.eclipse.org/content/uml-java-generator

base o *plugin UML to Java Generator*, o qual toma como entrada o arquivo de associação *.acctrace* e o arquivo de Artefatos UML *.uml*. Geram-se então, classes com comentários específicos que permitem ao usuário, diretamente em seu código, a recuperação das relações pertinentes à classe, auxiliando a implementação de acessibilidade da mesma.

4.2.3 Homero: Um Framework de Apoio ao Desenvolvimento de Interfaces de Aplicações Web Acessíveis

O trabalho desenvolvido por *Oliveira* [32] foi motivado pela necessidade de propiciar acessibilidade aos sistemas desenvolvidos e simplificar a utilização das diretrizes de acessibilidade propostas pelo W3C, com o objetivo de proporcionar interfaces de aplicações acessíveis.

O *Homero* [32] é um *framework* escrito em *PHP* que possibilita e simplifica a utilização das diretrizes de acessibilidade WCAG 2.0, propostas pelo W3C. O *Framework Homero* fornece, por meio de sintaxes definidas em seu projeto, a criação de páginas Web acessíveis no nível AAA da diretriz WCAG 2.0, em relação a critérios que podem ser avaliados automaticamente, indicando avisos de erros que prejudicam a execução do código fonte (*User Errors*) e de erros que prejudicam a acessibilidade da solução (*User Warnings*). O *framework* é composto por classes de elementos *HTML* que, ao serem instanciadas e executadas, fornecem um trecho de código acessível em HTML do respectivo elemento. O *Homero* oferece suporte a distintos tipos de elementos HTML, tais como tabelas, imagens, listas, textos, *links*, entre outros.

4.3 Acero

A solução proposta neste trabalho, denominada Acero, é uma ferramenta que possui o objetivo de integrar todo o arcabouço advindo dos trabalhos anteriores, com o intuito de proporcionar um ambiente de desenvolvimento mais amplo no âmbito do desenvolvimento de produtos acessíveis.

4.3.1 Decisão de Projeto (Design Rationale) do Ambiente de Desenvolvimento

Considerando os conhecimentos e as ferramentas advindas dos trabalhos anteriores, foi identificada a possibilidade de utilização da linguagem *Java*, visto que inclui várias bibliotecas e recursos de comunicação Web. Considerando que a ferramenta *AccTrace* foi desenvolvida na linguagem *Java* e o *framework Homero* foi desenvolvido na linguagem PHP, decidiu-se pela utilização da linguagem *Java* no desenvolvimento da Acero, pois a solução proposta

neste trabalho deveria estar alinhada a um ambiente de desenvolvimento que fosse amplamente utilizado pelos desenvolvedores. Além disso, o ambiente deveria permitir ao desenvolvedor adicionar novos recursos e deveria ser baseado em código aberto.

Segundo o relatório técnico [51], trazido pela *RebelLabs Tools and Technologies Landscape*, em 2016 a *IDE Eclipse* é utilizada por 41% dos desenvolvedores *Java*. A Figura 4.2 apresenta a utilização de *IDEs* que estão em destaque no desenvolvimento de softwares na linguagem *Java*, reforçando a aceitação mundial da *IDE Eclipse*.

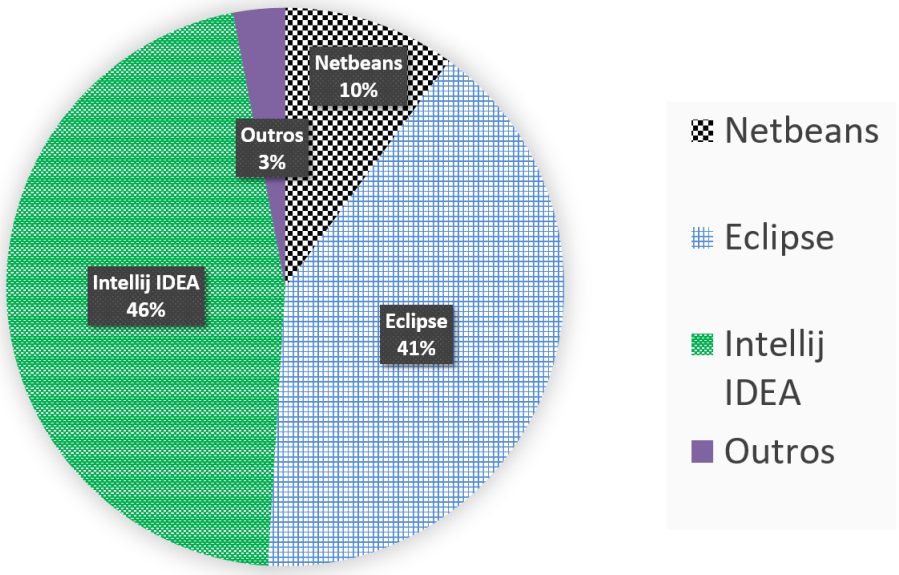

Figura 4.2: *IDEs* mais utilizadas em 2016 no contexto da linguagem Java. *Adaptada de [51]*

A *IDE Eclipse* foi idealizada inicialmente para o desenvolvimento *Java*, entretanto, atualmente oferece suporte a diversas outras linguagens tais como *PHP*, *C/C++* e *Python*. Além disso, a *IDE Eclipse* possui licença pública, o que possibilita ao desenvolvedor criar e instalar *plugins* para enriquecer seu ambiente de desenvolvimento.

Pelos fatores explanados anteriormente, entregar a solução de integração proposta neste trabalho como um *plugin* para a *IDE Eclipse* apresenta diversas vantagens pois será possível usufruir da ferramenta *AccTrace* que favorece a utilização do modelo MTA e suporte à linguagem PHP possibilitando assim a utilização do *framework Homero*. Além disso, permite a comunicação com ferramentas de análise de acessibilidade na Web, tais como a *Achecker*[4] e *Access Monitor*[5], que serão mencionadas posteriormente. Por fim, a *IDE Eclipse* apresenta licença pública, o que permite aos desenvolvedores criarem novas

[4] http://achecker.ca/checker/index.php
[5] http://www.acessibilidade.gov.pt/accessmonitor/

funcionalidades.

4.3.2 Funcionalidades da Ferramenta Acero

A *Acero* é uma ferramenta CASE desenvolvida na forma de um *plugin* para a *IDE Eclipse*, cujo objetivo primário é ofertar ao usuário um ambiente de desenvolvimento mais amplo, robusto e integrado no contexto do desenvolvimento de soluções acessíveis.

A camada de integração da solução proposta neste trabalho é apresentada na Figura 4.3. Destaca-se que foram consideradas as seguintes funcionalidades:

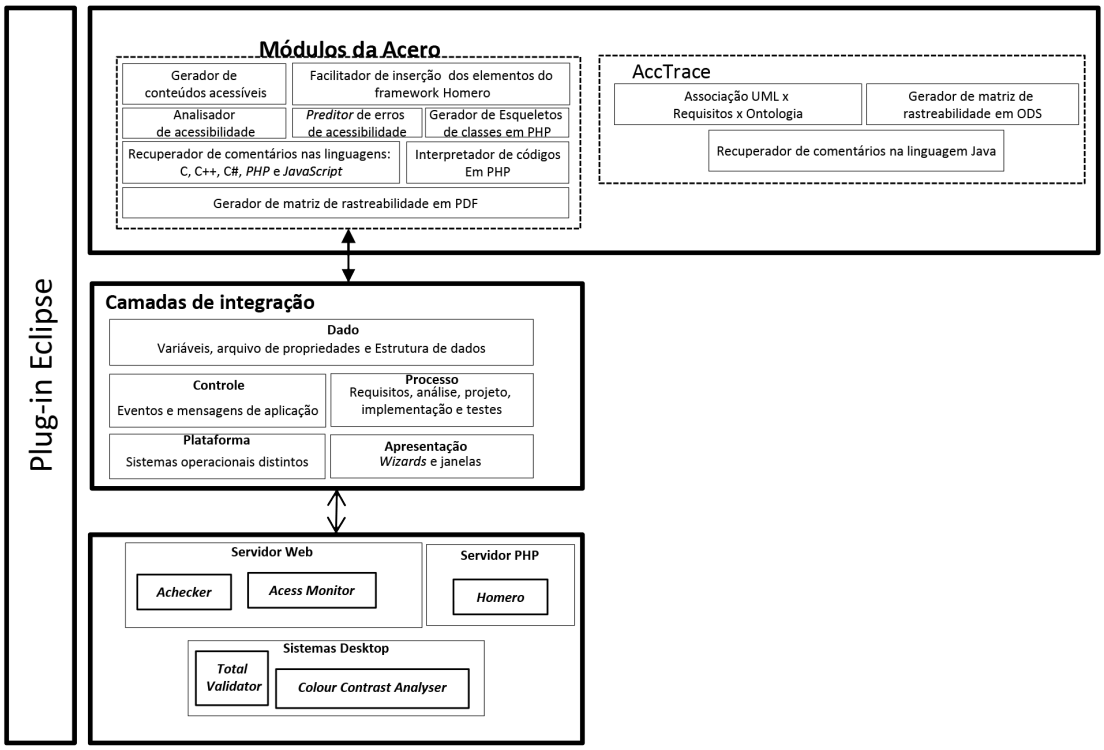

Figura 4.3: Integração entre ferramentas e obtenção da Acero

Facilitar a inserção de elementos do *framework Homero*: Ofertar ao desenvolvedor um nível de utilização mais elevado do *framework Homero*, por exemplo, oferecendo a utilização de *wizards*, para o preenchimento semiautomático dos campos necessários à instanciação dos elementos do *framework Homero*. Este módulo pode contribuir para a produtividade e acessibilidade, pois o usuário não precisaria pesquisar a documentação para conhecer os métodos e seus argumentos. Além disso, deixa o usuário ciente sobre quais campos podem afetar a acessibilidade do conteúdo.

Gerar conteúdos acessíveis: Utilizando a sintaxe determinada pelo *framework Homero*, é possível submeter automaticamente um código ao servidor

onde se encontra o *Homero* e obter como saída um código acessível na extensão HTML. Além disso, é possível verificar quais erros cometidos pelo usuário interferem na acessibilidade do conteúdo, por exemplo, caso o usuário tenha se esquecido de preencher algum campo obrigatório.

Interpretar códigos na linguagem PHP: Disponibilizar ao usuário a possibilidade de executar um código PHP na sintaxe definida pelo *framework Homero*, mesmo que em seu computador não possua um servidor PHP com o *framework* instalado. Tal funcionalidade é útil em ambientes limitados, que não haja um servidor PHP instalado.

Avaliar as diretrizes de acessibilidade dentro do ambiente de desenvolvimento: Oferecer no ambiente de desenvolvimento a possibilidade do usuário submeter seu código-fonte às ferramentas de avaliação automática de diretrizes de acessibilidade, por exemplo, *AChecker* e *Access Monitor* que são ferramentas relevantes no âmbito de avaliação de diretrizes de acessibilidade [48]. Esse módulo poderá contribuir na produtividade, pois, o usuário poderá avaliar diretamente em seu ambiente de desenvolvimento a conformidade de seu código às diretrizes de acessibilidade.

Criar a matriz de rastreabilidade no formato *PDF*: Ofertar a possibilidade do desenvolvedor escolher a extensão *PDF* para gerar a matriz de rastreabilidade.

Realizar a rastreabilidade reversa: Oferecer suporte à verificação das associações do arquivo *.acctrace* de modo automático, para verificar se as associações descritas neste arquivo existem, verificando se os comentários nas classes são coerentes com a associação descrita no arquivo de associação *.acctrace*. Caso exista uma entrada no arquivo de associação e não exista o comentário referente na classe indicada deverá ser possível apresentar a possibilidade de incluir automaticamente o comentário ou removê-lo caso a associação deixe de existir. Dessa forma será possível realizar a rastreabilidade inversa, ou a coerência entre o arquivo de associação *.acctrace* e os arquivos do projeto. Esta funcionalidade será útil quando houver modificações, por exemplo, na exclusões e criação de novas associações e requisitos. Com isso, não será necessñarui ajustar as classes de seu projeto por meio da análise visual da matriz de rastreabilidade visto que as mudanças serão ajustadas automaticamente pela *Acero*.

Construir classes em PHP com os comentários para implementação de acessibilidade: Ofertar ao usuário a possibilidade da criação automática de classes PHP por meio dos arquivos de artefatos UML e do arquivo de associação criado anteriormente pela ferramenta *AccTrace*. Esse módulo poderá contribuir na produtividade, permitindo a criação automática de esqueletos de classes PHP com os comentários para implementação de acessibilidade, con-

forme as associações descritas no arquivo *.acctrace* da ferramenta *AccTrace*.

Recuperar os comentários de acessibilidade em códigos PHP: Fornecer a possibilidade de recuperar a associação definida na *AccTrace*, dos artefatos UML, requisitos e ontologia de acessibilidade em códigos PHP. Essa função é importante pois a ferramenta *AccTrace* suporta somente códigos *Java*. Com este novo recurso será possível recuperar comentários em códigos que iniciam com barras duplas (//), englobando dessa forma, as linguagens: *PHP, C, C++, C#, PHP, JavaScript* e *Java*.

Predizer erros de acessibilidade no código: Com as diretrizes de acessibilidade WCAG 2.0 descritas no *framework Homero*, é possível auxiliar o desenvolvedor alertando sobre possíveis erros em seu código. Por exemplo se o usuário inseriu uma imagem, pode-se apresentar os principais erros que afetam a acessibilidade do elemento imagem, para que os evite. Além disso, é possível verificar se os arquivos declarados no código referenciam elementos existentes.

Integrar com outras ferramentas de apoio a acessibilidade: Ofertar ao usuário a possibilidade de utilizar as ferramentas *Colour Contrast Analyser* que permitem fazer a análise de contrastes de uma interface e possibilita simulá-la na perspectiva de usuários com deficiências visuais, tais como cataratas e daltonismo. Já a ferramenta *Total Validator Basic* permite a validação de diretrizes no código fonte em modo *offline*. Ambas as ferramentas são relevantes e consolidadas no âmbito do desenvolvimento de soluções acessíveis, sendo indicadas pelo *W3C* [43]. Além de ofertar as funcionalidades apresentadas anteriormente, ao utilizar estas ferramentas, será possível fornecer ao usuário um ambiente de desenvolvimento ainda mais amplo.

Para comunicação entre a *Acero* e os serviços ofertados alojados em servidores foi utilizado o método de requisição *POST*[6].

Além disso, uma das principais dificuldades em relação à ferramenta *AccTrace* era a sua utilização, pois não houve a preocupação com a facilidade de uso durante o seu desenvolvimento. Assim, foi elaborado um manual do usuário, apresentado no Apêndice A, em que a ferramenta *Acero* é descrita em detalhes.

Na seção 3, referente a integração de ferramentas, foi mencionado que uma solução de integração de ferramentas não é única, ou seja, a integração é dependente de diversas variáveis, tais como, o problema enfrentado, o que já se possui e qual objetivo pretende-se alcançar com a integração.

O suporte necessário para a integração das ferramentas supracitadas foi proporcionado pelo apoio ofertado da *IDE Eclipse*, que permite os desenvolvedores criarem extensões para solucionar ou auxiliar problemas enfrentados

[6]POST é um método de requisição suportado pelo protocolo HTTP, sendo amplamente utilizado na Web [63].

em seus contextos. Neste trabalho foram criadas extensões relacionadas a *wizards* as quais serão descritas nas próximas seções.

A seguir são apresentadas as implementações de cada uma das funcionalidades propostas pela *Acero*:

4.3.3 Wizard para o framework Homero

Para implementação desta funcionalidade foram tomados como base o código fonte e a documentação do *framework Homero*. Para cada elemento do *framework* Homero foi criada uma *wizard* na Acero. *Wizards* ou assistentes são comumente utilizados na *IDE Eclipse* para a criação de novos arquivos, importações e exportações. Em geral, são utilizados onde existem diversos passos para que o usuário realize uma tarefa.

No tocante ao desenvolvimento de extensões para a *IDE Eclipse*, a classe *Wizard* do pacote *org.eclipse.jface.wizard* fornece a funcionalidade de criar assistentes personalizados para o usuário. Esta funcionalidade controla a navegação entre as páginas e fornece a interface básica para o usuário. Uma *Wizard* pode ter uma ou várias páginas do tipo *Wizard Page* que possuem botões, campos de preenchimentos e outros recursos de apoio. As *Wizards Pages* são adicionadas ao *Wizard* controlador através do método *addPage()*.

Com o objetivo de fornecer informações para melhorias futuras na Acero, a Tabela 4.1 apresenta as classes correspondentes entre o *framework Homero* e as *Wizards* da Acero. Na primeira coluna são apresentadas as classes *PHP* dos elementos do *framework Homero*. Na segunda coluna as classes *Java* controladoras das páginas das *Wizards* na *Acero* e, por fim, as páginas das *Wizards* correspondentes. Alguns elementos, como *Table.php*, *Line.php* e *Cell.php* foram agrupadas na *Acero* em um único *Wizard* controlador, o *TableWizard.java* pois estão diretamente relacionados.

Dessa forma, utilizando janelas de *Wizards* é possível fornecer ao usuário uma forma mais precisa e simples de utilizar o *framework Homero*. Além disso, não é permitido avançar na *wizard* enquanto o usuário não preenche todos os campos requeridos, que afetam a acessibilidade do elemento. Esta informação de campos obrigatórios foi obtida analisando o código fonte de cada elemento do *framework* Homero.

Por exemplo, para o elemento *Media*, o *framework Homero* não indica no momento de definição dos parâmetros do construtor quais parâmetros influenciam na acessibilidade, sendo apenas indicados como um aviso em nível de desenvolvedor no momento da chamada do método *generate*, que fornece o código HTML do elemento *Media*. Utilizando a *Acero*, os campos preenchidos nas *Wizards* são convertidos automaticamente em códigos, já com os parâmetros necessários para ofertar a acessibilidade. Além disso, não é necessário

Tabela 4.1: Classes correspondentes entre o *framework* Homero e a Acero

Classes de Elementos na Homero	**Classes *Wizard* na Acero**	**Classes *Wizard Pages* na Acero**
ButtonImage.php	*ButtonImageWizard.java*	*ButtonImagePage.java*
ButtonReset.php	*ButtonResetWizard.java*	*ButtonResetPage.java*
ButtonSubmit.php	*ButtonSubmitWizard.java*	*ButtonSubmitPage.java*
Cite.php	*CiteWizard.java*	*CitePage.java*
ComboBox.php *Option.php*	*ComboBoxWizard.java*	*ComboBoxPage.java* *OptionPage.java*
Div.php	*DivWizard.java*	*DivPage.java*
Emphasis.php	*EmphasisWizard.java*	*EmphasisPage.java*
Group.php	*GroupWizard.java*	*GroupPage.java* *GroupPage2.java*
Image.php	*ImageWizard.java*	*ImagePage.java*
InputCheckbox.php	*InputCheckboxWizard.java*	*InputCheckboxPage.java*
InputFile.php	*InputFileWizard.java*	*InputFilePage.java*
InputPassword.php	*InputPasswordWizard.java*	*InputPasswordPage.java*
InputRadio.php	*InputRadioWizard.java*	*InputRadioPage.java*
InputText.php	*InputTexWizard.java*	*InputTextPage.java*
Link.php	*LinkWizard.java*	*LinkPage.java*
LongExplanatory.php	*LongExplanatoryWizard.java*	*LongExplanatoryPage.java*
Lst.php	*LstWizard.java*	*LstPage1.java* *LstPage2.java*
Media.php	*MediaWizard.java*	*MediaPage.java*
Object.php	*ObjectWizard.java*	*ObjectPage.java* *ObjectPage2.java*
Paragraph.php	*ParagraphWizard.java*	*ParagraphPage.java*
ShortExplanatory.php	*ShortExplanatoryWizard.java*	*ShortExplanatoryPage.java*
Span.php	*SpanWizard.java*	*SpanPage.java*
Strong.php	*StrongWizard.java*	*StrongWizard.java*
Subscript.php	*SubscriptWizard.java*	*SubscriptPage.java*
SuperScript.php	*SuperScriptWizard.java*	*SuperScriptPage.java*
Table.php *Line.php* *Cell.php*	*TableWizard.java*	*TablePage.java* *LinePage.java* *CellDialog.java*
Textarea.php	*InputTextAreaWizard.java*	*InputTextAreaPage.java*
Title.php	*TitleWizard.java*	*TitlePage.java*

que o desenvolvedor conheça a documentação do *Homero*, pois a *Acero* gera código PHP na sintaxe definida pelo *framework Homero*.

Com respeito às técnicas de integração utilizadas neste módulo, foram utilizadas a integração de apresentação e de dados. A integração de apresentação foi aplicada através da padronização das interfaces fornecidas pelas *Wizards* da *Acero*, visando facilitar o preenchimento dos campos requeridos pelos elementos do *Homero*. Todas as *Wizards* possuem o mesmo *layout*, a mesma sequência de preenchimento e de botões de ajuda ao usuário. A integração de dados foi obtida através da padronização dos códigos gerados a partir das saídas das *Wizards* com a sintaxe exigida pelo *Framework Homero*, possibilitando dessa forma, a execução direta do código, sem a necessidade de quaisquer ajustes manuais pelo desenvolvedor.

4.3.4 Gerador de Conteúdo Acessível e Interpretador PHP

O objetivo desta funcionalidade é proporcionar ao desenvolvedor condições de utilizar o *framework Homero* dentro da *IDE Eclipse*, pois, originalmente, o *framework* não foi idealizado para executar requisições de códigos partindo da *IDE Eclipse*. Para implementar estas funcionalidades foram modificadas as classes do *framework Homero* nos seguintes aspectos:

- Alteração em todas as classes do *Framework Homero* para a forma padrão de abertura de código (de $<?...?>$ para $<?php...?>$). Mesmo que seja possível utilizar os dois modos, ativando a diretriz *short_open_tag* no arquivo de configuração do servidor, por padrão a diretriz $<?...?>$ está desabilitada, não executando corretamente um código que contenha esta diretriz, ou seja utilizar $<?php...?>$ diminui problemas, visto que não é necessário modificar as configurações do servidor PHP.

- Criação de um arquivo de saída específico para erros de acessibilidade encontrados no código do desenvolvedor. Anteriormente os erros eram gerados como mensagens, através da função *trigger_error*. Salvando os erros encontrados em um arquivo de saída próprio, é possível desacoplar o código gerado dos erros presentes no código do desenvolvedor.

- Criação de uma classe auxiliar, denominada *Server.php*, que recebe um código PHP e retorna o código em *HTML*. Dessa forma, utilizando a sintaxe fornecida pelo *framework Homero*, é possível gerar um código *HTML* acessível. Além disso, a classe *Server.php* oferece suporte às tarefas de predição, por exemplo, o retorno de possíveis erros que influenciam na acessibilidade do conteúdo.

Tendo um código PHP aberto na área de desenvolvimento, a *Acero* submete o código ao servidor onde se encontra o *framework Homero*, através do método

POST para a classe *Server.php*. Ao receber o código fonte, a classe *Server.php* transforma o código recebido em um arquivo *.php* e o executa. A saída do código PHP executado, isso é, o código em HTML, é então retornado para a Acero. Quando o desenvolvedor cria um código em PHP com elementos acessíveis na sintaxe definida pelo *framework Homero*, é possível salvar a saída do servidor PHP criando um código HTML automaticamente, que representa a interface acessível. Também, é possível apresentar os erros de acessibilidade encontrados no código PHP e apresentá-los ao usuário na *Janela Acero Output*.

No âmbito de integração, foi utilizado na construção deste módulo as técnicas de integração de dados, de plataforma e de controle. A técnica de integração de dados foi utilizada no compartilhamento do arquivo/código gerado pela *Acero* com o servidor *PHP* que aloja o *Framework Homero*, pois os dados enviados pela *Acero* devem ser compreensíveis pelo servidor para que este possa funcionar corretamente. A técnica de integração de plataforma foi utilizada para que houvesse a abstração necessária para o funcionamento correto entre o servidor e a *Acero*, dessa forma, a *Acero* por meio da *IDE Eclipse* envia uma requisição para o servidor, podendo ou não estar na máquina local, que responde a requisição a Acero. A técnica de integração de controle foi obtida através da implementação de métodos de tratamento de requisições na classe *Server.php* que ao receber uma requisição, por exemplo, para interpretar um código em PHP, aguarda o servidor terminar a execução e retorna o código em HTML à *Acero*. Ao receber o código executado, a *Acero* notifica o módulo responsável pela impressão de informações e, caso selecionado a opção de criar um arquivo com a saída do servidor, ela notifica o módulo responsável pela criação do arquivo com extensão *HTML*, com nome definido previamente pelo usuário.

4.3.5 Avaliador de diretrizes de acessibilidade diretamente no ambiente

Visando a avaliação de forma automática dentro da *IDE Eclipse*, foram buscadas ferramentas de avaliação de diretrizes de acessibilidade que permitissem enviar um código HTML e receber o resultado da avaliação do mesmo. Como descrito anteriormente, a *Acero* integra em seu código a utilização de duas ferramentas Web de avaliação, a *AcessMonitor* e a *Achecker*.

As ferramentas não disponibilizam documentações específicas de como utilizá-las a partir de um ambiente de desenvolvimento. Dessa forma, foi necessário utilizar bibliotecas *Java* voltadas à requisições Web do tipo *POST* para a utilização das ferramentas citadas anteriormente. Para extrair informações de campos e sobrecargas de requisições *POST* foi utilizado o *plugin*

Tamper Data[7] no navegador Mozilla Firefox, que permite visualizar os campos enviados em uma requisição Web e permite editá-los. Em geral, uma requisição do tipo POST possui campos específicos que formam o cabeçalho, como Agente de Usuário (*User−Agent*), URL de Referência (*Referer*), Anfitrião do Serviço (*Host*), entre outros campos. Além disso, possui um campo específico de dados, o *POST_DATA* que é o local em que será enviado o código fonte do desenvolvedor para ser avaliado.

Preenchendo os campos básicos requeridos pelas ferramentas de avaliação através da *IDE Eclipse*, o envio e avaliação de um código HTML dentro do ambiente de desenvolvimento ocorre de forma similar a requisição realizada a partir de um navegador. A *Acero* apresenta uma janela de escolhas (*Wizard*) dentro do ambiente de desenvolvimento ao usuário, referente a qual diretriz será analisada o código, se as ferramentas de avaliação (*AcessMonitor* e *Achecker*) serão utilizadas e se será apresentado ao usuário o reporte das informações no navegador interno da *IDE Eclipse*. Com base nas informações definidas pelo usuário, a *Acero* envia as requisições às ferramentas selecionadas. Por fim, a *Acero* salva o reporte das avaliações em um arquivo com o nome definido pelo usuário.

Com respeito à integração de ferramentas, foram utilizadas as técnicas de integração de dados, de plataforma, de processos e controle. A integração de dados foi obtida pelo envio de informações através da *Acero* de maneira que as ferramentas Web entendessem o tipo de requisição, retornando o resultado da análise. Por isso, foi necessário capturar o modo como as ferramentas Web tratam as requisições realizadas. A integração de plataforma foi obtida através do protocolo *HTTP*, que através do método *POST* permite a comunicação com a ferramenta oriunda de outra plataforma. Ampliando o suporte à fase de testes de acessibilidade no processo de desenvolvimento, obtida pela integração com ferramentas de validação Web dentro da *IDE Eclipse*, foi obtida a integração de processos, englobando assim suporte desde a fase de definição de requisitos até a fase de testes de acessibilidade. Também foi utilizada a técnica de integração de controle, na comunicação entre a ferramenta *Acero* e os servidores onde se encontram as ferramentas de validação de acessibilidade Web, onde a *Acero* requisita a análise do código do usuário e as ferramentas Web respondem a análise. Em seguida, a *Acero* notifica o módulo responsável pela criação do arquivo de saída no projeto do usuário que representa a análise de acessibilidade realizada sobre a interface.

[7] *https://addons.mozilla.org/pt−br/firefox/addon/tamper-data/*

4.3.6 Matriz de Rastreabilidade em PDF

Visando a melhoria da matriz de rastreabilidade, foram criados dois módulos. Um módulo que realiza a construção da matriz de rastreabilidade no formato PDF e outro módulo que complementa a matriz de rastreabilidade, verificando e corrigindo problemas de inconsistência nos arquivos de associação e nas classes criadas automaticamente. Dessa forma, o ambiente de desenvolvimento do usuário se torna mais flexível, robusto e coeso.

O funcionamento do módulo construtor da matriz de rastreabilidade no formato PDF na *Acero* é similar ao módulo de construção da matriz de rastreabilidade no formato *.ods* na *AccTrace*. Tendo os dados das associações, realizadas pela *AccTrace*, a *Acero* extrai as informações das associações e utiliza a biblioteca Java *iText PDF* [8] que está encarregada de criar o arquivo PDF.

Como complemento da matriz de rastreabilidade, a *Acero* oferta um módulo que permite verificar se as associações do arquivo *.acctrace*, que são utilizadas para construir a matriz de rastreabilidade e gerar comentários de acessibilidades nas classes, referem-se corretamente aos comentários gerados nas classes, aos artefatos UML e requisitos associados na *AccTrace*. Dessa forma, caso o desenvolvedor altere alguma referência do arquivo de associação *.acctrace*, quando invocado, o módulo atualiza automaticamente as informações nas classes. Dessa forma, a *Acero* realiza a rastreabilidade reversa de forma automática.

Sob o ponto de vista de desenvolvimento, a *Acero* importa as referência descritas no arquivo de relacionamento *.acctrace*, os artefatos UML e requisitos. Em seguida, apresenta ao usuário uma janela para escolher somente avaliar a consistência no arquivo de relacionamento e/ou avaliar um projeto específico com busca recursiva nos comentários de associação de todas as classes. Atualmente a *Acero* suporta a avaliação de rastreabilidade reversa somente nos arquivos com extensão *.java* e *.php*. Caso o usuário escolha a opção para avaliar somente o arquivo de relacionamento *.acctrace*, a *Acero* comparará se os artefatos UML e os requisitos indicados no arquivo *.acctrace* ainda existem, caso contrário, será apresentado ao usuário um aviso e a opção para remover esta entrada obsoleta do arquivo de relacionamento. Caso o usuário escolha verificar o arquivo de relacionamento *.acctrace* em conjunto com a avaliação da consistência de um projeto completo, será realizada a mesma ação apresentada e, adicionalmente, todos os arquivos do projeto selecionado serão importados pela *Acero*. Para cada classe do projeto, será avaliado se as referências do arquivo de associação *.acctrace* correspondem ao nome da classe e aos comentários presentes. Caso contrário, o usuário será notificado e será apresentada a possibilidade de remover automaticamente a entrada de-

[8] *http://itextpdf.com/*

satualizada. De forma similar, caso o arquivo de associação *.acctrace* tenha uma entrada para uma classe e nesta classe não esteja presente a referência desta entrada, será apresentada ao usuário a possibilidade de incluir automaticamente o comentário de acessibilidade na classe.

No tocante à integração de ferramentas, a integração de dados foi a principal técnica utilizada na implementação neste módulo, pois, o módulo deveria compreender os dados advindos do arquivo de associação *.acctrace* e caso houvesse a necessidade de atualização de informações, tais como nas entradas do arquivo de associação *.acctrace* ou nas classes, que a *Acero* fizesse de maneira que não comprometesse a utilização da ferramenta *AccTrace*.

4.3.7 Geração Automática de Classes PHP e Comentários Para Implementação de Acessibilidade

Como mencionado anteriormente, a *AccTrace* utiliza como base para geração de classes Java, a ferramenta *UML to Java Generator*. Para isso, é considerado como entrada um arquivo *.uml*, que contém os artefatos *UML* e suas referências aos requisitos. A ferramenta recebe como entrada o arquivo de associação *.acctrace*, que possui as associações entre os requisitos, artefatos *UML* e as ontologias de acessibilidade.

Durante o desenvolvimento deste trabalho foi buscada uma ferramenta compatível para criação de classes PHP a partir de um arquivo *.uml*. No entanto, como não foi encontrada a alternativa foi considerar a mesma ferramenta utilizada para gerar esqueletos de classe *Java*, a *Uml to Java Generator*[9], porém, modificando através de um editor de texto as palavras reservadas da linguagem *Java* para serem compatíveis com a linguagem *PHP*.

A Figura 4.4 apresenta dois trechos de códigos gerados automaticamente, nas linguagens *PHP* e *Java*, respectivamente. É possível notar que ambas as linguagens possuem estruturas similares, diferenciando basicamente em suas sintaxes. Além disso, ambos os códigos possuem as mesmas referências nos comentários de apoio ao desenvolvimento de acessibilidade.

Foi utilizada na construção deste módulo a integração de dados para que o funcionamento da ferramenta de geração de classes não fosse afetado. Visto que foi utilizado uma ferramenta concebida para gerar classes *Java*, foram modificadas apenas as palavras e termos reservados da linguagem para *PHP*. Também foi utilizada a técnica de integração de apresentação, para que o usuário tenha a mesma interface na criação códigos *Java* e códigos PHP. Além

[9]http://marketplace.obeonetwork.com/module/uml2java-generator

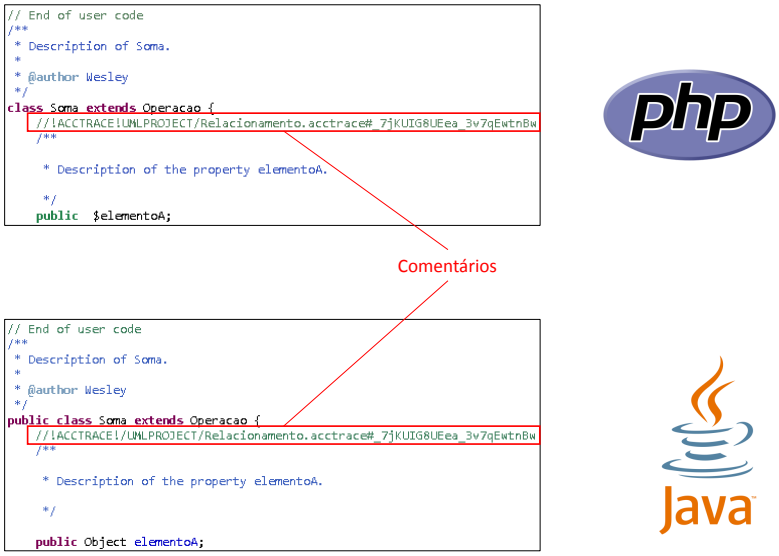

Figura 4.4: Exemplo de trechos de códigos gerados automaticamente para a linguagem *Java* e *PHP* com os comentários de apoio à implementação de acessibilidade

disso, foi utilizada a técnica de integração de processos, pois, através da definição de artefatos, requisitos e técnicas de implementação de acessibilidade, gera-se o código automaticamente na linguagem PHP.

4.3.8 Recuperador de Comentários PHP, C, C++, C#, PHP, JavaScript

Como mencionado, a recuperação dos comentários é o meio utilizado pela *AccTrace* para apresentar ao usuário as associações anteriormente realizadas para a classe com objetivo de auxiliá-lo na implementação de acessibilidade. Por motivos técnicos, os marcadores de recuperação de comentários da ferramenta *AccTrace* funcionam somente no editor *Java*, com arquivos na extensão *.java*, pois, a *AccTrace* utiliza o ponto de extensão *Compilation Participant* da biblioteca *Eclipse Java development tools (JDT)* que extrai a árvore de análise sintática para a linguagem *Java* e que permite extrair e recuperar informações do processo de compilação do código *Java*. Dessa forma, foi avaliada a necessidade de criar um botão que invocasse todos os comentários de apoio à implementação de acessibilidade do arquivo aberto pelo usuário e que também fosse compatível com linguagens que dispusessem da mesma expressão regular de início de comentários "//´´, isto é, *PHP, C, C++, C#, PHP, JavaScript* e *Java*.

Quando o usuário clica no botão *Recuperar Comentários de Acessibilidade* a *Acero* importa o arquivo de associação *.acctrace* definido pela *AccTrace* e apresenta ao usuário os comentários de acessibilidade do código que encontram-se

abertos na *IDE Eclipse*. A saída é apresentada na Janela (*view*) *AccTrace Comment View*. Para implementação desta funcionalidade, foi criado um botão do tipo *toolbar button*, que está localizado abaixo do menu de opções. A utilização do recurso é facilitado através da utilização do atalho de teclado *CTRL+6*. Quando o usuário realiza um clique simples sobre o botão ou utiliza o atalho indicado, a *IDE Eclipse* cria uma ação para que a *Acero* percorra o arquivo aberto pelo usuário e encontre os comentários no formato:

//!ACCTRACE!/PROJETO_DO_ARQUIVO_RELACIONAMENTO**/** NOME_ARQUIVO_ RELACIONAMENTO**.acctrace#** IDENTIFICADOR _DO_RELACIONAMENTO,

Foi utilizada na construção deste módulo, a integração de dados e de controle. A técnica de integração de dados foi utilizada para que a *Acero* extraísse os comentários de acessibilidade no código de distintas linguagens, pois, foi utilizado o mesmo arquivo de relacionamento proveniente da *AccTrace* que apresenta uma sintaxe baseada em *XML*. A integração de controle foi implementada da seguinte forma: quando o usuário clica no botão *Recuperar Comentários de Acessibilidade*, a *Acero* busca por comentários de acessibilidade e, caso encontre, requisita a ferramenta *AccTrace* a impressão na janela (*View*), denominada de *AccTrace Comment View*.

4.3.9 Preditor de erros de acessibilidade no código:

Como citado, o objetivo desta funcionalidade é utilizar as heurísticas do WCAG implementado no *framework Homero*, para extrair dicas de como implementadas acessibilidade no código do usuário. O módulo desenvolvido na *Acero* está organizado em duas partes: uma parte realiza a busca dos elementos declarados no código e outra parte fornece, com base no código, dicas para implementação da acessibilidade.

A função de buscar por arquivos no código realiza a procura por declarações no modelo *"ARQUIVO.EXTENSÃO "* e *'ARQUIVO.EXTENSÃO '* e suporta as extensões: *aif*, *ani*, *api*, *art*, *asc*, *asm*, *asp*, *avi*, *bak*, *bas*, *bat*, *bfc*, *bin*, *bin*, *bmp*, *bud*, *bz2*, *c*, *cat*, *cbl* , *cbt*, *cda*, *cdt*, *cfml*, *cgi*, *chm*, *class*, *clp*, *cmd*, *com*, *cpl*, *cpp*, *css*, *csv*, *cmf*, *cur*, *dao*, *dat*, *dd*, *deb*, *dev*, *dic*, *dir*, *dll*, *doc*, *dot*, *drv*, *ds*, *dun*, *dwg*, *dxf*, *emf*, *eml*, *eps*, *eps2*, *exe*, *ffl*, *ffo*, *fla*, *fnt*, *gif*, *gid*, *grp*, *gz*, *hex*, *hlp*, *ht*, *hqx*, *htm*, *html*, *icl*, *icm*, *ico*, *inf*, *ini*, *jar*, *jpeg*, *jpg*, *js*, *lab*, *lgo*, *lit*, *lnk*, *log*, *lsp*, *maq*, *mar*, *mdb*, *mdl*, *mid*, *mod*, *mov*, *mp3*, *mpeg*, *mpp*, *msg*, *msg*, *ncf*, *nlm*, *o*, *ocx*, *ogg*, *ost*, *pak*, *pcl*, *pct*, *pdf*, *pdf*, *pdr*, *php*, *phtml*, *pif*, *pif*, *pif*, *pl*, *pm*, *pm3*, *pm4*, *pm5*, *pm6*, *png*, *pol*, *pot*, *ppd*, *pps*, *ppt*, *prn*, *ps*, *psd*, *psp*, *pst*, *pub*, *pwl*, *qif*, *ram*, *rar*, *raw*, *rdo*, *reg*, *rm*, *rpm*, *rsc*, *rtf*, *scr*, *sea*, *sgml*, *sh*, *shtml*, *sit*, *smd*, *svg*, *swf*, *swp*, *sys*, *tar*, *tga*, *tiff*, *tmp*, *ttf*, *txt*, *udf*, *uue*, *vbx*, *vm*, *vxd*, *wav*, *wmf*, *wri* , *wsz* , *xcf*, *xif*, *xif*, *xif*, *xls*, *xlt*, *xml*, *xsl*, *zip*.

Quando encontra uma declaração, a *Acero* cria um objeto do tipo *File* com

o arquivo declarado e verifica por meio do método *exists()* se o arquivo existe. A existência de arquivos declarado no código são apresentados na *Janela Acero Output*.

Com base na diretriz WCAG 2.0, cada elemento do *framework Homero* possui informações referentes a erros que afetam sua acessibilidade. Por exemplo, o elemento *Image* possui erros que afetam diretamente e indiretamente a acessibilidade do elemento. Estes erros são apresentados na Tabela 4.2. Dessa forma, foi criado um método, denominado de *getPrediction* em que cada classe de elementos do *framework Homero*, quando invocado, retorna todos os erros que podem interferir diretamente e indiretamente no conteúdo do elemento.

Tabela 4.2: Erros que influenciam diretamente e indiretamente na acessibilidade do elemento *Image* do *framework Homero*

Erro	**Acessibilidade**	**Tipo de Erro**
Formato da imagem não permitido	Afeta indiretamente na acessibilidade do conteúdo	E_USER_ERROR
Arquivo de imagem não encontrado	Afeta indiretamente na acessibilidade do conteúdo	E_USER_ERROR
Texto auxiliar da imagem não especificado	Afeta diretamente na acessibilidade do conteúdo	E_USER_WARNING

A seguir é apresentado um exemplo de código que será utilizado para demostrar o funcionamento do módulo de predição da Acero:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
        <title></title>
    </head>
    <body>
        <?php
        echo ("Hello");
        ?>
        <div>
            <img src="myPic.jpg" alt="myPic" />
        </div>
    </body>
</html>
```

Ao executar o preditor da *Acero* sobre o código anterior, foram retornadas as seguintes informações na *Janela Acero Output*:

```
---Objects---
  File does not exist:[MYPIC.JPG] Line (13)


```

```
5 ------Prediction(Common Errors) ----
6
7
8 Line:4:(header)
9   E_USER_ERROR: Arquivo JS não encontrado.
10  E_USER_ERROR: Arquivo CSS não encontrado.
11  E_USER_WARNING: O título da página não foi especificado.
12
13 Line:12:(div)
14  E_USER_ERROR: Formato do conteúdo da Div não permitido.
15
16 Line:13:(image)
17  E_USER_ERROR: Formato da imagem não permitido.
18  E_USER_ERROR: Arquivo de imagem não encontrado.
19  E_USER_ERROR: O caminho da imagem não foi especificado.
20  E_USER_WARNING: O texto auxiliar da imagem não foi especificado.
```

Com base na saída da *Janela Output Acero*, pode-se notar que o arquivo *myPic.jpg* não foi encontrado na raiz do projeto. Na linha 4 existe um termo chave, o *header*, que leva a predição para o elemento *Header*. No elemento *Header*, existem dois erros*(E_USER_ERROR)* que afetam na execução do *script*, que são declarar arquivos JS e CSS inexistentes. Além disso, o elemento *Header*, apresenta um erro do tipo *E_USER_WARNING* que influencia diretamente na acessibilidade do conteúdo devido a não especificação do título (diretriz 2.4.2 da WCAG 2.0). O mesmo acontece para os elementos *Div* na linha 12 e para o elemento *Image*, na linha 13. Referente, ao elemento *Image*, existe um erro que pode afetar diretamente na acessibilidade, devido a não especificação do texto auxiliar da imagem (diretriz 1.1 da WCAG 2.0)

No âmbito de técnicas de integração de ferramentas, esta funcionalidade da *Acero* utilizou os seguintes modelos de integração: integração de plataforma, de dados, apresentação, controle e processo. No aspecto de integração de plataforma, a *Acero* utiliza o *framework Homero* que já possui a implementação de critérios de conhecimento de acessibilidade dos elementos Web. Para utilizar este conhecimento, a *Acero* faz requisição ao servidor que hospeda o *framework Homero* este servidor pode ou não estar na mesma plataforma de execução da *Acero*. Consequentemente, foi utilizada a integração de dados, pois, as informações provenientes da *Acero* devem ser compatíveis com as requisições aceitas pelas ferramentas Web. Além disso, é utilizada a integração de apresentação, pois são oferecidas aos usuários as informações de predição sobre um *layout* único, através da janela (*View*) *Acero Output*. De modo semelhante, é utilizada a integração de processos, pois, amplia-se a utilização do suporte às etapas do desenvolvimento, oferecendo a possibilidade de realizar um “pré-teste” do código em desenvolvimento.

4.3.10 Integração de outras ferramentas de apoio ao desenvolvimento de soluções acessíveis

No desenvolvimento da solução proposta *Acero* foi notada a possibilidade de integrar ferramentas indicadas pela W3C, com o objetivo de enriquecer a proposta de integração. As ferramentas integradas, a *Total Validator Basic* e a *Colour Contrast Analyser*, foram inseridas na pasta *tools* do projeto da Acero. Foram criadas duas entradas no *Menu Acero*, através do arquivo *plugin.xml*. Quando o usuário invoca uma das ferramentas pelo *Menu Acero*, é criado um evento e a ferramenta escolhida é aberta por meio do método *Runtime.getRuntime().exec(URL_DA_FERRAMENTA)*.

No contexto de técnicas de integração de ferramentas, utiliza-se a integração de processos, possibilitando ao usuário suporte à etapa de testes a fim de verificar pontos de acessibilidade específicos de seu código, tais como problemas de visão e conformidade com as diretrizes de acessibilidade.

4.4 Considerações Finais

Foi possível observar mediante a implementação da solução proposta neste trabalho que as possibilidades de criação de soluções de apoio a acessibilidade são amplas. Além disso, as técnicas de integração estudadas e utilizadas neste trabalho puderam ser aplicadas de acordo com o contexto já existente. Outro ponto relevante é que as técnicas de integração de ferramentas não estão separadas, ou seja, para a construção de uma solução geralmente são utilizadas distintas técnicas visando a interoperabilidade entre as ferramentas. Desse modo, as técnicas mencionadas nas implementações das funcionalidades da *Acero*, foram apenas aquelas que mais se destacaram durante a implementação da solução.

Além disso, foi percebido com a utilização da *IDE Eclipse* como base na implementação da ferramenta *Acero*, que a IDE possibilita o desenvolvimento de maneira rápida, fácil e integrada de ferramentas no contexto de acessibilidade.

Com o objetivo de avaliar a eficiência da ferramenta *Acero*, foi realizado um estudo empírico. A proposta do estudo e os resultados obtidos são apresentados em detalhes no Capítulo 5.

CAPÍTULO

5

Estudo Empírico

5.1 Considerações Iniciais

O processo de experimentação em Engenharia de Software permite verificar de maneira sistemática, disciplinada e computável novos métodos, técnicas, ferramentas, com o objetivo de avaliar novos processos e invenções [66]. Dessa forma, o objetivo desta seção é apresentar o protocolo utilizado e os resultados obtidos no estudo de caso realizado neste trabalho, visando avaliar a eficiência da solução de integração proposta. Para isso, foi conduzido um estudo empírico, planejado seguindo um protocolo consolidado e utilizado pela comunidade de Engenharia de Software, proposto por *Wohlin* [66], que sistematiza o estudo empírico.

Considerando que poucas avaliações foram realizadas com as soluções utilizadas na integração (*Acctrace* e *Homero*), o experimento apresentado neste capítulo teve como objetivo avaliar se a solução final obtida (*Acero*) poderia contribuir para preencher a lacuna existente no contexto do desenvolvimento de produtos de software acessíveis considerando o número reduzido de ferramentas que auxiliam os desenvolvedores a seguirem as diretrizes de acessibilidade na concepção de seus produtos.

O estudo empírico avaliou se a proposta fornecida pela *Acero* com a integração de outras ferramentas de apoio à acessibilidade permite, a partir de requisitos definidos pelo usuário, chegar a uma aplicação acessível e que permita a rastreabilidade entre os artefatos UML, requisitos e técnicas de implementação de acessibilidade. De forma geral, buscou-se investigar se a ferramenta obtida contribui para integrar acessibilidade no processo de desenvolvimento

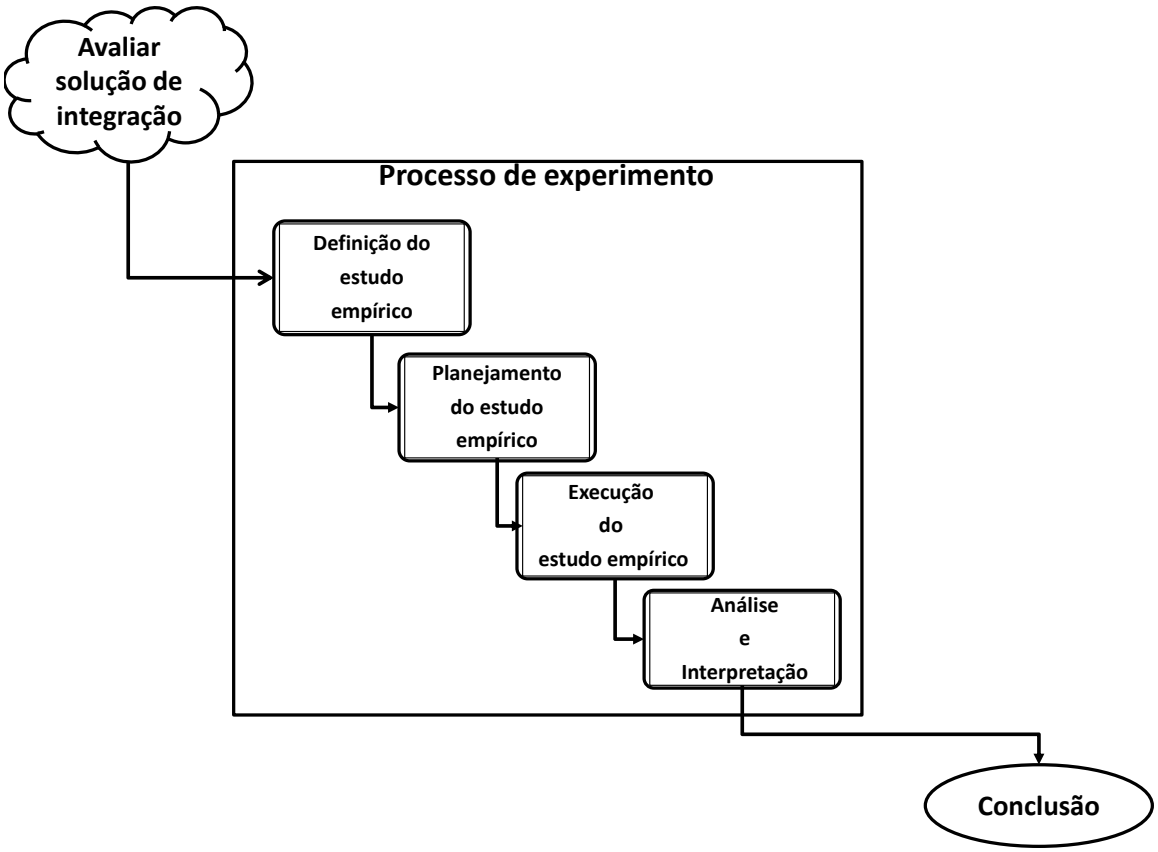

Figura 5.1: Estruturação do estudo empírico realizado neste trabalho. Adaptado de [66]

de software. A Figura 5.1 apresenta a estruturação do estudo realizado neste trabalho. Seguindo o protocolo de estudo empírico proposto por *Wohlin*, o estudo encontra-se dividido nas etapas: definição, planejamento, execução e análise dos dados e interpretação dos dados. A descrição de cada etapa está apresentada nas seções seguintes.

5.2 Definição

O objetivo do estudo empírico foi avaliar a eficácia da solução de integração *Acero*, analisando se a ferramenta auxilia na concepção de produtos acessíveis, assistindo nas etapas do processo de desenvolvimento, desde a definição de requisitos até a fase de testes de acessibilidade. Tal eficácia foi medida em termos do tempo decorrido para conclusão das tarefas propostas, facilidade de utilização e dificuldades encontradas pelos participantes para implementar uma interface acessível no nível 2.0 do WCAG da aplicação proposta neste estudo.

O estudo foi conduzido com um grupo de estudantes matriculados na *FACOM/UFMS*, nas modalidades de graduação, mestrado acadêmico e douto-

rado. Todos os participantes utilizaram a ferramenta *Acero* e avaliaram-na qualitativamente com o objetivo de extrair observações e identificar melhorias. A seguir são apresentados os fundamentos estabelecidos nesta etapa:

- **Objeto de Estudo:** o objeto de estudo é a entidade que foi estudada no estudo empírico. No âmbito deste trabalho, o objeto de estudo foi a solução de integração proposta.
- **Propósito:** o propósito define qual foi a intenção do estudo empírico. No âmbito deste trabalho, o propósito foi avaliar se a *Acero* contribui na concepção de produtos de software acessíveis.
- **Foco Qualitativo:** o foco qualitativo constitui a principal variável qualitativa que foi avaliada no estudo empírico. No âmbito deste trabalho, foram a eficiência, eficácia e facilidade de utilização da solução.
- **Perspectiva:** a perspectiva se refere ao ponto de vista a partir do qual os resultados experimentais foram interpretados. No âmbito deste trabalho, o estudo empírico foi realizado na perspectiva de desenvolvedores de software.
- **Contexto:** o contexto retrata qual a composição de sujeitos que executaram o estudo empírico. No contexto deste trabalho, foi por alunos de graduação, mestrado e doutorado da UFMS.

5.3 Planejamento

- **Seleção do Contexto:** Foi utilizada a amostragem por conveniência pois foram escolhidos estudantes das disciplinas ministradas por autores do trabalho.
- **Formulação de Hipóteses:** Foram formuladas dois tipos de hipóteses para o estudo, a fim de analisar o efeito da utilização da *Acero*, uma hipótese foi relativa ao auxílio da ferramenta na concepção de aplicações acessíveis e outra foi relativa à funcionalidade da matriz de rastreabilidade provida pela ferramenta. H representa as hipóteses nulas e H_a representa as hipóteses alternativas:
  - **Acessibilidade da aplicação desenvolvida**
    1. H_0: A ferramenta *Acero* não auxilia na concepção de aplicações acessíveis.
    2. H_{a0}: A ferramenta *Acero* auxilia na concepção de aplicações acessíveis.

– **Matriz de Rastreabilidade**

1. H_1: A matriz de rastreabilidade gerada pela ferramenta não auxilia na modificação de artefatos UML e requisitos.
2. H_{a1}: A matriz de rastreabilidade gerada pela ferramenta auxilia na modificação de artefatos UML e requisitos.

- **Seleção de Variáveis:** As variáveis independentes, ou seja, aquelas que são controladas no estudo são a ferramenta *Acero* e o modelo da interface a ser desenvolvido. Dentre as variáveis dependentes estão o tempo decorrido para implementação da interface acessível, facilidade de utilização da ferramenta proposta e a funcionalidade da matriz de rastreabilidade.

- **Seleção dos Participantes:** 8 alunos, sendo destes, 1 aluno de graduação, 6 alunos do mestrado acadêmico em Ciência da Computação e 1 aluno de doutorado em Ciência da Computação participaram do estudo de caso.

- **Treinamento:** Antes da realização do estudo os alunos assistiram uma aula expositiva em forma de vídeo sobre introdução a acessibilidade (https://www.youtube.com/watch?v=UutxdnHJD2Y) que teve a duração de 22 minutos e foi realizado um levantamento de perfil para analisar o nível de conhecimento técnico dos participantes. O levantamento de perfil dos participantes possuiu o objetivo de identificar a familiaridade dos participantes no desenvolvimento *Web*, conhecimentos sobre acessibilidade, ambiente de desenvolvimento *IDE Eclipse* e linguagens de programação *PHP*, *HTML* e *Java*. Este levantamento foi realizado por meio de um questionário (https://goo.gl/Na2jy6) respondido pelos participantes. As informações coletadas são apresentadas na Tabela 5.1.

Tabela 5.1: Nível de experiência teórica dos participantes.

Nível de Conhecimento Teórico	**Nenhum**	**Pouco**	**Razoável**	**Alto**
Orientação a Objetos	0	1	2	5
PHP	1	2	4	1
HTML	0	1	4	3
Java	0	1	4	3
IDE Eclipse	2	4	1	1
Acessibilidade Web	1	4	3	0
Diretrizes WCAG 2.0	2	5	1	0

A coleta de informações referente aos conhecimentos práticos avaliou se os participantes já tinham trabalhado em projetos de software, tanto na academia quanto na indústria. As informações coletadas podem ser visualizadas na Tabela 5.2.

Tabela 5.2: Nível de experiência prática dos participantes.

Nível de Conhecimento Prático	**Nenhum**	**Estudado em aula ou a partir de materiais**	**Usado em um projeto ou na indústria**	**Usado em vários projetos na indústria**
PHP	0	3	3	1
HTML	0	3	3	2
Java	0	4	2	2
IDE Eclipse	2	4	1	1
Acessibilidade Web	0	6	2	0
Diretrizes WCAG 2.0	5	3	0	0

O levantamento demonstrou que a maioria dos participantes não possuía conhecimento dos princípios de acessibilidade Web. A minoria dos participantes que conhecia, afirmou apenas ter conhecimento superficial a respeito das diretrizes *WCAG 2.0*. Considerando estes dados prover uma ferramenta para promover a utilização destas diretrizes de maneira simplificada tornou-se ainda mais relevante.

- **Projeto de estudo empírico realizado:** Com o objetivo de avaliar a ferramenta na perspectiva do desenvolvedor de software, todos os participantes do estudo fizeram a avaliação individualmente. Esta abordagem foi utilizada para verificar a facilidade de uso da ferramenta na perspectiva da primeira utilização do usuário.

  Para permitir a avaliação da ferramenta *Acero*, foi criado um diagrama de classes da aplicação que deveria ser desenvolvida por eles. A aplicação sugerida trata-se de uma calculadora simples. A Figura 5.3 apresenta o diagrama de classes.

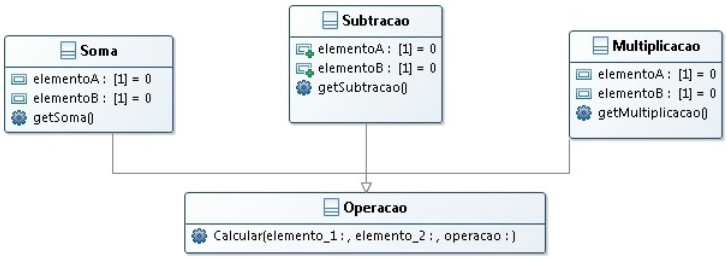

Figura 5.2: Diagrama de classes da aplicação proposta no estudo empírico

Foi criado também o diagrama de Casos de Uso, conforme apresentado na Figura 5.3.

Os requisitos funcionais e não funcionais são:

- RF1: O sistema deve realizar cálculos de operações (soma, subtração

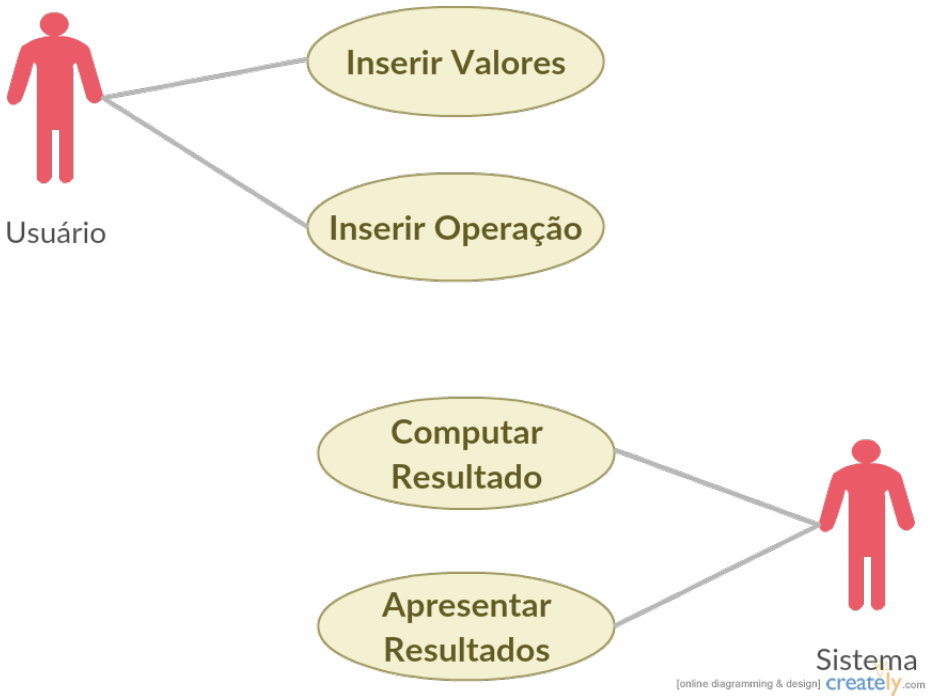

Figura 5.3: Diagrama de caso de uso da aplicação proposta no estudo empírico.

e multiplicação) com dois valores;

- RNF1: O sistema deve retornar o resultado do cálculo em menos de 2 segundos;
- RNFA1: O sistema deve prover textos alternativos a todos os elementos não textuais;
- RNFA2: O sistema deve permitir que todas as funcionalidades sejam acessíveis via teclado;
- RNFA3: O sistema deve entregar um conteúdo que não cause convulsão.

O *layout* da interface esperada que deverá ser apresentada pela aplicação é indicada na Figura 5.4.

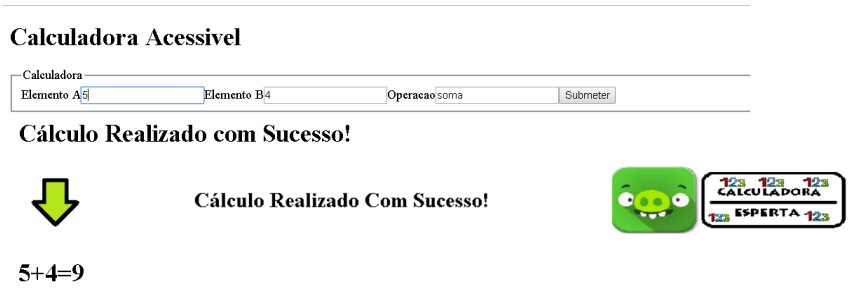

Figura 5.4: Interface da aplicação a ser desenvolvida no estudo empírico.

Espera-se que os participantes consigam executar todas as etapas do

processo de desenvolvimento a fim de obter uma aplicação com interface acessível e que, além disso, consigam gerar uma matriz de rastreabilidade em PDF, avaliar a interface desenvolvida por meio de um validador automático integrado na *Acero* e hospedem a aplicação em um servidor público da UFMS.

O plano de execução do estudo foi dividido em duas fases, uma para apresentação dos conceitos básicos de acessibilidade, outra dedicada ao desenvolvimento da aplicação com interface acessível e coleta de informações. A primeira fase foi realizada de forma virtual, através da disponibilização de um vídeo sobre introdução à acessibilidade. A segunda etapa foi realizada de forma presencial, no qual foram fornecidos aos participantes os formulários de consentimento, de perfil e de execução. Além disso, foram entregues os documentos necessários para o desenvolvimento da aplicação.

O formulário de consentimento teve como objetivo solicitar a permissão dos participantes do estudo quanto a utilização das informações concedidas. O formulário de execução teve como objetivo acompanhar com exatidão o tempo decorrido para implementação da aplicação e de obter métricas, tais como eficiência e nível de acessibilidade atingida, além disso também foi coletado opiniões e sugestões de melhoria. O formulário de execução é composto por questões quantitativas e qualitativas.

- **Instrumentação:** O estudo foi planejado para ser executado em ambiente controlado. O tempo estipulado para a execução do estudo foi de 120 minutos sendo que o responsável pelo estudo acompanhou os participantes no momento da execução. No início, os participantes receberam o modelo de aplicação a ser desenvolvida, o formulário de consentimento e os formulários de execução do estudo, a documentação de todas as funcionalidades e implementação da ferramenta *Acero*, exposta no Capítulo 4, reunidas por um roteiro.

  Houve o interesse em registrar o tempo gasto na avaliação porque desejava-se observar se a ferramenta auxiliava na elaboração de aplicações acessíveis considerando um tempo razoável. Em outras palavras, não seria aceitável que o tempo gasto fosse exorbitante.

  Em seguida, os participantes tiveram o tempo de 20 minutos para observar o roteiro e esclarecer dúvidas referentes à interface a ser desenvolvida. Finalizado esse tempo, os participantes foram instruídos a iniciar o desenvolvimento, marcando em cada tarefa exposta no formulário de execução o horário de início e o horário final.

Na Tabela 5.3, são apresentados os documentos utilizados durante a condução do estudo, indicando o momento de utilização de cada um deles.

Tabela 5.3: Documentos do estudo empírico

Documento	**Descrição**	**Momento**
Questionário de perfil	Auto identificação pelos participantes do nível de conhecimento e experiência prática com desenvolvimento web.	Antes da execução
Formulário de consentimento	Autorização dos participantes para utilização dos seus dados coletados no estudo.	Após a execução
Formulário de execução	Acompanhamento quantitativo e qualitativo do estudo.	Durante a execução
Diagrama de classes	Diagrama de classes da aplicação a ser desenvolvida	Durante a execução
Requisitos funcionais e não funcionais	Requisitos funcionais e não funcionais da aplicação a ser desenvolvida	Durante a execução
Imagem de aparência esperada da interface	Uma imagem de como a interface deve ficar depois de pronta	Durante a execução
Imagens a serem utilizadas na interface	Quatro imagens utilizadas na interface da aplicação a ser desenvolvida pelos participantes	Durante a execução

- **Ameaças à validade:** O tratamento de ameaças à validade tem o propósito de garantir que os resultados produzidos são válidos. As ameaças à validade foram divididas em três grupos. A validade interna determina que o resultado não é influenciado por fatores não medidos no experimento. A validade de construção considera os relacionamentos entre a teoria e a observação. Por fim, a validade externa determina o quanto é possível generalizar os resultados do estudo.

  Dentre as ameaças à validade interna estão:

  - Interferência no desempenho: Como o tempo necessário para implementar a aplicação proposta era uma métrica para medir a eficácia, pode-se argumentar que esta medida pode ser influenciada pelo tempo necessário para o entendimento da aplicação a ser desenvolvida. Para mitigar essa possibilidade, os participantes tiveram um tempo de 20 minutos para analisar o modelo e o roteiro para sanarem dúvidas referentes ao entendimento do estudo.

  Dentre as ameaças à validade de construção estão:

  - Conhecimento das hipóteses: Com o objetivo de que os resultados não fossem influenciados pelo conhecimento prévio dos participantes do estudo das hipóteses planejadas, as mesmas não foram reveladas.
  - Expectativas dos participantes: Para evitar que os participantes se baseassem no trabalho desenvolvido por outros participantes, estes não puderam se comunicar durante a execução do estudo.
  - Favorecimento do ambiente de desenvolvimento: Para evitar que algum dos participantes fosse beneficiado pelo ambiente, computador

ou outro recurso, foi padronizado um ambiente de desenvolvimento idêntico, com o mesmo sistema operacional e versão de *IDE Eclipse* e *Java*.

- Desinteresse dos participantes: Para evitar que os participantes do estudo perdessem o interesse pelo experimento, este foi realizado em sala de aula, compondo uma nota parcial dos participantes de pós-graduação matriculados na disciplina "Desenvolvimento de Software".

Dentre as ameaças à validade externa estão:

- Participantes e Usuários Reais: O público alvo da ferramenta proposta neste trabalho são usuários reais com distintos níveis de experiência em desenvolvimento Web e que utilizem *PHP*, *HTML* e *Java*. Dada a aspiração de que a amostra utilizada no estudo represente o público alvo, se dentre os participantes não tiverem ao menos um participante em cada nível de experiência em desenvolvimento Web, PHP e *Java*, pode haver comprometimento à representatividade da amostra. Dessa forma, foi levantado o questionário de perfil para avaliar os conhecimentos técnicos dos participantes e alocação dos membros nos grupos.

- **Roteiro**: Com o objetivo de avaliar detalhadamente a implementação da aplicação e aumentar a precisão do tempo decorrido, o roteiro dividiu o desenvolvimento em treze tarefas, ou tarefas:

  1. Definir os requisitos funcionais e não funcionais de acessibilidade da aplicação;
  2. Desenhar o diagrama de classes da aplicação;
  3. Relacionar os artefatos UML aos requisitos;
  4. Gerar quatro relacionamentos entre os artefatos UML, requisitos e técnicas de implementação de acessibilidade;
  5. Criar automaticamente os esqueletos das classes em PHP;
  6. Gerar a matriz de rastreabilidade no formato PDF;
  7. Gerar dois novos relacionamentos entre os artefatos UML, requisitos e técnicas de implementação de acessibilidade;
  8. Verificar e atualizar os comentários de acessibilidade das classes com o arquivo de relacionamento através da Acero;
  9. Definir a lógica de programação da aplicação;
  10. Desenhar a interface acessível da aplicação;

11. Gerar a interface Acessível da Aplicação
12. Analisar a acessibilidade da interface oferecida pela Acero.
13. Hospedar aplicação na Web.

5.4 Execução do Estudo de Caso e Análise dos Resultados

O tempo planeado para a realização das tarefas é apresentado na Tabela 5.4.

Tabela 5.4: Procedimentos adotados na execução do estudo

Procedimento Adotado	**Tempo**	**Modo de Aplicação**
1. Apresentação dos conceitos básicos de acessibilidade	30 minutos	Virtualmente
2. Análise do roteiro e do modelo de aplicação a ser desenvolvida	20 minutos	Presencial
3º Desenvolvimento da aplicação e interface	100 minutos	Presencial

O tempo planejado para execução do estudo foi de 120 minutos, dos quais 20 minutos foram utilizados para que os alunos lessem o roteiro e solucionassem dúvidas de execução. Devido ao número limitado de participantes no estudo, a eliminação de *outliers*[1] não foi realizada. Ao concluir a implementação da aplicação os arquivos dos participantes foram imediatamente recolhidos para evitar uma possível modificação dos mesmos.

A Figura 5.5 apresenta a compilação dos resultados do tempo médio, menor e maior utilizado pelos participantes para concluírem cada tarefa do roteiro. Observando o tempo médio de execução das tarefas, a etapa 10 foi a que mais demandou tempo, aproximadamente 14 minutos para seu término (os participantes adicionaram nesta etapa elementos acessíveis em distintas classes da aplicação) já a tarefa 11 foi a que demandou menor tempo médio de conclusão, com um valor médio de 1,6 minutos. É importante notar que na tarefa 11, as correções de rastreabilidade no código fonte foram realizadas de forma automática, sobre o projeto escolhido pelo usuário. Outro ponto de destaque é o valor expressivo obtido na tarefa 2 por um participante, com o tempo de 36 minutos, tal valor foi distinto da distribuição normal do conjunto.

Como descrito anteriormente, a execução do roteiro demanda a utilização de diversas ferramentas que foram integradas por meio deste trabalho de mestrado no ambiente de desenvolvimento IDE Eclipse. Essa distribuição é apresentada na Tabela 5.5. Já para as outras tarefas que foi utilizado especificadamente a *Acero*, o tempo médio para conclusão foi relativamente pequeno

[1] *Outlier* em estatística é uma dado que possui um valor atípico em consideração ao conjunto.

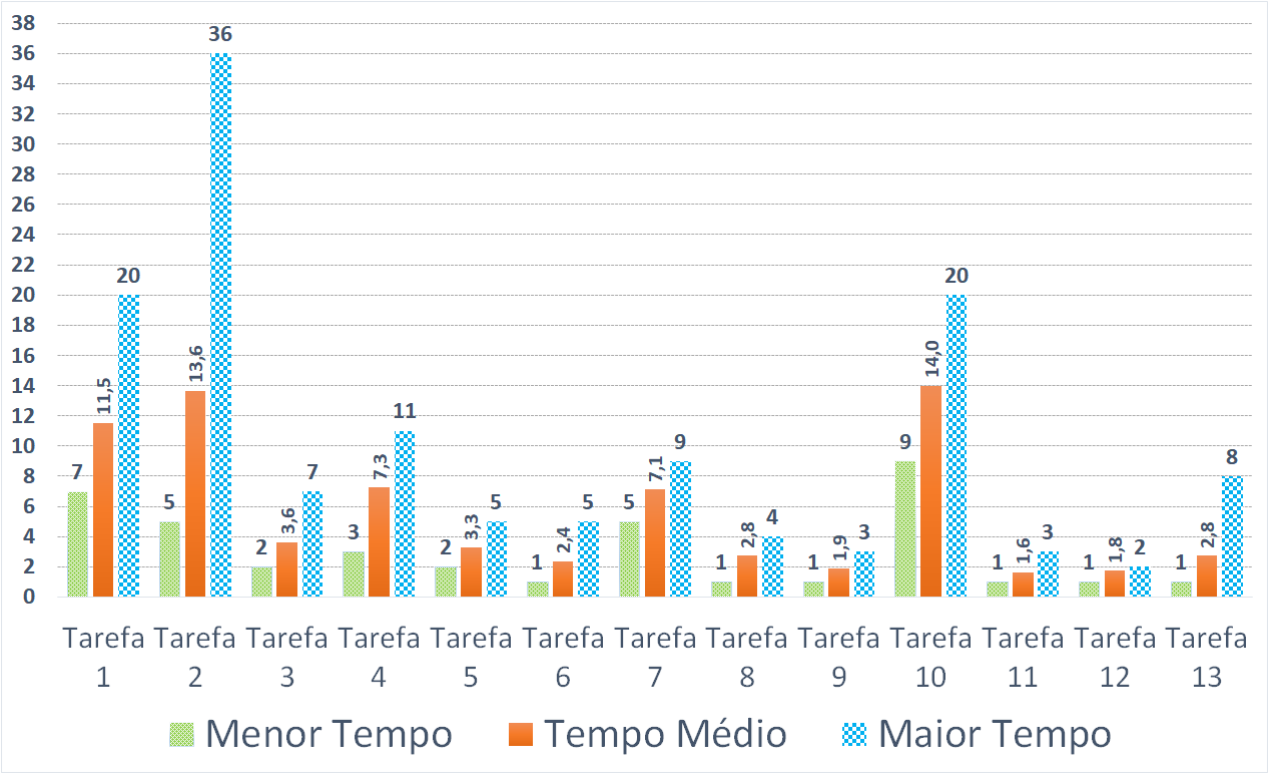

Figura 5.5: Tempo utilizado pelos participantes para concluírem as tarefas propostas pelo roteiro, em relação ao menor, maior e tempo médio

em comparação às tarefas que utilizaram outras ferramentas, tais como a *Requirement Designer*, *UML Designer* e a *AccTrace*.

Torna-se importante notar o reduzido valor médio de tempo obtido pelos participantes do experimento na tarefa 8. Nesta tarefa, os arquivos de código fonte foram atualizados de forma automática pela *Acero*, fato que antes da concepção deste trabalho não era possível. Dessa forma, neste estudo empírico o participante não precisou utilizar a matriz de rastreabilidade a todo momento para verificar quais classes deveriam ser atualizadas para novas entradas. Outro ponto de destaque é o tempo médio levado pelos participantes para avaliar a acessibilidade da interface. Provavelmente este valor relativamente pequeno foi alcançado devido aos usuários conseguirem submeter seus códigos diretamente dentro do ambiente de desenvolvimento utilizando um validador automático.

5.4.1 Facilidade de Uso

A maioria dos participantes considerou a utilização da ferramenta *Acero* simples. Apenas um participante encontrou dificuldades para utilização da *Acero*, pois, segundo o mesmo, há anos não tinha contato com o desenvolvimento de aplicações.

Segundo os participantes, a maior dificuldade para utilização da *Acero* foi

Etapas	Ferramenta Necessária Para Execução	Tempo Médio (Minutos)
Etapa 1	Requirement Designer	11,5
Etapa 2	UML Designer	13,6
Etapa 3	***Acero***	***3,6***
Etapa 4	AccTrace	7,3
Etapa 5	***Acero***	***3,3***
Etapa 6	***Acero***	***2,4***
Etapa 7	*AccTrace*	7,1
Etapa 8	***Acero***	***2,8***
Etapa 9	***Acero***	***1,9***
Etapa 10	***Acero***	***14***
Etapa 11	***Acero***	***1,6***
Etapa 12	***Acero***	***1,8***
Etapa 13	*	2,8

Tabela 5.5: Distribuição das etapas entre as ferramentas integradas. ** Processo Manual*

o desconhecimento de utilização da *IDE Eclipse*. Como pode ser visto na Figura 5.6, três participantes afirmaram que a falta de conhecimento sobre a *IDE Eclipse* foi o que mais prejudicou a utilização da *Acero*. Contudo, para outros dois participantes, o desconhecimento da ferramenta *Acero* foi o maior empecilho. Além disso, dois participantes disseram que por outros motivos a utilização da *Acero* foi influenciada, tais como a falta de prática no desenvolvimento e a união de todos os outros motivos citados anteriormente.

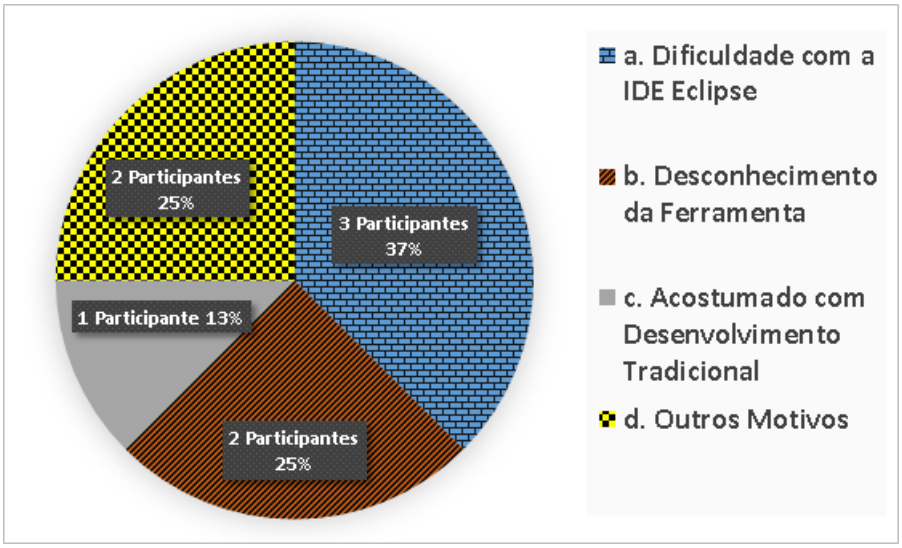

Figura 5.6: Maiores dificuldades segundo os usuários na utilização da ferramenta *Acero*

5.4.2 Influência da Matriz de Rastreabilidade no Gerenciamento de Mudanças e Importância da ferramenta

Segundo os participantes do estudo empírico a matriz de rastreabilidade no formato PDF fornecida pela Acero favorece no gerenciamento de mudanças. Todos os participantes concordaram que a matriz fornecida pela Acero é importante no gerenciamento de mudanças.

No âmbito de importância da ferramenta, todos os participantes também concordaram que o modo de abordagem da *Acero* é interessante e auxilia no processo de desenvolvimento de aplicações acessíveis.

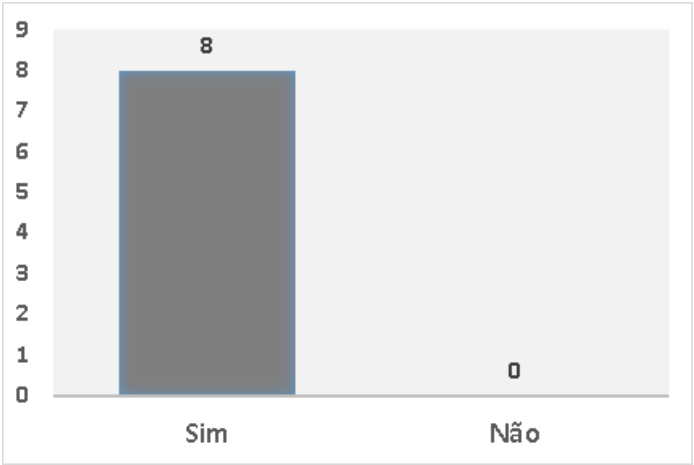

Figura 5.7: Importância e interesse da ferramenta Acero no processo de desenvolvimento de soluções acessíveis

5.4.3 Número de Elementos Desenvolvidos no Tempo Proposto

A fim de avaliar a qualidade das soluções desenvolvidas, foram coletados os códigos fontes de todos os participantes do estudo e avaliados individualmente, com o objetivo de verificar se foram implementados corretamente e se atendem as diretrizes de acessibilidade propostas. Cada tarefa proposta pelo roteiro foi classificada com o valor 1 ou 0, 1 para etapa corretamente realizada e 0 para a etapa erroneamente realizada. Dessa forma, a Figura 5.8 apresenta um gráfico que corresponde à soma dos valores atribuídos às tarefas corretamente realizadas. Houve apenas dois erros, de um participante na tarefa 8 e de outro participante na tarefa 10. O participante da tarefa 8 não inseriu corretamente os dois novos relacionamentos propostos. O participante da tarefa 10 preencheu incorretamente a *tag id* do elemento *img*, fato que influencia na acessibilidade do conteúdo. É importante notar que na tarefa 12 foi considerado acerto quando o participante analisou a acessibilidade da interface diretamente dentro do ambiente de desenvolvimento, independentemente da

interface ser acessível ou não. A análise da acessibilidade da interface será explanada posteriormente.

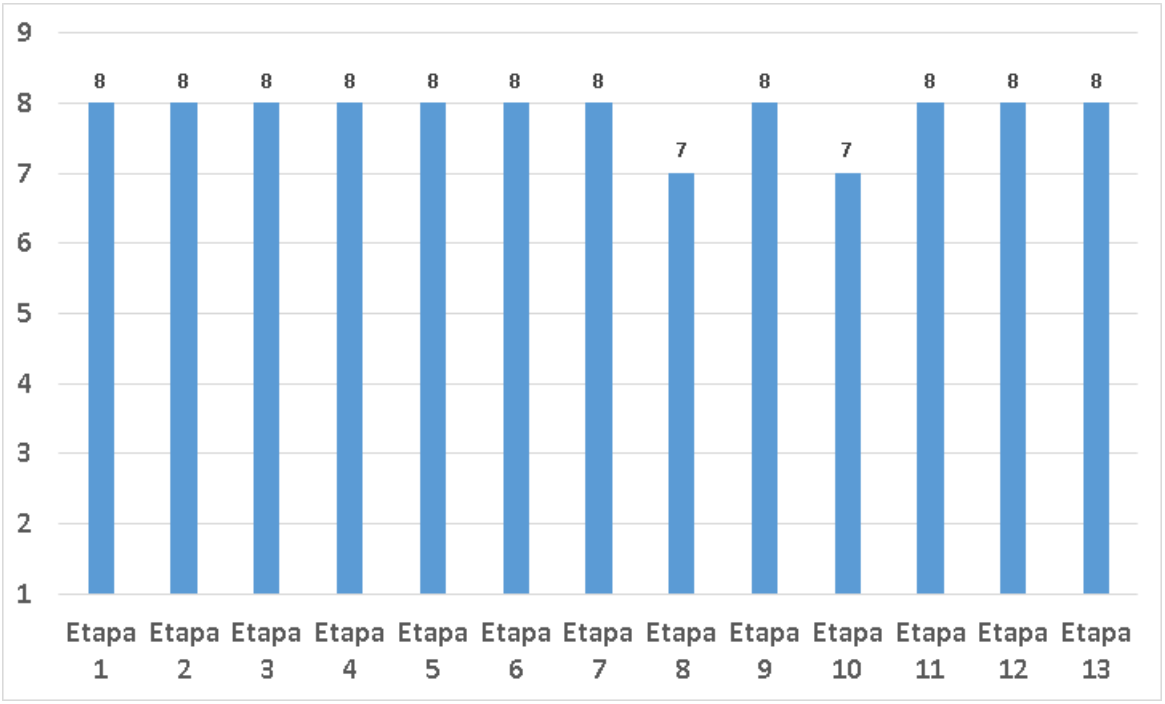

Figura 5.8: Etapas desenvolvidas corretamente

5.4.4 *Nível de Acessibilidade da WCAG 2.0 Atingido na Avaliação de Acessibilidade*

Com o objetivo de avaliar o nível de acessibilidade alcançado pela aplicação de cada participante, foram avaliados seus arquivos de saída de acessibilidade separadamente. Assim, cada participante submeteu seu código através do ambiente de desenvolvimento e, com isso, a *Acero* gerou um arquivo de saída com a análise de código. Apenas um dos participante não alcançou o nível de acessibilidade proposto para este estudo, devido ao erro na *tag* id, conforme mencionado anteriormente.

Com base nos resultados obtidos no experimento, é possível inferir que a hipótese H0, relacionada a acessibilidade da aplicação desenvolvida, é confirmada. Como mencionado, mesmo com um número de conjunto limitado, dos oito participantes apenas um não conseguiu atingir o nível AAA do WCAG 2.0.

No tocante a hipótese H1, relacionada com a importância da matriz de rastreabilidade no gerenciamento de mudanças, infere-se também que pode ser confirmada. Apesar do número reduzido dos participantes, todos relataram que a matriz é considerada útil e importante.

5.5 Considerações Finais

O estudo empírico conduzido foi relativamente limitado quanto a amostra e tempo pois participaram somente 8 alunos. Mesmo com tal limitação, o estudo realizado apresentou pontos positivos sobre a utilização da Acero e os benefícios trazidos por esta ferramenta no processo de desenvolvimento de *software*.

Em primeiro lugar, observou-se que a utilização da *Acero* atingiu os objetivos propostos, oferecendo suporte a distintas etapas do processo de desenvolvimento. Além disso, mostrou-se compatível e promissora no suporte à integração com outras ferramentas, visto que todos os participantes conseguiram finalizar o roteiro proposto para o estudo.

Pode-se afirmar que a facilidade de utilização da ferramenta Acero foi contemplada pois mesmo os participantes que possuíam conhecimentos de desenvolvimento e conhecimentos técnicos limitados conseguiram concluir o experimento proposto em um tempo menor que o limite proposto para o estudo. Além disso, os alunos conseguiram instanciar os elementos do *framework Homero* diretamente pela *Acero*, sem a necessidade de modificações secundárias no código ou da consulta à documentação fornecida pelo *framework*.

A ferramenta *Acero* estendeu as ferramentas já existentes. No âmbito da *Acctrace*, conforme pôde ser observado com a realização do estudo empírico, a *Acero* estendeu-a possibilitando que os usuários consigam gerar a matriz de rastreabilidade no formato PDF, um formato mais portátil do que o tradicional ofertado pela *AccTrace*. Além disso, a *Acero* possibilitou a criação automática de classes PHP com os comentários específicos para a implementação de acessibilidade. Deve-se notar também que as modificações dos novos relacionamentos entre requisitos, artefatos UML e técnicas de implementação de acessibilidade são adicionados automaticamente nas classes do projeto selecionado pelo usuário de forma mais eficiente. Esta tarefa demandou um tempo médio 2.8 minutos, tempo que dificilmente seria alcançado por meio da verificação da matriz de rastreabilidade e da modificação manual das classes do projeto.

O estudo apresentou ainda a importância de disseminar o conhecimento e conceitos de acessibilidade aos desenvolvedores. Também demonstrou a importância da concepção de ferramentas que auxiliem no processo de desenvolvimento de aplicações acessíveis, pois a acessibilidade deve ser considerada durante todo o processo para, ao final, obter uma solução acessível. Conforme observado no perfil dos participantes, o conhecimento sobre acessibilidade é restrito, entretanto, a ferramenta *Acero* em integração com outras demonstrou auxílio nas tarefas de acessibilidade no desenvolvimento de aplicações,

percorrendo as etapas do processo (ciclo de vida).

CAPÍTULO

6

Conclusões

Neste capítulo são apresentadas as principais conclusões deste trabalho. Primeiramente são discutidas as contribuições observadas e depois as limitações do mesmo. Ao final, são sugeridas futuras modificações e possibilidades de trabalhos futuros.

6.1 Objetivo do trabalho e Contribuições

Em primeiro lugar, este trabalho de mestrado foi iniciado com o objetivo de conceber uma solução que integrasse diversas ferramentas do âmbito do desenvolvimento de aplicações acessíveis, visando fornecer apoio ao usuário em todas as fases do desenvolvimento e conhecimento a respeito de integração de ferramentas. A partir do estudo empírico realizado neste trabalho, foi verificado que este objetivo foi alcançado, pois, diretamente no ambiente de desenvolvimento os participantes conseguiram percorrer todo o processo de desenvolvimento e geraram uma aplicação acessível no nível AAA do *WCAG* 2.0. Além disso, foi apresentado na Seção 4 o processo para integrar ferramentas e desenvolver novas funcionalidades, visando aportar conhecimentos e o funcionamento interno da solução desenvolvida neste trabalho.

A abordagem proposta pela *Acero*, pode contribuir de diversas formas:

1. Integração dos trabalhos desenvolvidos anteriormente pelo grupo de pesquisa de Engenharia de Software/UFMS.

2. Garantindo que a acessibilidade seja preocupação constante durante toda a fase de desenvolvimento, ofertando suporte a cada uma das fases de desenvolvimento diretamente no ambiente de desenvolvimento.

3. Contribuindo com a familiarização dos desenvolvedores com a acessibilidade e das diretrizes internacionais propostas para alcançar as mesmas.

4. Promovendo a facilidade de utilização, pois a ferramenta se encontra integrada no ambiente de desenvolvimento do usuário. Além disso, fornece *wizards* que facilitam ações e o preenchimento de campos importantes.

5. Automatizando processos que outrora eram manuais, por exemplo, o processo de atualização de relacionamento entre requisitos, artefatos UML e técnicas de implementação de acessibilidade das classes. Anteriormente à *Acero* o usuário deveria observar a matriz de rastreabilidade para verificar quais classes deveriam ser alteradas para então modificá-las. A *Acero* automatizou este processo para que através da seleção do projeto desejado pelo usuário, as classes possam ser atualizadas automaticamente.

6. Integrando e incrementando funcionalidades nas ferramentas, com a ampliação da *Acctrace* e do *framework Homero*. A matriz de rastreabilidade que antes era gerada unicamente na extensão *ODS* agora também é gerada no formato PDF, um formato mais portátil e comum. Além disso, foi estendido o *framework Homero* para que fosse utilizado diretamente dentro da *IDE Eclipse*, sem a necessidade do usuário ser dependente de sua documentação ou ter conhecimentos sobre programação orientada a objetos.

O foco deste trabalho foi a utilização de técnicas de integração de software no processo de desenvolvimento de ferramentas de acessibilidade. Esta seção são apresentados alguns pontos e contribuições percebidos deste trabalho de mestrado neste contexto.

Durante a fase de levantamento teórico foram encontradas as técnicas de integração de ferramentas de Software propostas por *Wasserman*[65]. O autor desenvolveu um trabalho pioneiro, propondo modelos de integração que são amplamente utilizados atualmente [24]. Ao desenvolver este trabalho de mestrado, verificou-se que, apesar da tecnologia ter avançado consideravelmente desde o estudo do *Wasserman*, os conceitos e aspectos de integração de ferramentas ainda são aplicáveis. Isso ocorre porque os modelos são descritos em alto nível e independem de linguagens de programação e tecnologia. No entanto, no desenvolvimento deste trabalho, foi percebido que o sucesso da integração está ligado à combinação das técnicas com o planejamento da integração. Por exemplo, para duas ferramentas utilizarem um mesmo arquivo é necessário planejar como será realizada a integração de dados.

É importante notar também que na literatura as técnicas de integração de software não especificam a maneira ou o recurso que será utilizado, pois, como supracitado estão descritas em alto nível. Entretanto, este trabalho de

mestrado percebeu exemplos de como as técnicas de integração de soluções de acessibilidade:

- Integração de controle: é possível utilizar mensagens entre as ferramentas, indicando algum aviso e sinal de sincronização entre as ferramentas;
- Integração de dados: é possível usar estruturas de dados, variáveis de programação, barramento de serviços, para o acesso simultâneo de dados, ou bloqueio de arquivos para acessos isolados, evitando que duas ou mais ferramentas acessem um arquivo ao mesmo tempo, ocasionando inconsistência de dados;
- Integração de plataforma: é possível utilizar servidores Web, serviços e *sockets* para utilizar ferramentas de diferentes plataformas;
- Integração de apresentação: é possível usar interface Web, de forma adaptável à necessidade de cada usuário;
- Integração de Processos: é possível utilizar o envio de mensagens, arquivos e dados entre as ferramentas, visando apoiar as fases do processo de desenvolvimento;

Como resultado deste trabalho destaca-se também a importância do planejamento da integração. Foi percebido que a etapa de planejamento é a principal etapa do processo de integração. O sucesso da solução integrada depende em grande parte do que já se tem e do que se pretende alcançar. É importante que as funcionalidades, tecnologias utilizadas, entradas e saídas de cada ferramenta sejam documentadas e se possível, comparadas com as funcionalidades, tecnologias, entradas e saídas das outras ferramentas que serão integradas. Com estas associações, a definição da solução final se tornará mais simples. Após definido este escopo da solução, é importante verificar quais tecnologias podem ser utilizadas para realizar a integração. Além disso, é importante verificar o custo da integração, pois é possível que o custo para integrar alguma funcionalidade seja maior do que desenvolvê-la novamente. Por fim, ao conhecer o que se tem e o que se pretende integrar, são identificadas quais técnicas de integração serão utilizadas para alcançar o objetivo proposto. Passa-se então para a fase de desenvolvimento.

Na etapa de desenvolvimento, os pontos que foram planejados anteriormente são executados. As técnicas de integração são utilizadas com o objetivo de fazer uma ligação entre ferramentas, criar e aprimorar novas funcionalidades. No entanto, é possível que nesta etapa apareçam problemas que não foram preditos na etapa de planejamento, tornando necessário replanejar a solução, analisando se o problema afeta somente alguma ferramenta ou funcionalidade.

Na etapa de testes devem ser estabelecidos mecanismos para verificar o funcionamento da integração e descobrir falhas. Foi percebido neste trabalho que quanto antes iniciar esta etapa, mais fácil será prever e corrigir problemas. Dessa forma, é interessante realizá-la concomitante ao desenvolvimento e planejamento.

As técnicas de integração fornecem mecanismos para unir ferramentas no processo de desenvolvimento. Dessa forma, o sucesso de uma solução de integração não depende exclusivamente das técnicas utilizadas mas também de todo processo realizado para alcançar a integração. Sendo assim, o planejamento é fundamental para o sucesso da solução final.

6.2 Dificuldades e Limitações

Este trabalho apresentou vários desafios. O primeiro deles foi descobrir como integrar ferramentas que utilizam distintas tecnologias com o objetivo de enriquecer o ambiente de desenvolvimento de software. A escolha das ferramentas tecnológicas foi primordial, por isso, foi um grande desafio descobrir quais ferramentas eram apropriadas e a forma que cada uma opera dentro do processo de desenvolvimento de aplicações acessíveis.

As seguintes limitações foram identificadas em relação a *Acero* em conjunto com as ferramentas integradas:

- Suporta apenas as linguagens de programação Java e PHP;
- Cobre somente as diretrizes WCAG 2.0;
- Necessita de um servidor PHP para o suporte ao *framework Homero*.

6.3 Trabalhos Futuros

Pesquisas sobre acessibilidade no processo de desenvolvimento e integração de ferramentas de apoio a acessibilidade são relativamente recentes e carecem de avanços, por isso, com o embasamento teórico adquirido e documentado por meio deste trabalho de mestrado, vários ramos de pesquisa podem ser explorados. A seguir são elencados sugestões de trabalhos futuros:

- Estender a solução para que possa atender o outros modelos de acessibilidade;
- Oferecer suporte a outras linguagens de programação;
- Investigar as possibilidades de utilização da solução desenvolvida em cenários reais, por exemplo na indústria de software;

- Desenvolver e integrar novas ferramentas no ambiente de desenvolvimento, com o objetivo de enriquecê-lo.

Além disso, o grupo de pesquisa de Engenharia de Software da UFMS também aporta conhecimentos que poderão ser aprimorados e transformados em novas ferramentas, por exemplo, ofertar dentro do ambiente de desenvolvimento novas ferramentas que apoiam a utilização das diretrizes de acessibilidade.

6.4 Considerações Finais

Este trabalho contribuiu em relação à integração de ferramentas de software no contexto do processo de desenvolvimento de software. Além disso, a solução desenvolvida neste trabalho de mestrado pode aportar auxílio aos desenvolvedores na concepção de soluções acessíveis.

Referências Bibliográficas

[1] (1992). Disability discrimination act 1992. disponível em: http://www.austlii.edu.au/au/legis/cth/consol_act/dda1992264/, acessado em 24 de julho de 2015. Citado na página 10.

[2] (2000a). Lei no 10.048, de 8 de novembro de 2000, disponível em:http://www.planalto.gov.br/ccivil_03/leis/l10048.htm, acessado em: 24 de julho de 2015. Citado na página 10.

[3] (2000b). Lei no 10.098, de 19 de dezembro de 2000, disponível em:http://www.planalto.gov.br/ccivil_03/leis/l10098.htm, acessado em: 24 de julho de 2015. Citado na página 10.

[4] (2004). Decreto nº 5.296 de 2 de dezembro de 2004, disponível em: http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2004/decreto/d5296.htm, acessado em 24 de julho de 2015. Citado na página 10.

[5] (2005). Disability act, disponível em: http://www.oireachtas.ie/documents /bills28/acts/2005/a1405.pdf, acessado em: 27 de julho de 2015. Citado na página 13.

[6] (2005). W3c, acessibilidade para o wai, disponível em: http://www.w3.org/wai/, acessado em: 20 de julho de 2015. Citado nas páginas 2, 10, 11, 13, 14, e 17.

[7] (2008). Web content accessibility guidelines (wcag) 2.0, disponível em:http://www.w3.org/tr/wcag20/, acessado em: 10 de agosto de 2015. Citado na página 12.

[8] (2009). How wcag 2.0 differs from wcag 1.0, disponível em:http://www.w3.org/wai/wcag20/from10/diff.php, acessado em: 10 de agosto de 2015. Citado na página 12.

[9] (2010). Japanese industrial standards and web accessibility infrastructure commission, disponível em:http://waic.jp/docs/jis2010/understanding.html, acessado em: 27 de julho de 2015. Citado na página 13.

[10] (2010). World wide web access: Disability discrimination act advisory notes ver 4.0 (2010), disponível em:https://www.humanrights.gov.au/world-wide-web-access-disability-discrimination-act-advisory-notes-ver-40-2010, acessado em: 24 de julho de 2015. Citado na página 10.

[11] (2011). Web standards for the government of canada, disponível em:http://www.tbs-sct.gc.ca/ws-nw/index-eng.asp, acessado em: 28 de julho de 2015. Citado na página 13.

[12] (2013a). Israeli standard 5568, disponível em:http://index.justice.gov.il/units/netzivutshivyon/kvatzim/1395_teken 5568negishutatareiinternet.pdf, acessado em: 24 de julho de 2015. Citado nas páginas 10 e 13.

[13] (2013b). Legge stanca, disponível em:http://www.webaccessibile.org/, acessado em: 27 de julho de 2015. Citado na página 13.

[14] (2014). European parliament, meps vote to make online public services accessible to everyone, disponível em:http://www.europarl.europa.eu/news/en/news-room/content/20140220ipr36573/html/meps-vote-to-make-online-public-services-accessible-to-everyone, acessado em: 24 de julho de 2015. Citado na página 10.

[15] (2014). Who-world health organization, disability and health, disponível em:http://www.who.int/mediacentre/factsheets/fs352/en/, acessado em: 19 de julho de 2015. Citado nas páginas 1 e 8.

[16] (2015). Brasil, lei 13.146, de 6 de julho de 2015, disponível em:http://www.planalto.gov.br/ccivil_03/_ ato2015-2018/2015/lei/l13146.htm, acessado em: 16 de julho de 2016. Citado na página 10.

[17] (2015). International telecommunication union, itu ict facts and figures:the world in 2015 features end-2015, disponível em:http://www.itu.int/en/itu-d/statistics/documents/facts/ictfactsfigures2015.pdf, acessado em: 19 de julho de 2015. Citado na página 1.

[18] (2015). W3c, authoring tool accessibility guidelines, disponível em:http://www.w3.org/tr/atag20/, acessado em 27 de julho de 2015. Citado na página 13.

[19] Aalst, W. V. D., Mylopoulos, J., and Sadeh, N. (2011). *Advanced Information Systems Engineering Workshops*. Citado na página 23.

[20] Agarwal, B., Tayal, S., and Gupta, M. (2010). *Software Engineering and Testing*. Computer science series. Jones & Bartlett Learning. Citado nas páginas 27 e 28.

[21] Alves, L. C., Leite, I. A. d. C., and Machado, C. J. (2008). Conceituando e mensurando a incapacidade funcional da população idosa: uma revisão de literatura. *Ciência e saúde coletiva*, 13:1199 – 1207. Citado na página 8.

[22] Aniche, M. (2015). *Testes automatizados de software: Um guia prático*. Casa do Código. Citado na página 27.

[23] Asplund, F., Biehl, M., El-khoury, J., and Törngren, M. (2011). Tool Integration Beyond Wasserman Diverging from Wasserman. *Advanced Information Systems Engineering Workshops*, pages 270–281. Citado na página 26.

[24] Asplund, F. and Törngren, M. (2015). The discourse on tool integration beyond technology, a literature survey. *Journal of Systems and Software*, 106:117–131. Citado na página 70.

[25] Baik, J. and Boehm, B. (2000). Empirical analysis of case tool effects on software development effort. *ACIS Int. J Comp. Inf. Sci.*, 1(1):1–10. Citado na página 21.

[26] Bangemann, T., Rebeuf, X., Reboul, D., Schulze, A., Szymanski, J., Thomesse, J.-P., Thron, M., and Zerhouni, N. (2006). Proteus?creating distributed maintenance systems through an integration platform. *Computers in Industry*, 57(6):539 – 551. E-maintenance Special Issue. Citado nas páginas ix, 23, e 24.

[27] Brown, A., Carney, D., Morris, E., Smith, D., and University, P. (1994). *Principles of CASE Tool Integration*. Oxford University Press, USA. Citado nas páginas ix, 23, 25, 26, 28, e 29.

[28] Cabrera-Umpiérrez, M. (2014). 3rd generation accessibility: information and communication technologies towards universal access. *Universal Access in the Information Society*, pages 1–3. Citado na página 2.

[29] da Presidência da República SDH/PR, S. D. H. (2009). *Tecnologia Assistiva*. Comitê de Ajudas Técnicas. Citado na página 8.

[30] de Branco, R. G. (2013). Acessibilidade nas fases de engenharia de requisitos, projeto e codificação de software: Uma ferramenta de apoio. Master's thesis, UFMS. Citado nas páginas ix, 4, 34, e 35.

[32] de Oliveira, R. C., Freire, A. P., Paiva, D. M. B., Cagnin, M. I., and Rubinsztejn, H. (2014). *A Framework to Facilitate the Implementation of Technical Aspects of Web Accessibility*, pages 3–13. Springer International Publishing, Cham. Citado nas páginas 4 e 36.

[33] Dennis, A. (2012). *Systems Analysis and Design, 5th Edition*. John Wiley & Sons. Citado na página 22.

[34] eMAG (2014). emag - modelo de acessibilidade em governo eletrônico, disponível em:http://emag.governoeletronico.gov.br/, acessado em: 20 de julho de 2015. Citado nas páginas 13 e 17.

[35] Farias, A. C. D. and Farias, A. C. D. (2001). Ferramentas CASE: Suporte, Adoção e Integração. Citado nas páginas 23 e 25.

[36] Fernandes, N., Batista, A. S., Costa, D., Duarte, C., and Carriço, L. (2013). Three web accessibility evaluation perspectives for ria. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, W4A '13, pages 12:1–12:9, New York, NY, USA. ACM. Citado na página 19.

[37] Freire, A. P. (2012). *Disabled people and the Web: User-based measurement of accessibility*. PhD in Computer Science, University of York. Citado na página 19.

[38] Grieves, J., Kaneko, M., and Corporation, M. (2009). *Engineering Software for Accessibility*. Developer Reference Series. Microsoft Press. Citado nas páginas xiii e 9.

[39] Groves, K. (2011). How expensive is web accessibility?, disponível em:http://www.karlgroves.com/2011/11/30/how-expensive-is-accessibility/, acessado em: 21 de julho de 2015. Citado na página 3.

[40] Hanson, V. L. and Richards, J. T. (2013). Progress on website accessibility? *ACM Trans. Web*, 7(1):2:1–2:30. Citado na página 19.

[41] Hennessy, J. and Patterson, D. (2014). *Organização e Projeto de Computadores, 4ª Edição: Interface Hardware / Software*. Citado na página 1.

[42] IBGE (2010). Cartilha do censo 2010, pessoas com deficiência, disponível em: http://www.pessoacomdeficiencia.gov.br/app/sites/default/files/

publicacoes/cartilha-censo-2010-pessoas-com-deficienciareduzido.pdf, acessado em: 19 de julho de 2015. Citado na página 1.

[43] Initiative, W. W. A. (2016). Web accessibility evaluation tools list, disponível em:https://www.w3.org/wai/er/tools/, acessado em: 4 de maio de 2016. Citado nas páginas xiii, 18, e 40.

[44] Karl Kurbel, T. S. (1994). Integration issues of information engineering based i-case tools. *Working Papers of the Institute of Business Informatics*. Citado nas páginas 22, 27, 28, e 29.

[45] Lazar, J., Goldstein, D., and Taylor, A. (2015). *Ensuring Digital Accessibility through Process and Policy*. Elsevier Science. Citado nas páginas 12, 13, 14, 15, 16, e 17.

[46] Lazar, J. and Greenidge, K.-D. (2006). One year older, but not necessarily wiser: an evaluation of homepage accessibility problems over time. *Universal Access in the Information Society*, 4(4):285–291. Citado na página 3.

[47] Loiacono, E. and Djamasbi, S. (2013). Corporate website accessibility: does legislation matter? *Universal Access in the Information Society*, 12(1):115–124. Citado na página 2.

[48] Loureiro, J. R. (2014). Acessibilidade web em redes sociais. Master's thesis, UFMS. Citado na página 39.

[49] Maia, L. S. (2010). Um processo para o desenvolvimento de aplicações web acessíveis. Master's thesis, UFMS. Citado nas páginas 3 e 34.

[50] Management Association, I. (2013). *Assistive Technologies: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications*. Information Science Reference. Citado nas páginas 8, 10, e 11.

[51] Maple, S. (2016). Java tools and technologies landscape 2016. Technical report, RebelLabs. Citado nas páginas ix e 37.

[52] Mattias Ganslandt, M. S. (2009). Web standardization. *Center for European Law and Economics*. Citado na página 11.

[53] Naik, K. and Tripathy, P. (2008). *Software testing and quality assurance: theory and practice*. Citado na página 30.

[54] Norman, R. J. and CHEN, M. (1992). A Framework for Integrated CASE. (March):18–22. Citado na página 25.

[55] Paulsen, H. B. (2016). Finding an optimal method for conducting accessibility evaluations of the norwegian tax administration website. Master's thesis, Oslo and Akershus University College of Applied Sciences. Citado na página 3.

[56] Rau, P.-L., Zhou, L., Sun, N., and Zhong, R. (2014). Evaluation of web accessibility in china: changes from 2009 to 2013. *Universal Access in the Information Society*, pages 1–7. Citado na página 3.

[57] Richards, J. T., Montague, K., and Hanson, V. L. (2012). Web accessibility as a side effect. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '12, pages 79–86, New York, NY, USA. ACM. Citado na página 19.

[58] Rierson, L. (2013). *Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance*. CRC Press. Citado nas páginas ix, 29, e 30.

[59] Rutter, R., Lauke, P., Waddell, C., Thatcher, J., Henry, S., Lawson, B., Kirkpatrick, A., Heilmann, C., Burks, M., Regan, B., et al. (2006). *Web Accessibility: Web Standards and Regulatory Compliance*. Designer to designer. Apress. Citado nas páginas 2 e 7.

[60] Schach, S. (2009). *Engenharia de Software - 7.ed.: Os Paradigmas Clássico e Orientado a Objetos*. McGraw Hill Brasil. Citado nas páginas 21 e 22.

[61] Sherman, P. (2001). Cost-justifying accessibility, disponível em:https://www.ischool.utexas.edu/ l385t21/au_wp_cost_justifying_accessibility.pdf, acessado em: 21 de julho de 2015. Citado na página 3.

[62] Souza Rodrigues, S., de Mattos Fortes, R. P., and Freire, A. P. (2016). *Towards Characteristics of Accessibility and Usability Issues for Older People - A Brazilian Case Study*, pages 117–128. Springer International Publishing, Cham. Citado nas páginas 3, 7, e 33.

[63] Tanenbaum, A. (2003). *Redes de computadores*. CAMPUS - RJ. Citado na página 40.

[64] Vigo, M. and Harper, S. (2013). Evaluating accessibility-in-use. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, W4A '13, pages 7:1–7:4, New York, NY, USA. ACM. Citado na página 19.

[65] Wasserman, A. I. (1990). Tool integration in software engineering environments. In *Proceedings of the International Workshop on Environments on Software Engineering Environments*, pages 137–149, New York, NY, USA. Springer-Verlag New York, Inc. Citado na página 70.

[66] Wohlin, C. (2000). *Experimentation in Software Engineering: An Introduction*. International Series in Engineering and Computer Science. Kluwer Academic. Citado nas páginas ix, 53, e 54.

[67] Zimmermann, G., Jordan, J. B., Thakur, P., and Gohil, Y. (2013). Genurc: Generation platform for personal and context-driven user interfaces. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, W4A '13, pages 6:1–6:4, New York, NY, USA. ACM. Citado na página 19.

Anexos

APÊNDICE

A

Como utilizar a Ferramenta *Acero*

A.1 Visão Geral Sobre o Ambiente de Desenvolvimento

Como citado anteriormente, a *IDE Eclipse* foi idealizada para o desenvolvimento *Java*, entretanto, suporta várias outras linguagens. Desenvolvida em *Java*, segue o modelo *Open Source*, possibilitando aos desenvolvedores a possibilidade de conceberem seus próprios *plugins*/extensões.

A Figura A.1 apresenta a tela principal do ambiente, apontando os campos da *IDE Eclipse* destacados. No lado esquerdo da figura é apresentado o explorador de projetos, onde é possível navegar pelos arquivos de projetos e realizar outras ações nos arquivos. Na parte inferior média, são apresentadas as janelas de visualização que representam as saídas de informações dos *plugins*. Na parte central da figura é apresentado a área de desenvolvimento, onde é possível modificar códigos e acessar outros recursos de desenvolvimento. Na parte superior da imagem são apresentados as opções (*menus*) utilizados para executar ações dentro do ambiente.

A.2 Acero

Como expostas, a *Acero* possui distintas funcionalidades que juntas possuem o objetivo de ofertar ao desenvolvedor um ambiente de desenvolvimento mais amplo e robusto.

Adiante serão apresentados exemplos de utilização dos recursos ofertados pela *Acero* e a forma que eles operam.

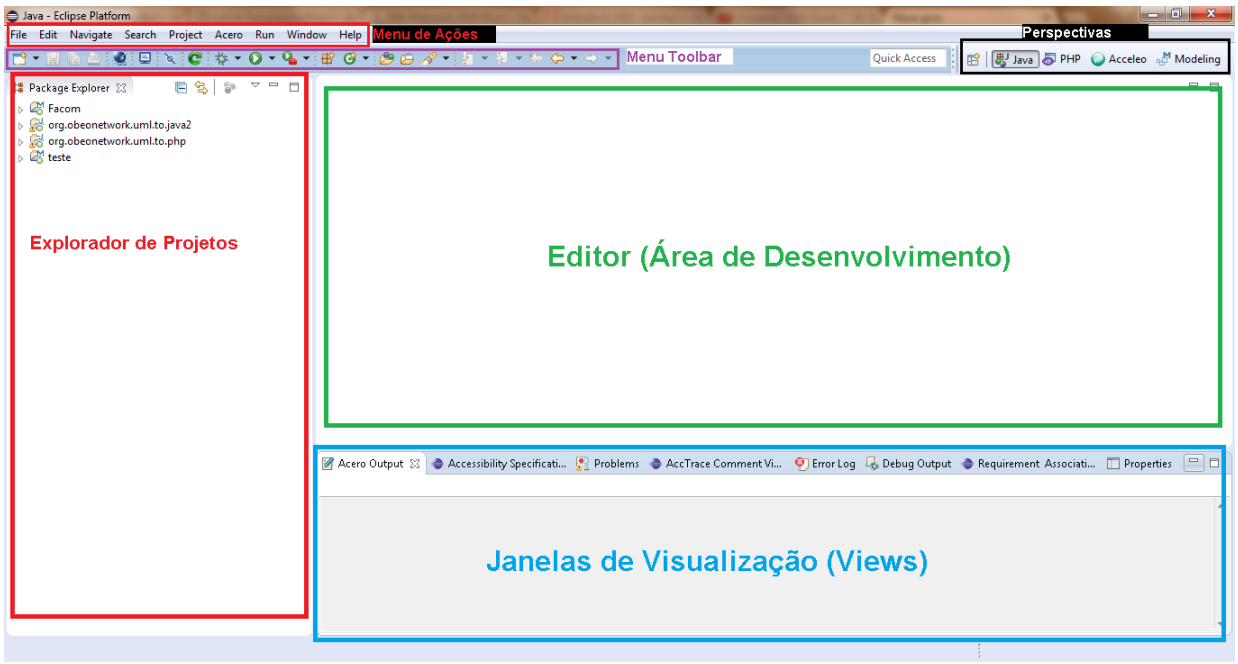

Figura A.1: Principais campos da IDE Eclipse

A.2.1 Facilitar a inserção de elementos do framework Homero

Esta funcionalidade da *Acero* tem por objetivo auxiliar o desenvolvedor na inserção dos elementos descritos pelo *framework Homero*.

A Tabela A.2.1 apresenta os elementos do *framework Homero* suportados pela ferramenta *Acero*, a descrição de cada um dos elementos e a etiqueta (*tag*) HTML associada ao elemento. Para cada elemento Homero descrito na tabela, a ferramenta *Acero* dispõe de *wizards* que guiarão o usuário passo a passo para criar os elementos acessíveis.

Elemento	**Descrição**	**Tag HTML**
Homero Template (Modelo de Documento)	Template ou modelo básico para o funcionamento do {framework Homero}	*
HTML Code (Código em HTML)	Inserir um trecho de código em HTML, por exemplo para espaço	*
Cite (Citação)	Inserir uma citação acessível no código	*<cite...*
Button Image (Botão Imagem)	Inserir um botão imagem acessível no código	*<input type="image"...*
Button Reset (Botão Reset)	Inserir um botão de *reset* acessível no código	*<input type="reset"...*
Button Submit (Botão de Submissão)	Inserir um botão de submissão acessível no código	*<input type="submit" ...*
Combo Box (Caixa de Combinação)	Inserir uma caixa de combinação acessível com diversas opções no código	*<select> <option...*
Input File (Entrada de Arquivo)	Inserir uma entrada do tipo file (arquivo) acessível no código	*<input type="file"...*
Input Password (Campo para Senha)	Inserir um campo de senha acessível no código	*<input type="password"...*
Input Radio (Botão de Opção)	Inserir um botão de opção acessível no código	*<input type="radio"...*
Input Text (Campo de Texto)	Inserir um campo de texto acessível no código	*<input type="text"...*
Text Area (Área de Texto)	Inserir uma área de texto acessível no código	*<textarea...*
Div (Divisão)	Definir uma divisão ou uma seção no código	*<div...*
Emphasis (Ênfase)	Inserir um texto em ênfase acessível no código	*<em...*
Group (Grupo de Elementos)	Inserir um grupo acessível de elementos no código	*<fieldset>...*
Imagem (Imagem)	Inserir uma imagem acessível no código	*<img...*
Link (Enlace)	Inserir um enlace acessível no código	*<a...*
Long Explanatory (Explicação Longa)	Inserir uma explicação longa acessível no código	*<blockquote...*
Lst (Lista)	Inserir uma lista acessível no código	*<ol .../ <ul... / <dl...*
Media (Mídia)	Inserir uma mídia acessível no código	*<audio... / <video...*
Object (Objeto)	Inserir um objeto acessível no código	*<object ...*
Paragraph (Parágrafo)	Inserir um parágrafo acessível no código	*<p ...*
Short Explanatory (Explicação Curta)	Inserir uma explicação curta acessível no código	*<q ...*
Span (Agrupamento na Linha)	Inserir um agrupamento acessível no código / Separar trecho da linha	*<span ...*
Strong (Texto Importante)	Inserir um texto importante acessível no código	*<strong ...*
Subscript (Subscrito)	Inserir um texto subscrito acessível no código	*<sub ...*
Superscript (Sobrescrito)	Inserir um texto sobrescrito acessível no código	*<sup*
Table (Tabela)	Inserir uma tabela acessível no código	*<table ...*
Title (Título)	Inserir um título acessível no código	*<h1 ... / <h2 ... / ... / <h6 ...*

Tabela A.1: Elementos do *framework Homero* suportados na Acero.

A *Acero* possui três partes que auxiliam o usuário a criar elementos acessíveis do *framework Homero*:

1. Adicionar o *template* necessário para o correto funcionamento do *framework Homero* em um arquivo: Caso o usuário deseje criar um novo arquivo PHP contendo o *template* básico para o funcionamento do *framework*, é possível realizar diretamente por meio da ferramenta *Acero*, seguindo o seguinte processo:
   No Explorador de Projetos pressionando o botão direito do *mouse* sobre o projeto atual → *New* → *Other* → *Acero* → *Homero Page*. Seguindo a sequência anterior, o usuário será redirecionado a tela apresentada na Figura A.2.

   Ao inserir os dados nos campos da *wizard*, tais como, nome do arquivo, cabeceira da página, codificação da página e o idioma da página, então

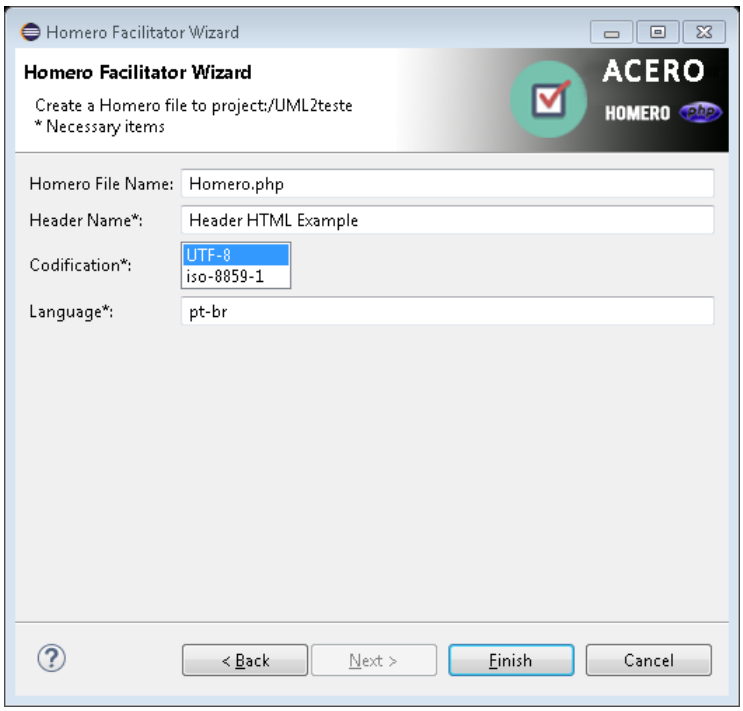

Figura A.2: Criação de um novo arquivo com a *template da Homero*

a *Acero* criará automaticamente no projeto o *template* em um arquivo com o nome escolhido pelo usuário. Utilizando os dados apresentados no exemplo da Figura A.2 será criado o arquivo *Homero.php* com o respectivo *template*. A Figura A.3 apresenta estas saídas. Do lado esquerdo da figura é possível observar o arquivo criado dentro do projeto atual e no centro da imagem é possível observar o conteúdo do arquivo *Homero.php* que possui o *template* básico do *framework Homero*.

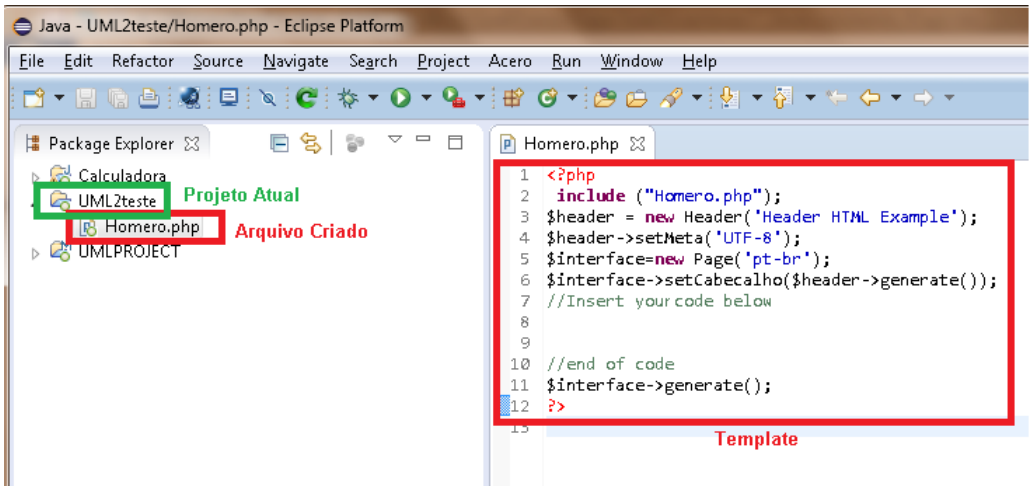

Figura A.3: Exemplo de arquivo criado no projeto atual contendo o *template* básico para o funcionamento do *framework Homero*

Caso o usuário já tenha um arquivo criado e ainda não possua o *template*

básico para inserir elementos do *Homero*, pode inseri-lo automaticamente através o seguinte processo: clicando uma vez com o botão esquerdo do *mouse* sobre o local em que será inserido o *template* → Menu *Acero* → *Wizard* de Elementos Homero. A Figura A.4 apresenta esta sequência.

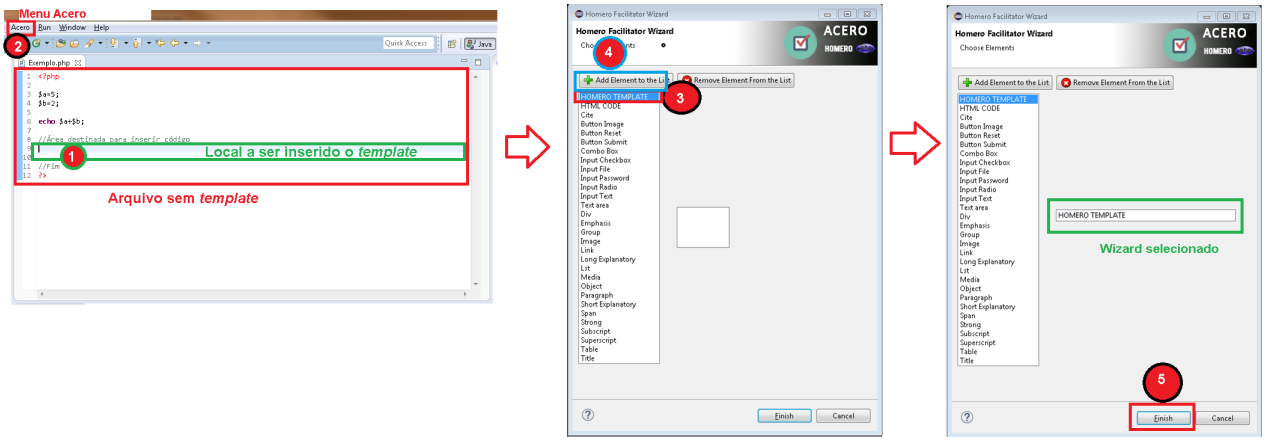

Figura A.4: Inserção do *template* básico em um arquivo já existente

Ao pressionar o botão *Finish* o usuário será direcionado a uma tela similar a apresentada na Figura A.5. No campo *Homero's file path* é inserido o local de onde se encontra o código principal do *framework* Homero, no caso da figura, o *framework* se encontra em uma pasta no disco *C:*. Caso o usuário não possua o *framework Homero* em sua máquina e tenha um servidor com o *framework* instalado em sua rede de acesso, basta preencher o campo com *Homero.php*. Após o usuário inserir os outros campos da *wizard*, será criado no local selecionado anteriormente, um trecho de código que representa o *template* para o funcionamento básico do *framework Homero*. A saída do exemplo é apresentada na Figura A.6.

2. Inserir elementos da Homero com acessibilidade através de *wizards* de apoio ao usuário: Caso o usuário queira adicionar um novo elemento do *Homero* no código, é possível inseri-lo de forma semiautomática na Acero, através de *wizards*. Para o correto funcionamento, os elementos devem ser inseridos no espaço pré determinado pelo *Template*, entre os comentários *// Insert your code below e //end of code* e deve possuir o *template* básico de funcionamento, explanado anteriormente. A área de inserção de elementos encontra-se destacada na Figura A.7.

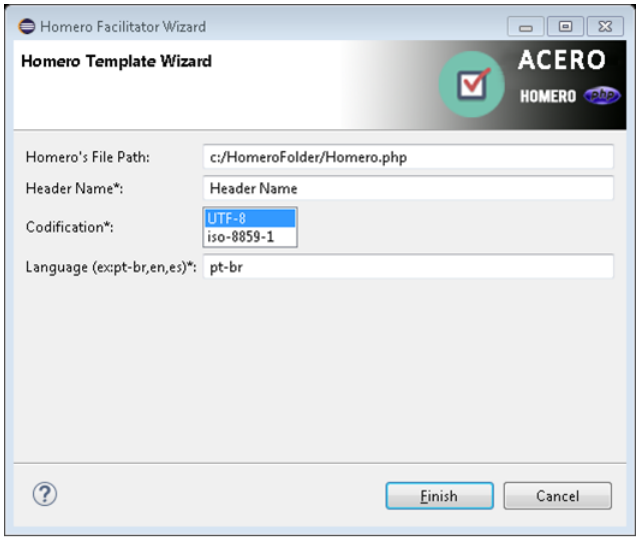

Figura A.5: Inserção de dados na *wizard* responsável por inserir o *template* do *framework* no código

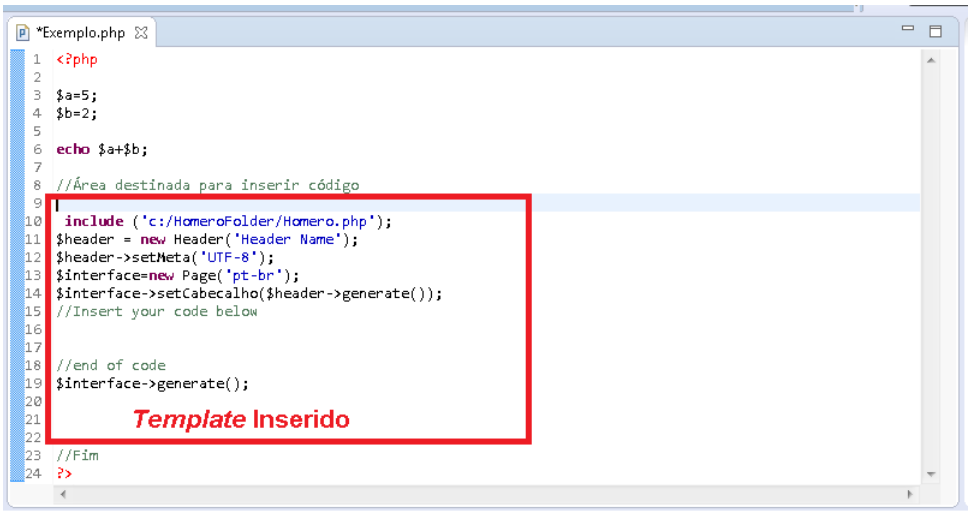

Figura A.6: Inserção de dados na *wizard* responsável por inserir o *template* do *framework* no código

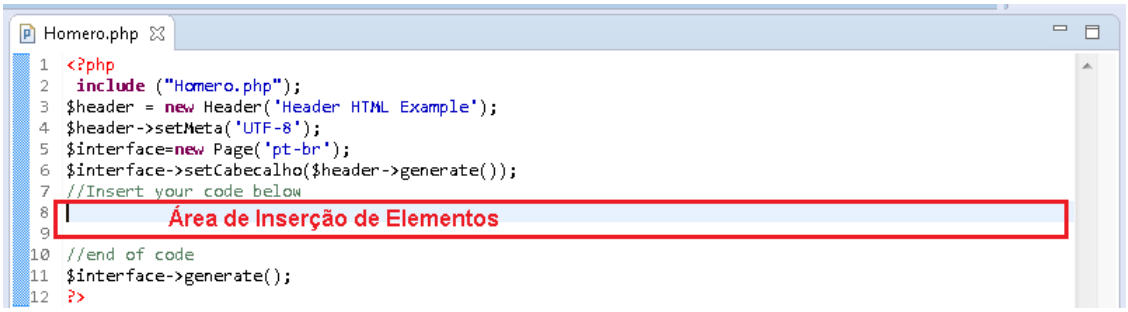

Figura A.7: Espaço reservado para inserção de elementos.

Para o usuário inserir um elemento *Homero* no código, é necessário seguir os seguintes passos: Pressionar um clique simples com o botão esquerdo do *mouse* sobre o local em que será inserido o elemento → Menu *Acero* → *Wizard* de Elementos Homero. Realizando este processo o usuário será redirecionado a seguinte tela, apresentada na Figura A.8.

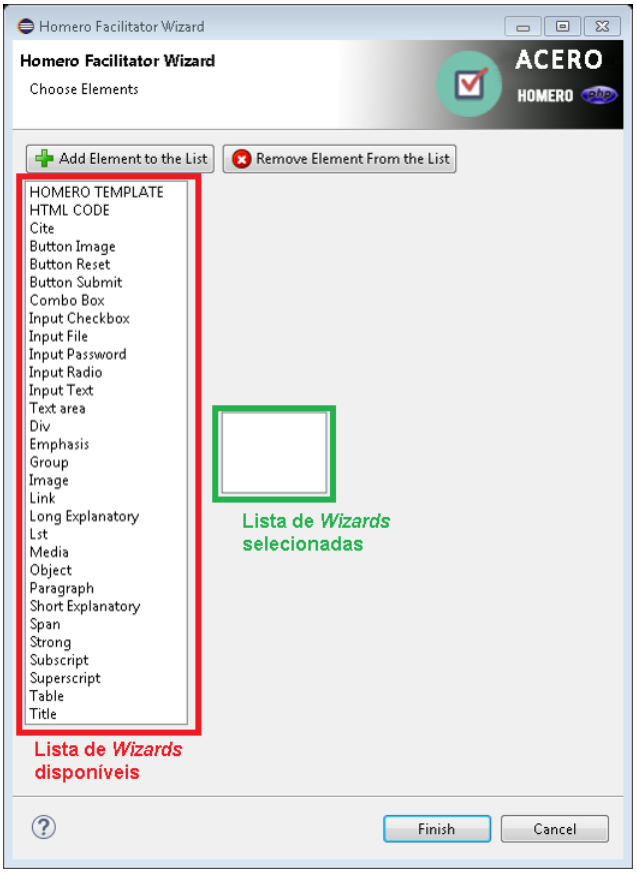

Figura A.8: Tela para escolha dos elementos a serem inseridos no código do usuário

Como apresentado na Figura A.8, esta tela possui duas listas, uma lista de *wizards* de elementos do *framework Homero* que a *Acero* suporta e outra segunda lista das *wizards* selecionadas.

Na Figura A.9 apresenta a sequência de passos necessários para incluir dois elementos da Homero no código, um título (*Title*) e uma imagem (*Image*) acessível, na lista de *wizards* selecionadas. Inicialmente, seguindo a ordem de inserção de elementos, o usuário adiciona o elemento *Title* (Título), pressionando uma vez com o botão esquerdo do *mouse* sobre *Title* e em seguida pressionando o botão *Add Element to the List*. Após

o elemento título na lista de escolhidos, o usuário adiciona a imagem, selecionando com o botão esquerdo do *mouse* o elemento *Image* (Imagem) e em seguida clicando o botão *Add Element to the List*. Caso o usuário deseje remover um elemento na lista de escolhidos, basta selecionar com o botão esquerdo do *mouse* o elemento na lista de *wizards* selecionadas e pressionar o botão *Remove Element From the List*. Como neste exemplo anterior não mencionava a exclusão de um elemento, o usuário pode concluir a escolha pressionando o botão *Finish*, no passo cinco.

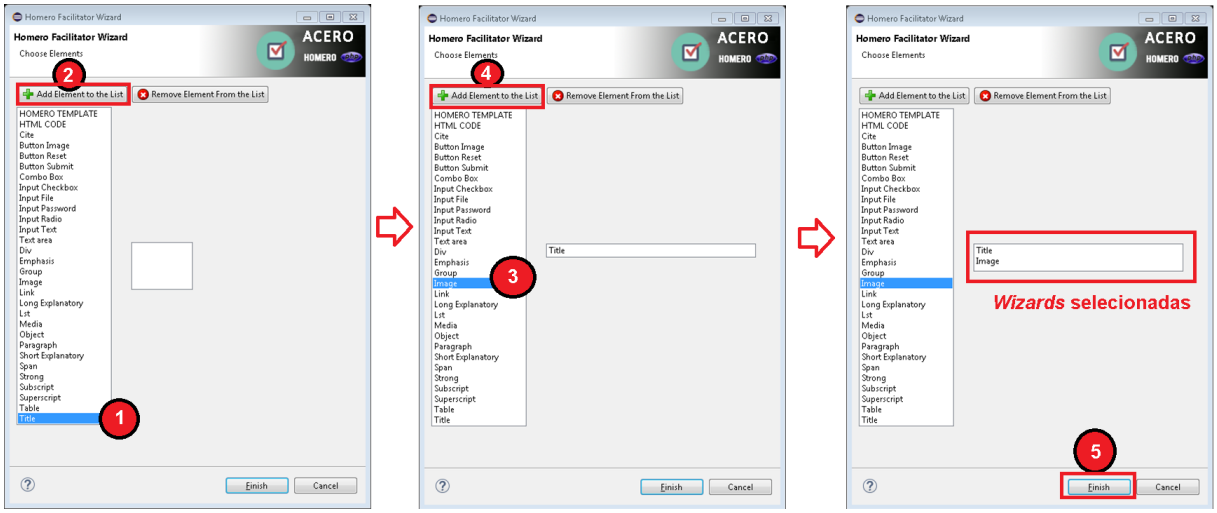

Figura A.9: Tela para escolha das *wizards* dos elementos a serem inseridos no código

Após o botão *Finish*, descrito na Figura A.9, o usuário será redirecionado as *wizards* específicas dos elementos, neste caso a *wizard* do elemento *Title* (Título) e a *wizard* do elemento *Image* (Imagem). Seguindo a ordem de inserção o primeiro elemento a ser definido será o Título. A Figura A.10 apresenta a sequência de preenchimento da *wizard* do elemento Título. Neste elemento específico, não existem campos que afetam na acessibilidade do conteúdo. Após o usuário definir a ordem do elemento na página, o texto do título, o identificador e caso houver, a classe CSS associada ao elemento, quando o usuário clicar no botão *Finish* será inserido no espaço selecionado anteriormente do código, o trecho necessário para gerar o elemento Título no *framework Homero*. Além disso, cada *wizard* de elementos possui botões de apoio ao usuário, tais como para obter a documentação do elemento trazido pela *Homero*, o código fonte do elemento no *framework Homero* e informações e exemplos de utilização do elemento advindos do sítio de aprendizado HTML da W3C[1].

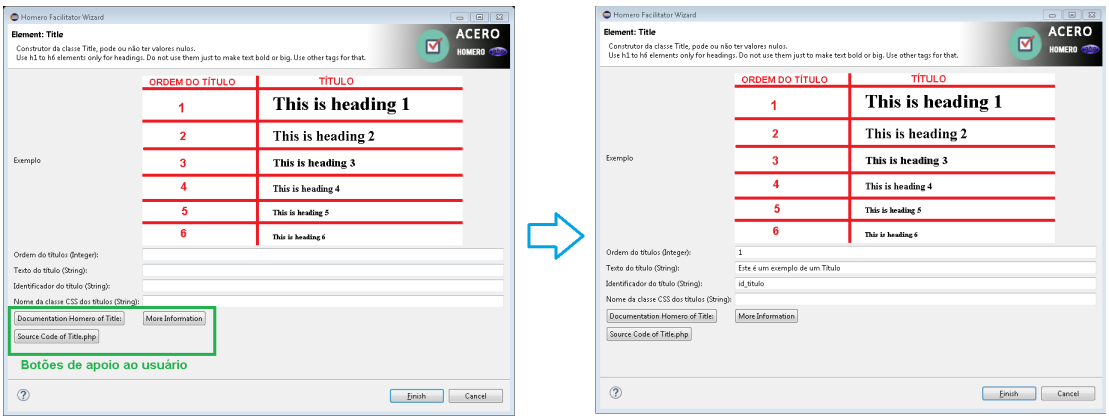

Figura A.10: *Wizard* do elemento Homero *Title* (Título)

Após o preenchimento da *wizard Title*, apresentada na Figura A.10, o usuário será redirecionado a outra *wizard* selecionada anteriormente, referente ao elemento *Image*. A Figura A.11 apresenta a sequência de utilização desta *wizard*. Pode-se notar que a *wizard* apresenta um campo de preenchimento que possui um asterisco (*). Este símbolo indica que o campo indicado afeta diretamente na acessibilidade do elemento e deve obrigatoriamente ser preenchido, inclusive o botão *Finish* da *wizard* Imagem encontra-se desabilitado, tornando-se habilitado somente quando o usuário preencher o campo de texto alternativo.

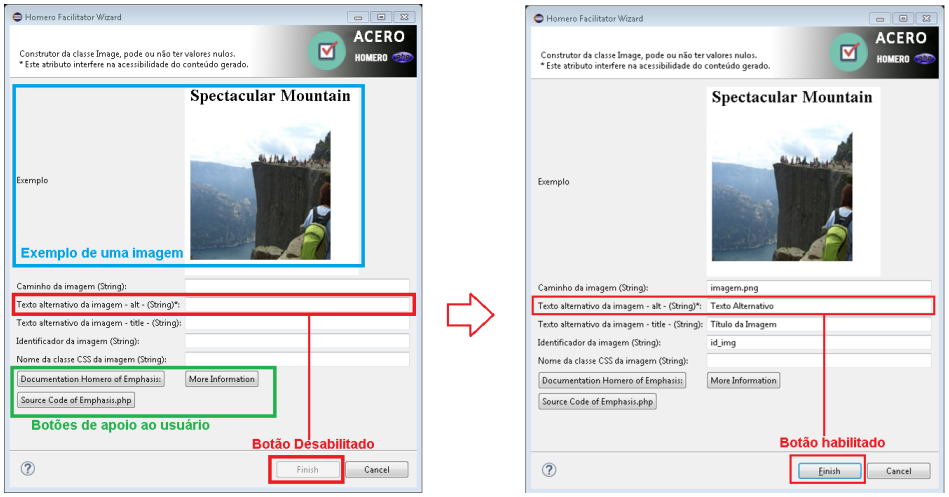

Figura A.11: *Wizard* do elemento Homero *Image* (Imagem)

Seguindo os passos anteriores, o usuário terá como saída um código similar ao apresentado na Figura A.12. Os campos não preenchidos são

[1] http://www.w3schools.com/

preenchidos com valor nulo (*null*), de acordo com a especificação trazida pelo *framework* Homero.

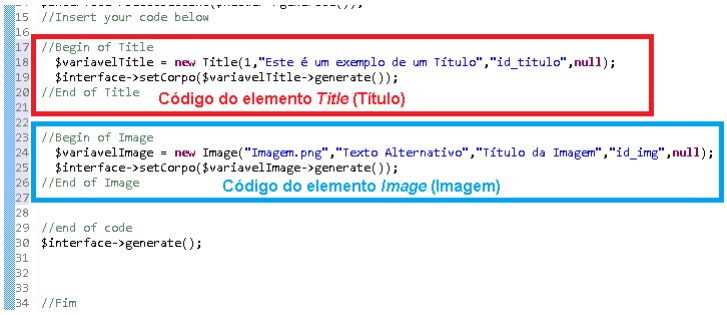

Figura A.12: Código gerado através das *wizards* de Título e Imagem

De forma geral a *Acero* possui como um de seus objetivos, facilitar a utilização do *framework Homero* no desenvolvimento de elementos acessíveis, evitando que o usuário consulte a todo momento a documentação e o código fonte do elemento no *framework Homero* para saber quais campos são obrigatórios e como devem ser preenchidos.

3. Fornecer documentação de utilização, métodos e campos necessários para utilização da *Homero* de maneira semi automática ao usuário: Caso o usuário queira adicionar elementos de uma maneira mais flexível, sem a utilização de *wizards* fornecidas pela *Acero*, pode utilizar o recurso *Gerador Rápido de Elementos Homero*, no menu *Acero*.

   A Figura A.13 apresenta a tela do recurso descrito anteriormente. É possível observar que a área de texto já apresenta o *template* básico para o funcionamento da *Homero*. Este recurso permite que um usuário que tenha conhecimento do *framework Homero* consiga inserir elementos mais rapidamente.

Por exemplo, para o usuário adicionar um *Button Image* (Botão Imagem) acessível ou caso queira conhecer os campos necessários para instanciar um elemento acessível do tipo botão imagem, seguirá a sequência apresentada na Figura A.14. Após pressionar o botão do elemento desejado, no exemplo *Button Image*, o usuário pode editar os campos necessários, depois copiar o trecho de código e colar no arquivo em que instanciará a interface com elementos acessíveis.

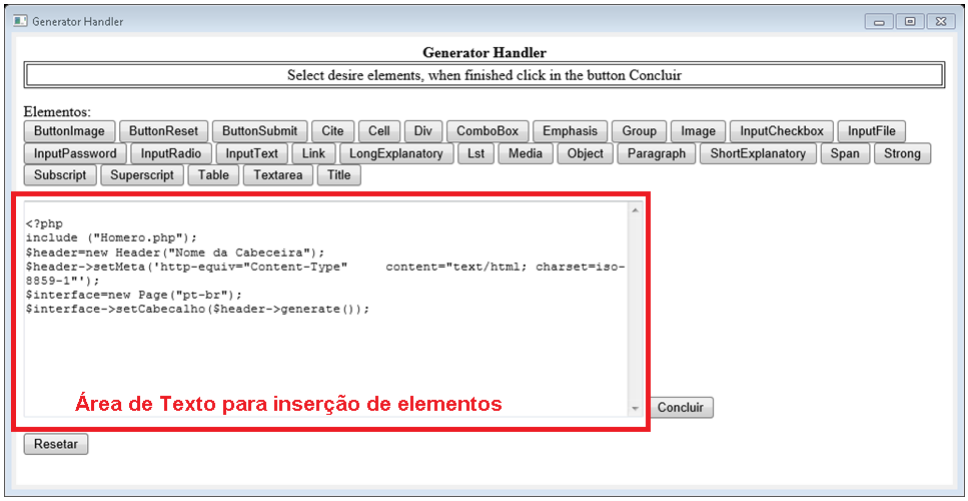

Figura A.13: Gerador rápido de elementos Homero

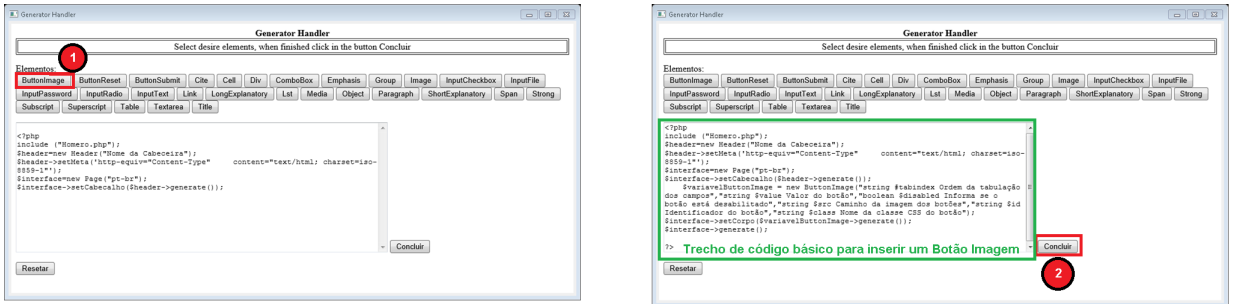

Figura A.14: Inserção do elemento *Button Image* no *Gerador rápido de elementos Homero*

A.2.2 Gerador de Conteúdo Acessível e Interpretador PHP

Esta funcionalidade da *Acero* tem por objetivo fornecer os recursos necessários para que o usuário execute seu código PHP dentro da IDE Eclipse. Desta forma, é possível executar um código em PHP dos elementos acessíveis fornecidos pelo *Homero* e obter um arquivo HTML da interface acessível.

A ferramenta *Acero* possui um *Console*, que se encontra na parte de *Janelas de Visualização*, conforme apresentado na Figura A.15. Denominado de *Acero Output*, esta janela apresenta como saída o código PHP interpretado e outros dados que serão mencionados posteriormente. Caso o ambiente de desenvolvimento não esteja visível esta janela, o usuário pode adicioná-la seguindo o *Menu Window* → *Show View* → *Other* → *Acero* → *Acero Output*.

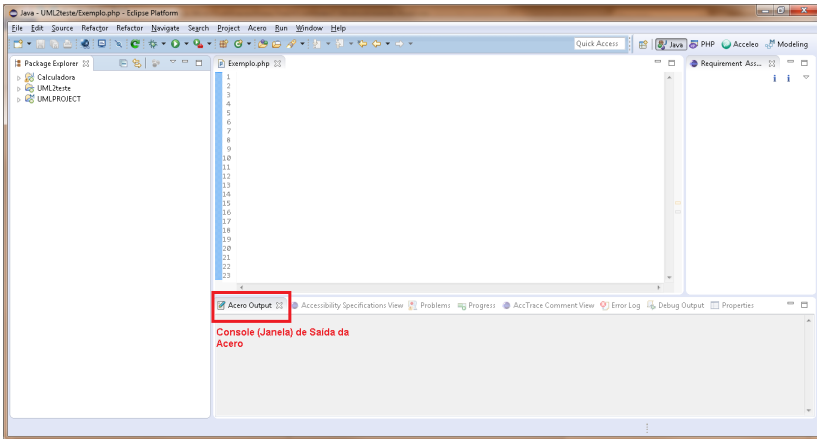

Figura A.15: Janela (View) *Acero Output* na IDE Eclipse

Um exemplo simples de utilização do interpretador PHP é apresentado na Figura A.16, onde é apresentado um código simples, de somar duas variáveis e apresentar o resultado na janela *Acero Output*. Para o usuário executar um código deve realizar o seguinte processo:

Depois de abrir o arquivo a ser executado na área de edição (duplo clique sobre o arquivo desejado), o usuário deve seguir ao *Menu Acero* → Executar no Servidor. Então, a saída será apresentada na Janela *Acero Output*.

De forma similar, é possível executar um código em PHP contendo o *template* e os elementos da *Homero* para obter um código Web acessível. A Figura A.17 apresenta a execução do código fornecido anteriormente pelas *wizards Title* e *Image* expostas na Figura A.12. Na parte destacada na figura como *Saída*, encontra-se o código fonte acessível em *HTML*.

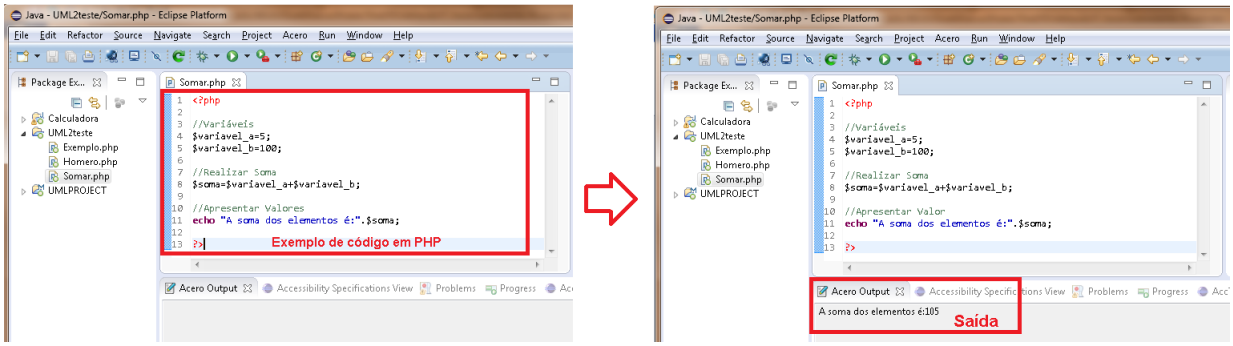

Figura A.16: Exemplo de um código em PHP e saída deste apresentado na Janela (View) *Acero Output* na IDE Eclipse

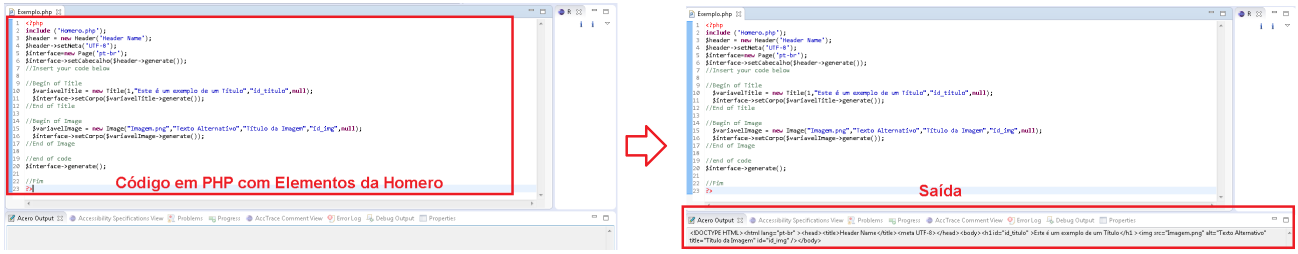

Figura A.17: Exemplo de um código em PHP com *template* e elementos acessíveis do *Homero*. A saída do código é apresentada na janela *Acero Output*

Caso o usuário queira criar um arquivo *HTML* com a interface acessível, identificar quais problemas em seu código afetam a acessibilidade e buscar se os arquivos declarados no código existem, pode-se utilizar outro recurso fornecido pela ferramenta *Acero*.

Uma vez aberto o arquivo em PHP que possui o *template* básico e os elementos da Homero (duplo clique sobre o arquivo), o usuário pode seguir o seguinte processo: *Menu Acero* → *Gerar Interface Acessível*. Com isso, será redirecionado a uma tela similar à apresentada na Figura A.18. Esta tela, possui alguns campos que devem ser preenchidos, o primeiro campo, o *File Name Output*, trata-se do nome do arquivo de saída, que deve possuir extensão *.html*. O segundo campo, o *User Error*, caso selecionado, apresenta na janela *Acero Output* os erros encontrados no código do usuário que afetam o funcionamento do *script*, por exemplo inserir uma imagem com um formato não suportado pelo interpretador HTML. O terceiro campo, o *User Warning*, caso selecionado, apresenta ao usuário através da janela *Acero Output* os erros encontrados no código que afetam na acessibilidade do conteúdo, por exemplo, a ausência de texto alternativo em uma imagem. O quarto campo, o *Search For Objects*, caso selecionado apresenta ao usuário, através da Janela *Acero Output* se os arquivos declarados no código realmente existem no computador do usuário. O último botão de opção, definido como *Show HTML file after compilation*, trata-se de apresentar ao usuário o conteúdo gerado, no navegador

interno da IDE Eclipse. Por fim, o texto *Server Status* indica se o servidor que realiza o processo de interpretação PHP encontra-se disponível ou não.

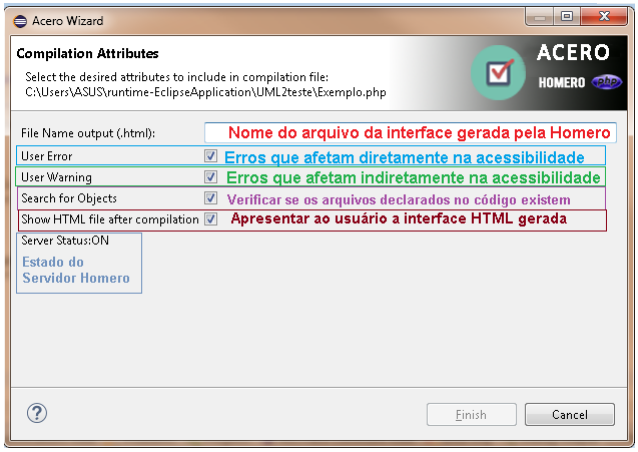

Figura A.18: Tela responsável por gerar interfaces acessíveis

A Figura A.19 apresenta a execução do código fornecido anteriormente pelas *wizards Title* e *Image* expostas na Figura A.12. Após serem definidos os campos da tela responsável por gerar a interface acessível, apresentada na Figura A.18 e o usuário pressionar o botão *Finish*, será então gerado um arquivo no formato *HTML* que representa a interface acessível. De acordo com os botões de opções escolhidos na tela para gerar conteúdos acessíveis, o usuário poderá visualizar a interface no navegador interno da *IDE Eclipse* e obter na Janela *Acero Output* o código fonte, erros e se os objetos declarados no código existem.

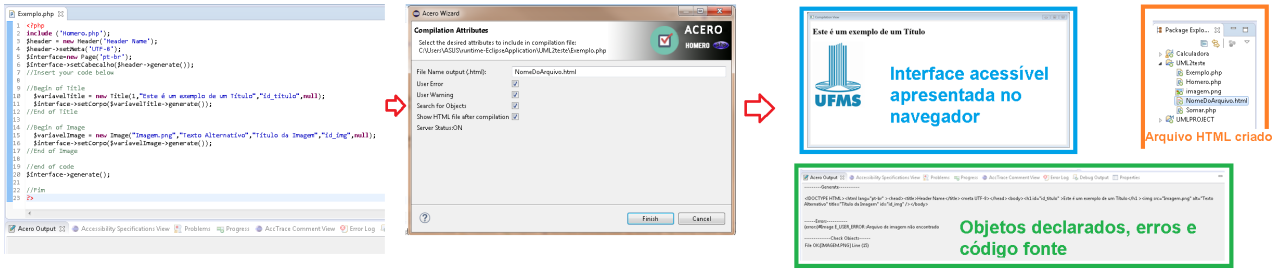

Figura A.19: Exemplo do processo para gerar uma interface acessível a partir de um código PHP e utilizando o *framework Homero*

Como observado, não é exigido que o usuário realize alterações sobre o código, ou seja, códigos PHP obtidos das *wizards* da *Acero* são interpretados diretamente pelo servidor onde se encontra o *framework Homero*, possibilitando a criação de interfaces acessíveis de uma maneira mais simples e veloz.

A.2.3 Avaliar diretrizes de acessibilidade dentro do ambiente de desenvolvimento

Como complemento dos recursos descritos anteriormente, a *Acero* fornece a também a possibilidade do usuário avaliar diretamente dentro da *IDE Eclipse* se seu código obedece às diretrizes de acessibilidade escolhida.

Uma vez criado um arquivo *HTML* com a interface acessível, é possível submeter o código à avaliação com base nas diretrizes de acessibilidade definida e obter um arquivo contendo resultado da avaliação. A *Acero* possui suporte a avaliação das seguintes diretrizes:

- *BITV* 1.0 (*Level* 2);
- *Section* 508;
- *Stanca Act*;
- *WCAG* 1.0 (Nível A, AA e AAA)
- *WCAG* 2.0 (Nível A, AA e AAA).

Para o usuário avaliar seu código, deve seguir o seguinte processo: Abrir o arquivo HMTL da interface gerada na IDE Eclipse → *Menu Acero* → *Verificador de Diretrizes de Acessibilidade*. Seguindo o processo descrito anteriormente, o usuário será direcionado a tela apresentada na Figura A.20. Nesta tela mencionada, são apresentados os campos que devem ser preenchidos. O primeiro campo, trata-se do nome do arquivo resultante da avaliação de acessibilidade. O segundo campo, trata-se de um botão de opção que, caso selecionado, a *Acero* inclua no resultado de avaliação, o código fonte da interface verificada. O terceiro campo, apresenta um botão de opção que indica se o usuário deseja incluir na análise de acessibilidade a avaliação da ferramenta *Acess Monitor*, entretanto, esta ferramenta suporta unicamente as diretrizes *WCAG 1.0 e 2.0*. Por fim, são apresentadas as opções para a escolha da diretriz que será utilizada na análise da interface.

A Figura A.21 apresenta um exemplo de avaliação de diretrizes de acessibilidade sobre uma interface gerada pelo *framework Homero*, através do código fornecido pelas *wizards Title* e *Image* na Acero. Após a análise, a *Acero* cria um arquivo de saída com a concatenação dos resultados de avaliação das diretrizes de acessibilidade das ferramentas (*Achecker* e *AcessMonitor*). Caso selecionada, é aberto este arquivo e apresentado o resultado ao usuário em um

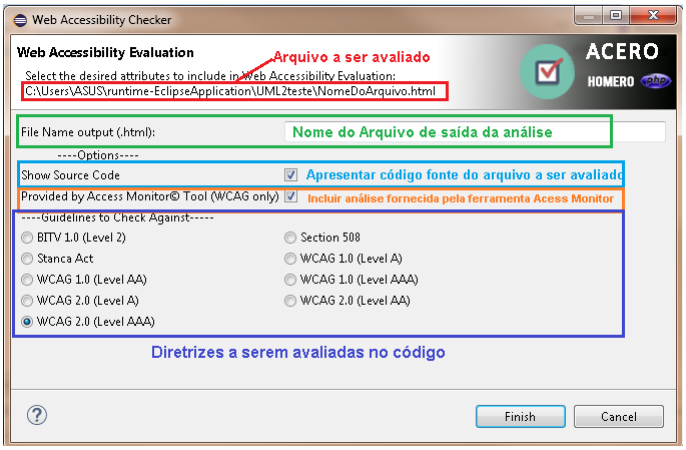

Figura A.20: Tela da ferramenta *Acero* responsável da avaliação de diretrizes de acessibilidade em um código

navegador interno da IDE Eclipse. Torna-se importante destacar que na Figura A.21 a ferramenta *AcessMonitor* encontrou um erro nas diretrizes WCAG 2.0 no nível AA, que refere-se a ausência de pelo menos um *link* na página para informações adicionais aos usuários que as necessitem.

A.2.4 Rastreabilidade

Como descrito anteriormente, um dos objetivos da *Acero* é ampliar o escopo da ferramenta *AccTrace*. A *AccTrace* entrega ao desenvolvedor, através da rastreabilidade dos requisitos de acessibilidade e comentários no código-fonte, informações úteis para implementação de tais requisitos. Além disso, é possível gerar uma tabela de rastreabilidade que relaciona os requisitos, artefatos UML e técnicas de implementação. A extensão da matriz de rastreabilidade suportada pela AccTrace é *.ods*. A *Acero* ampliou a utilização para suportar a extensão *.pdf*.

A Figura A.22 apresenta um exemplo de criação de um arquivo da matriz de rastreabilidade na extensão *.pdf*. Após o relacionamento dos requisitos, artefatos UML e técnicas de implementação é possível gerar a matriz de rastreabilidade, através da seleção no Projeto Atual → *New* → *Other* → *Traceability matrix file wizard*.

O suporte a extensão *PDF* oferece flexibilidade ao usuário, para que escolha qual formato é mais apropriado em seu contexto. A escolha da extensão ocorre no final do nome do arquivo da *wizard* de criação, NOME_DO_ARQUIVO.*ods* para a extensão *ODS* ou NOME_DO_ARQUIVO.*pdf* para a extensão *PDF*.

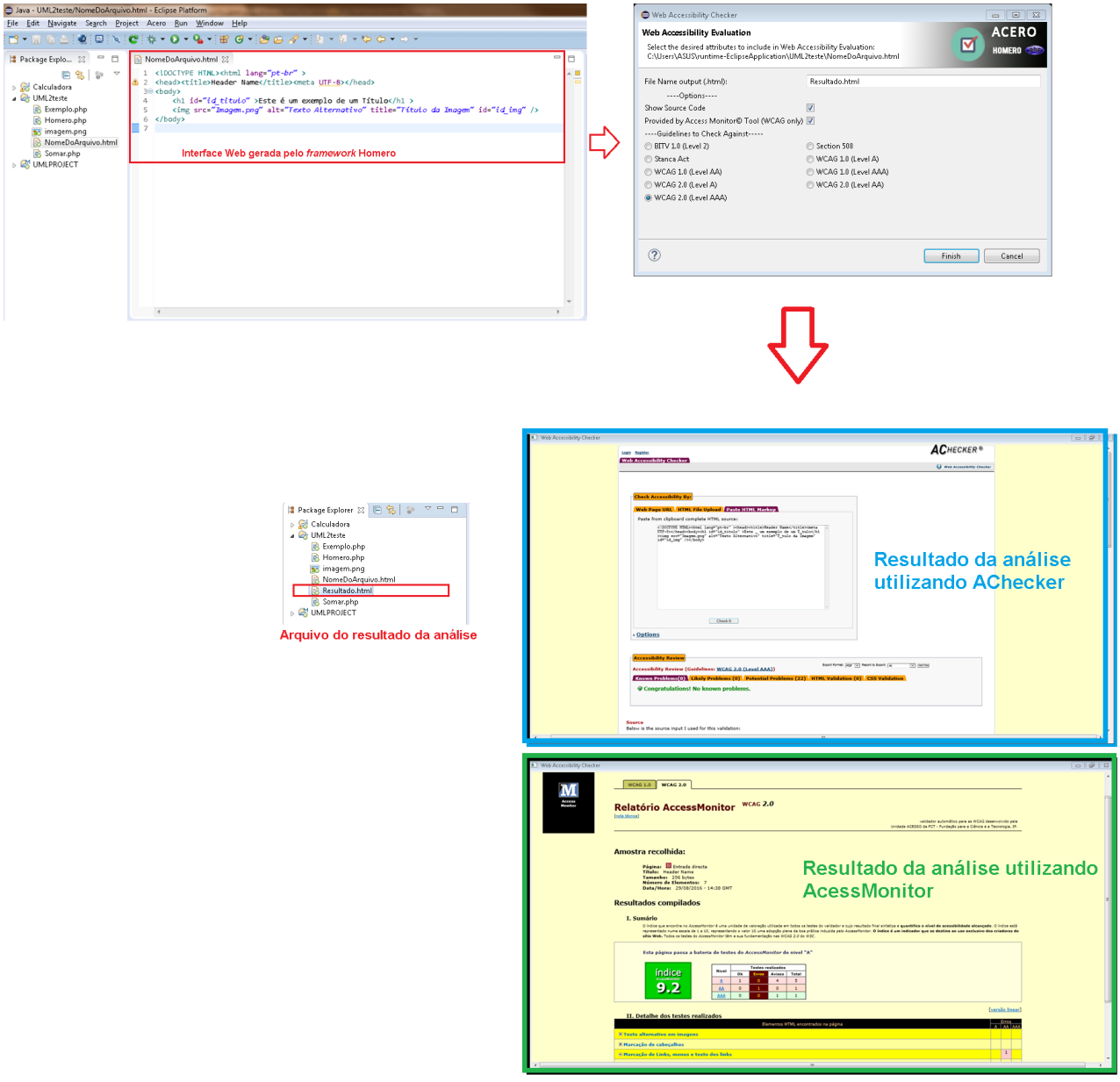

Figura A.21: Exemplo de avaliação de diretrizes de acessibilidade sobre um código HTML

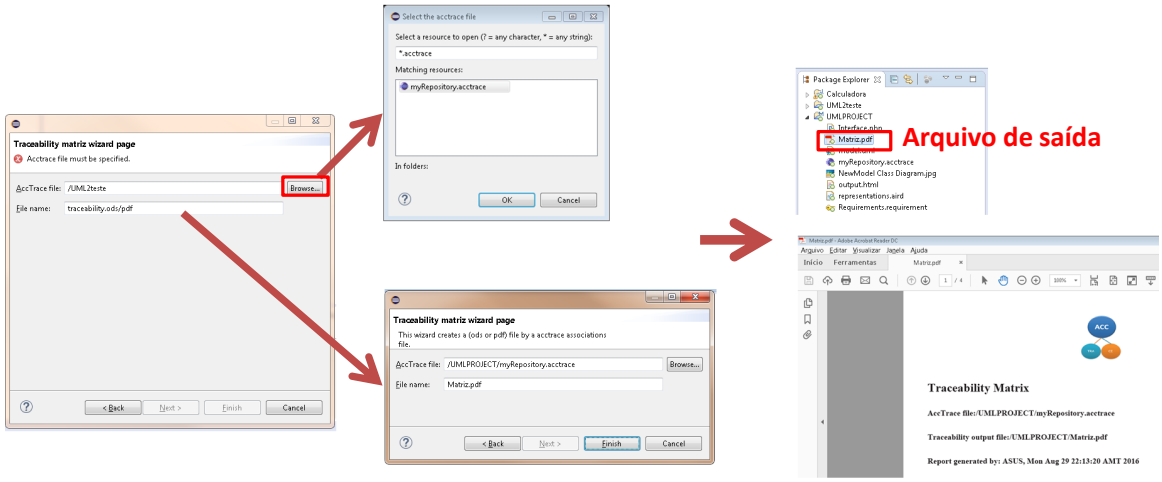

Figura A.22: Exemplo de criação de uma matriz de rastreabilidade no formado PDF

Além disso, a *Acero* possibilita verificar e corrigir problemas de consistência entre o arquivo de relacionamento de requisitos, artefatos UML e técnicas de implementação com os comentários presentes no código fonte. Dessa forma, é possível fazer a rastreabilidade reversa, isso é, verificar se os comentários presentes no código fonte refletem ao definido pelo relacionamento do arquivo *.acctrace*. Além disso, também é verificado a consistência entre o arquivo de relacionamento com os arquivos de requisitos e artefatos *UML*.

Para utilizar o recurso de rastreabilidade reversa, basta o usuário pressionar o botão direito do *mouse* sobre o arquivo de relacionamento *.acctrace* e escolher a opção *Consistency Check Acctrace*. A Figura A.23 apresenta a sequência de passos e a janela principal responsável pela rastreabilidade reversa. Nesta janela é possível escolher em qual projeto serão verificadas a consistência dos códigos fonte em relação ao arquivo de relacionamento *.acctrace* e a consistência do arquivo de relacionamento *.acctrace* com os arquivos de requisitos e artefatos UML. Caso o usuário queira verificar somente a consistência do arquivo de relacionamento *.acctrace* com os arquivos de requisitos e artefatos UML, sem a análise de códigos do projeto, é possível marcar a opção *Verificar Somente Arquivo AccTrace*.

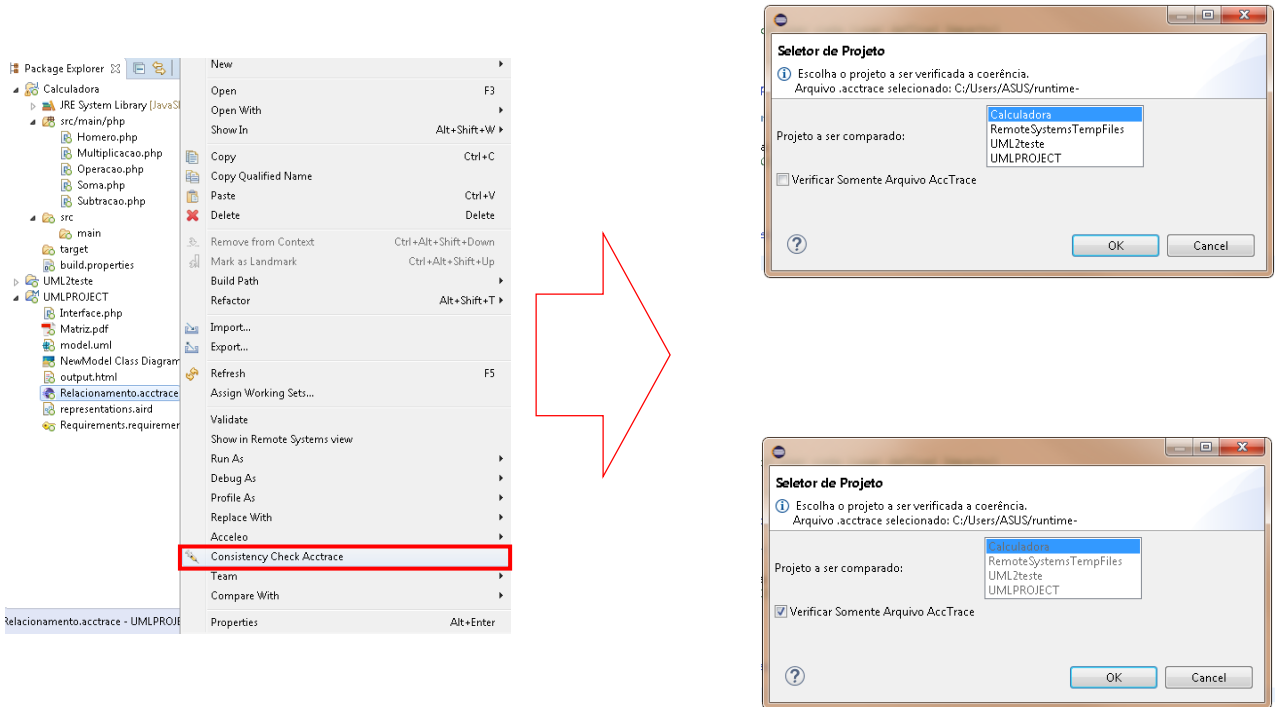

Figura A.23: Exemplo de utilização da rastreabilidade reversa

A Figura A.24 apresenta um exemplo de utilização da rastreabilidade reversa. A classe *Soma*, do projeto Calculadora, foi gerada automaticamente sem a inclusão de comentários que auxiliam na implementação de acessibilidade, pois, não se havia criado referência no arquivo de relacionamento *.acctrace*. Após a criação de uma referência entre requisitos e técnicas de implementação de acessibilidade relativo a classe no arquivo de relacionamento *.acctrace*, a ferramenta *Acero* possibilita a inclusão do comentário referente ao relacionamento na classe Soma, de maneira automática. De outra maneira o usuário teria que gerar novamente todo o projeto ou inserir manualmente com base na matriz de rastreabilidade e no arquivo de associação *.acctrace*.

A grande vantagem trazida pela Acero, na rastreabilidade reversa é de ofertar ao usuário a possibilidade de manter seu código fonte e arquivo de relacionamento *.acctrace* coerentes, complementando a utilização da matriz de acessibilidade.

A.2.5 Geração automática de Classes PHP

A *Acero* também permite que o usuário crie esqueletos de classes PHP com comentários de apoio a implementação de acessibilidade de forma automática, a partir de um arquivo de relacionamento *.acctrace* e de um arquivo de artefatos UML *.uml*. Dessa forma, amplia-se o escopo da *AccTrace*, que suporta apenas a criação de classes *Java*.

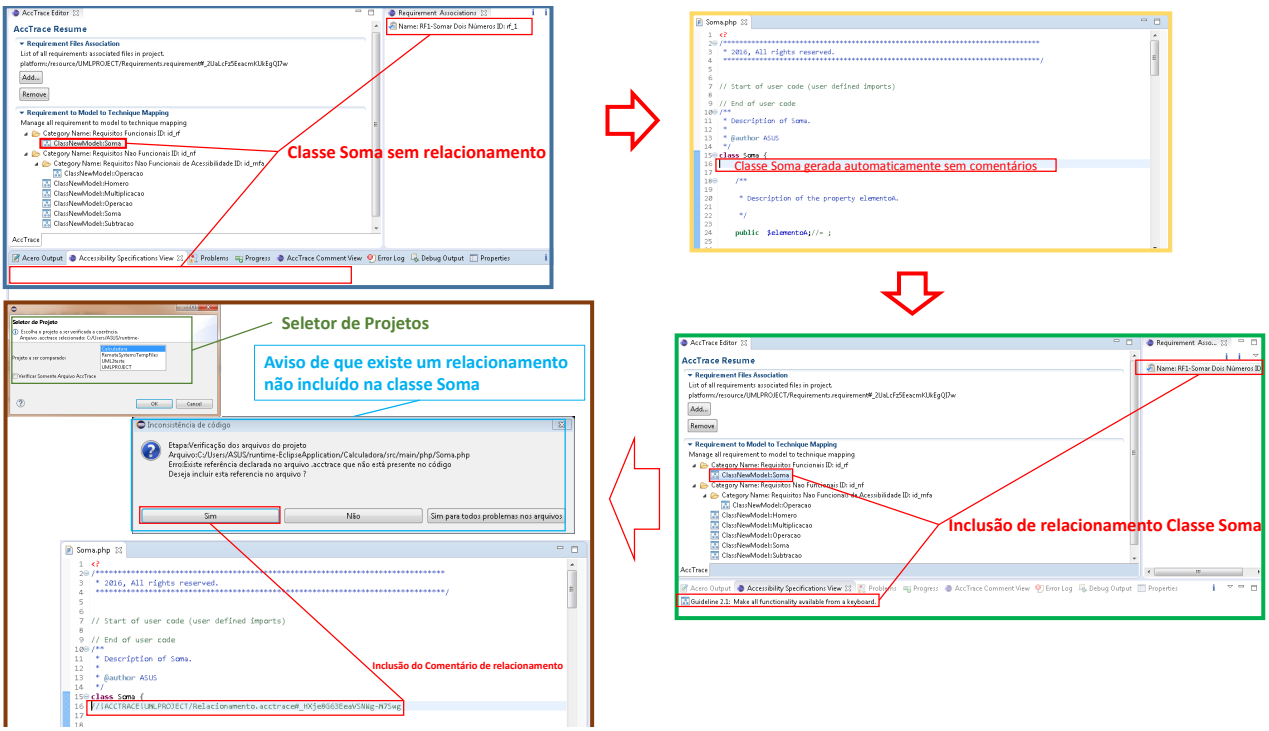

Figura A.24: Exemplo de utilização da rastreabilidade reversa

A Figura A.25 apresenta um exemplo de processo para criar os esqueletos das classes PHP de forma automática. Inicialmente, escolhido os arquivos de artefatos UML e do relacionamento *.acctrace*, que representa o relacionamento entre os requisitos, artefatos UML e técnicas de implementação, é possível construir as classes PHP através dos seguintes passos:

Pressionando o botão direito do mouse sobre o arquivo de artefatos UML, de extensão *.uml*, no *Explorador de Projetos*, após *Run As* → *Run Configurations* → duplo clique sobre *Acceleo UML2 to PHP Generation* → Definir os arquivos *.uml* dos artefatos e o arquivo *.acctrace* do relacionamento → *Apply* → *Run*.

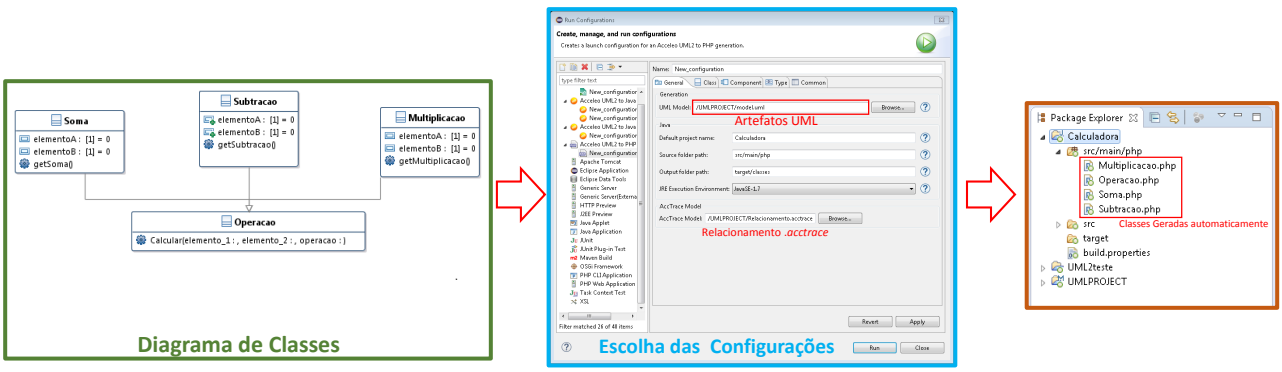

Figura A.25: Exemplo de geração de código PHP de forma automática

A.2.6 Recuperação de Comentários PHP, C, C++, C#, PHP, JavaScript

A *Acero* estende a ferramenta *AccTrace*, permitindo que o usuário recupere o significado dos comentários de relacionamento *.acctrace* nas linguagens que apresentam a expressão regular de comentários *//**, englobando assim as linguagens *Java*, *PHP*, *C*, *C++*, C#, *PHP* e *JavaScript*.

Para o usuário utilizar este recurso, basta ter um código fonte aberto na IDE Eclipse com os comentários de apoio a implementação de acessibilidade e pressionar o botão destacado na Figura A.26, na parte superior da *IDE Eclipse*. De mesma forma mais simples, a recuperação dos comentários também pode ser realizada através do atalho de teclado *CTRL+6*.

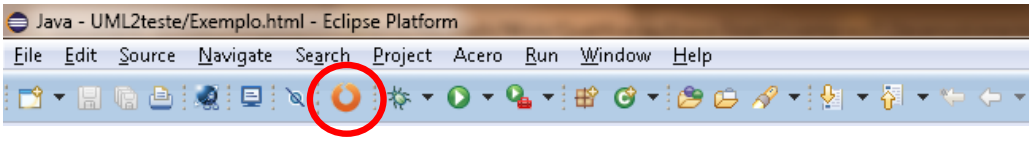

Figura A.26: Botão responsável por recuperar os comentários de acessibilidade no código. A recuperação também pode ser realizada através do atalho do teclado *CTRL+6*

A Figura A.27 apresenta um exemplo de recuperação de comentário em um código fonte em *PHP*. Com o código fonte aberto na parte de edição da *IDE*, basta pressionar o atalho de teclado *CTRL+6*, ou acessar pelo botão indicado no canto superior, com isso, será apresentado na Janela (*View*) *AccTrace Comment View*, localizada na figura no canto inferior central, o significado do comentário que visa facilitar ao usuário a implementação de acessibilidade.

A.2.7 Preditor de erros de acessibilidade no código

A *Acero* possui duas partes que apoiam a realização de predição no código fonte. A primeira parte é buscar se os arquivos declarados no código fonte existem. A segunda parte é buscar por palavras chaves de elementos no código do usuário e fornecer com base nestas palavras encontradas, os principais erros que afetam a acessibilidade do elemento.

Para o usuário utilizar este recurso da Acero, deve seguir o seguinte processo: *Abrir o arquivo que será analisado na IDE Eclipse* → *Menu Acero* → *Usar*

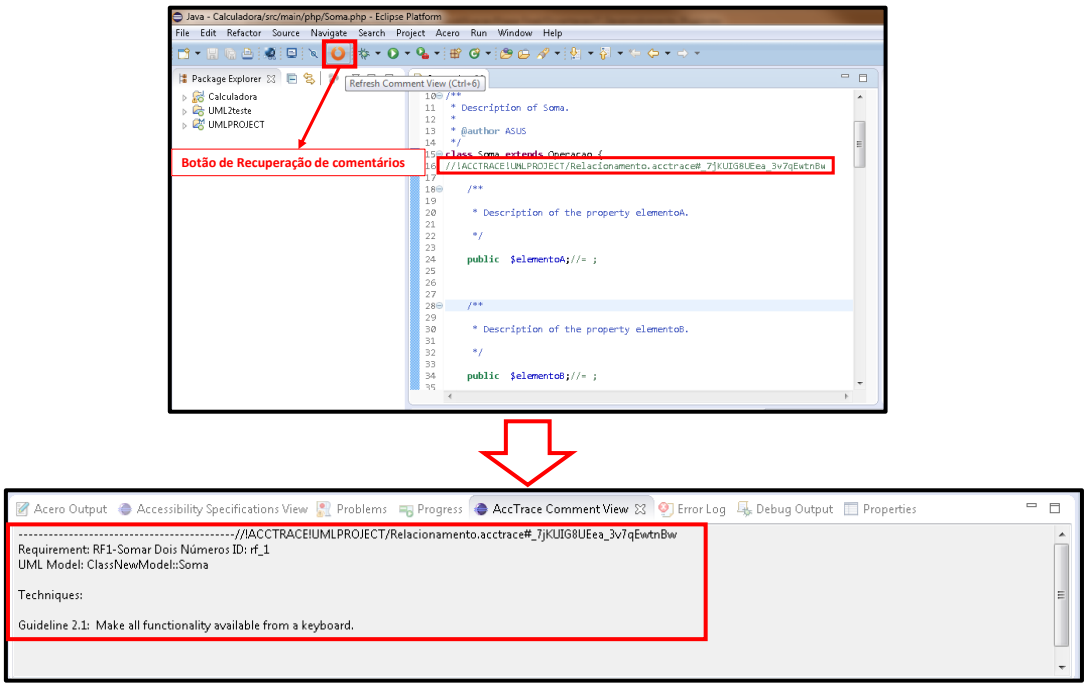

Figura A.27: Exemplo de recuperação de um comentário de apoio a implementação de acessibilidade em um código PHP

Predição de Acessibilidade no Código. Após o usuário seguir os passos descritos anteriormente, será apresentada a janela mostrada na Figura A.28. O primeiro campo de opção, o *Search for Objects*, possibilita a busca pelos arquivos selecionados no código, o segundo campo, o *Server Status*, apresenta o status do servidor que fornece as informações de acessibilidade. Ao pressionar o botão *Finish*, as informações serão apresentadas na Janela (View) *Acero Output*.

A Figura A.29 apresenta um exemplo de utilização do preditor da *Acero* sobre um código HTML. É possível notar, que no local destinado a apresentação dos objetos declarados em código, é mostrado que o elemento *imagem.png*, na linha 7 existe (*OK*). Na outra parte, destinada a predição de elementos, pode-se notar que para o elemento *header (<head)*, na linha 3, existem três erros que podem afetar na acessibilidade. Dois dos erros, denominados de E_USER_ERROR, são erros fatais em que o *script* pode não funcionar se cometidos, para o elemento *<head>* caso o usuário declare um arquivo *JS* ou um *CSS* que não exista no sistema de arquivos do usuário o *script* não funcionará corretamente. De outro modo, erros denominados de E_USER_WARNING, são erros que prejudicam a acessibilidade do código, no caso da predição do elemento *image*, na linha 7, caso o desenvolvedor não adicione um texto auxiliar na imagem, o *script* funcionará, entretanto, a imagem não terá acessibilidade.

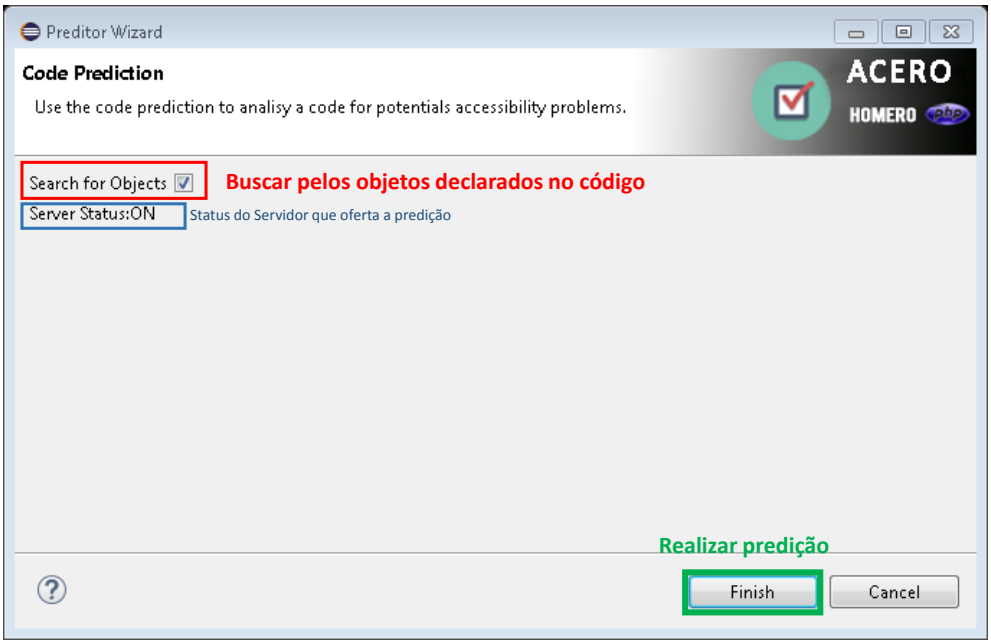

Figura A.28: Janela de predição da *Acero* apresentada ao usuário

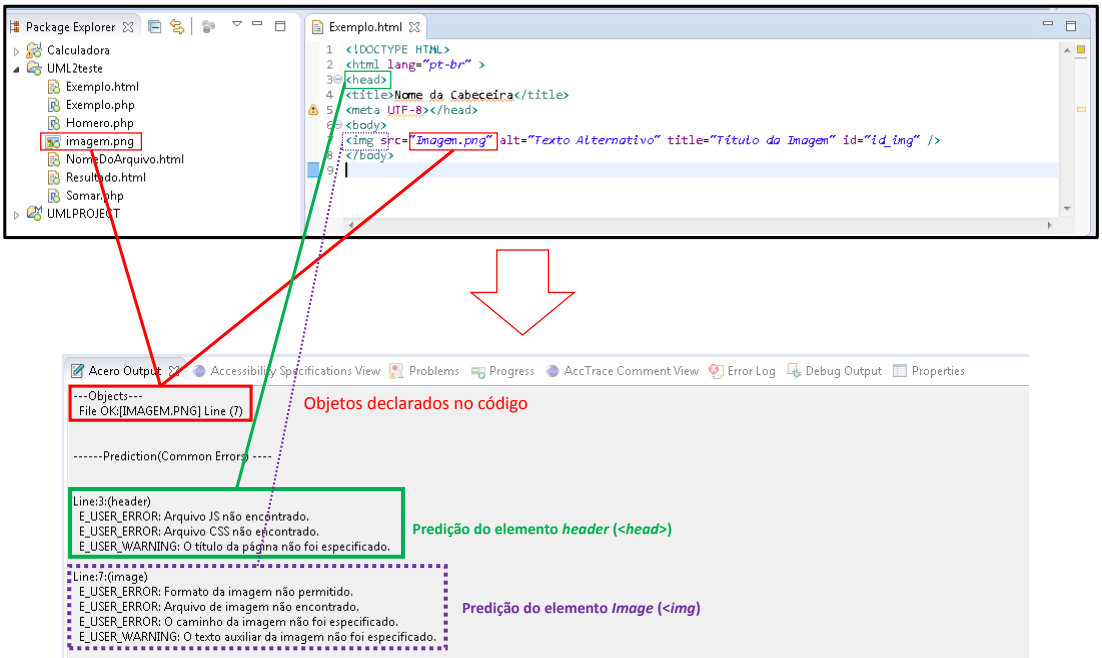

Figura A.29: Exemplo de utilização de predição sobre o código da ferramenta Acero

A.2.8 Integrações Experimentais

Foi buscado inserir outras ferramentas, que não vieram de indicações de trabalhos anteriores do grupo de pesquisa da UFMS, com o objetivo de estender a utilização da Acero. Dessa forma, foram integradas a *Acero* duas ferramentas no âmbito de apoio a análise de acessibilidade com o objetivo de incrementar a solução de integração proposta. A primeira ferramenta é a *Colour Contrast Analyser*[2], indicado pelo grupo de acessibilidade do W3C, a ferramenta permite que o desenvolvedor avalie se os contrastes entre as cores aplicadas em uma página Web estão ou não acessíveis, podendo inclusive simular problemas visuais sobre páginas Web, por exemplo na visão de uma pessoa com catarata e com daltonismo, no entanto, a ferramenta está somente disponível na plataforma *Windows*. A segunda ferramenta integrada a *Acero* é a *Total Validator Basic*[3] que permite a verificação de acessibilidade em páginas Web em modo *offline*, isso é, o usuário não necessita estar conectado a internet para avaliar a acessibilidade de seu código fonte.

Para que o usuário possa utilizar a ferramenta *Colour Contrast Analyser*, basta abrir o arquivo de interface gerado automaticamente pela *Acero* e seguir os seguintes passos: *Menu Acero* → *Colour Contrast Analyser*.

A Figura A.30 apresenta um exemplo de utilização da ferramenta *Colour Contrast Analyser*, que avalia o contraste de uma página Web. Com a interface Web aberta, o usuário seleciona um *pixel* da região de primeiro plano (*Foreground*) através da botão com o ícone de conta gotas, depois o usuário seleciona um pixel do local de segundo plano (*Backgroud*) através do outro botão de conta gotas. Após a seleção das cores será apresentado ao usuário o resultado obtido, caso o constaste da página esteja correto, será apresentado o texto *Pass()*, caso contrário será apresentado o texto *Fail()*. Na figura apresentada, a solução foi aprovada tanto para textos normais como para textos largos, pois a cor de fundo é branca e a cor do escrito é preta, gerando-se um contraste aceitável de acessibilidade no nível AA e AAA.

Com mencionado, outra funcionalidade útil da ferramenta *Colour Contrast Analyser* é a possibilidade de simular problemas visuais sobre uma página Web. Esta funcionalidade é útil, visto que muitas vezes o desenvolvedor não possui conhecimento das necessidades visuais de seus clientes. A Figura A.31

[2] https://www.paciellogroup.com/resources/contrastanalyser/
[3] https://www.totalvalidator.com/tools/

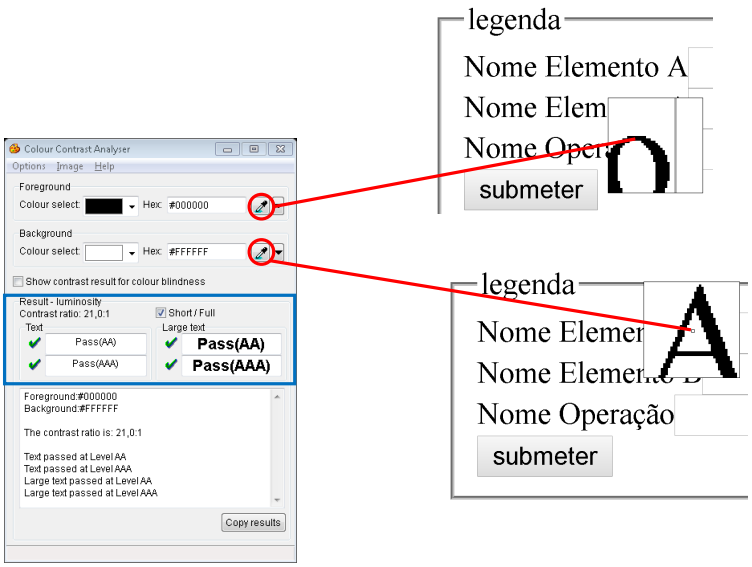

Figura A.30: Exemplo de avaliação de contraste na ferramenta *Colour Contrast Analyser*

apresenta um exemplo de utilização da ferramenta para simulação de diferentes problemas visuais. Neste exemplo, cada imagem possui doze lápis de cores, em cores distintas, sendo simuladas a distintos problemas visuais. É possível observar que na visão normal, pode-se distingui-los claramente suas cores e a quantidade de lápis, entretanto, para pessoas com problemas visuais, como por exemplo àquelas com Protanopia [4], não é possível distinguir as cores dos lápis. Já indivíduos com Catarata, torna-se difícil contestar a quantidade de lápis, principalmente quando a imagem é pequena, pois, quanto menor a imagem é, maior a suavização e consequentemente menor a compreensão das informações presentes na imagem.

A outra ferramenta mencionada, é a *Total Validator Basic*, que permite a avaliação das diretrizes de acessibilidade no modo *offline*. Para o usuário da *Acero* utilizar a ferramenta *Total Validator Basic* dentro do ambiente de desenvolvimento, basta seguir ao *Menu Acero* → *Total Validator Basic*. Ao seguir dos passos anteriores, o usuário será direcionado a uma janela como apresentada na Figura A.32. A ferramenta *Total Validator Basic* apresenta uma interface de simples utilização, a primeira caixa de texto, a *What to check*, representa qual arquivo de código será validado com as diretrizes de acessibilidade, o botão *Browse* permite a escolhe do arquivo. No campo *Accessibility validation*, permite que o usuário escolha por qual diretriz de acessibilidade, a ferramenta

[4]Ausência na retina de receptores da cor vermelha - http://www.color-blindness.com/protanopia-red-green-color-blindness/

Figura A.31: Exemplo de utilização da ferramenta *Colour Contrast Analyser* para simulação de diferentes problemas visuais sobre uma imagem contendo doze lápis de cores

validará o código. Após definidos todos os campos, o botão *Validate*, localizado na canto inferior esquerdo, inicia o processo de validação. A saída contendo o resultado da análise será apresentado no navegador padrão do usuário, sob a forma de um arquivo *HTML*.

A Figura A.33 apresenta um exemplo de validação de acessibilidade em um código fonte, com as diretrizes WCAG 2.0 AAA, realizada *offline* por meio da ferramenta *Total Validator Basic*. Pode-se notar que o código avaliado apresenta, segundo a ferramenta, três erros na linha 23. O erro *E604* refere-se em não existir a *tag* de fim do formulário, o erro *E630* diz que o campo *nome* e *id* devem possuir o mesmo nome. Por fim, o erro *E872* aponta que não existe um botão de submissão no formulário.

A.2.9 Configuração de conexão da Acero

Como citado anteriormente a *Acero* realiza a comunicação com o *framework* Homero através de um servidor PHP. Para que a *Acero* comunique com o *framework Homero*, é necessário configurar o endereço, ou URL, do servidor. Para

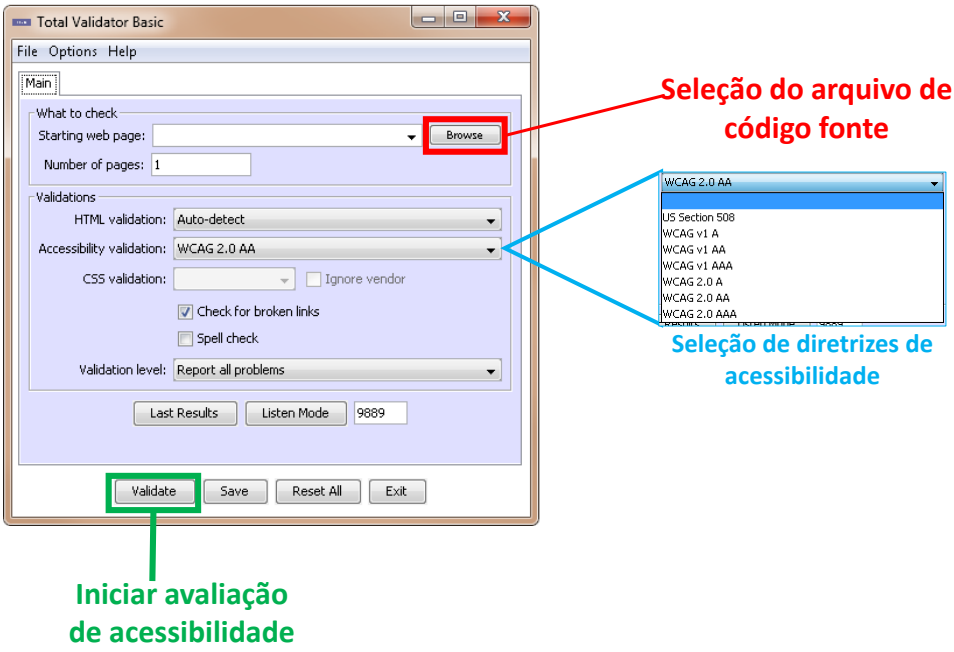

Figura A.32: Janela de utilização da ferramenta *Total Validator Basic*

Figura A.33: Exemplo de validação de acessibilidade em um código fonte, com as diretrizes WCAG 2.0 AAA, realizada por meio da ferramenta *Total Validator Basic*

isso, basta o usuário seguir o *Menu Acero* → *Configurar Endereço do Servidor* →. Ao seguir os passos descritos anteriormente o usuário será direcionado a janela de configuração apresentada na Figura A.34. Nesta janela é definido a *url* do servidor. O formato do endereço possui o seguinte formato:

- *http://IP_do_Servidor/Homero/Server.php*

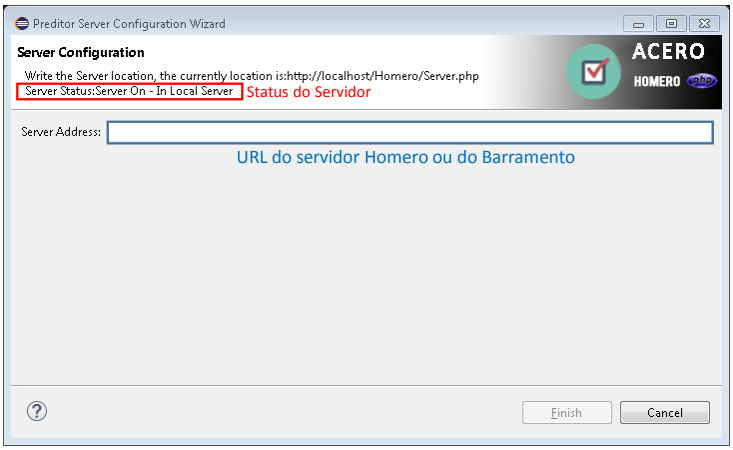

Figura A.34: Janela de configuração do endereço do servidor

Após o usuário definir a URL do servidor como na janela apresentada na Figura A.34 e pressionar o botão *Finish*, a *Acero* apresenta se a conexão com o servidor foi bem ou mal sucedida. A Figura A.35 apresenta um exemplo de configuração da *Acero* com o servidor bem sucedida. É possível notar que é apresentada uma mensagem *Successful!*, que refere-se a uma comunicação bem sucedida com o servidor.

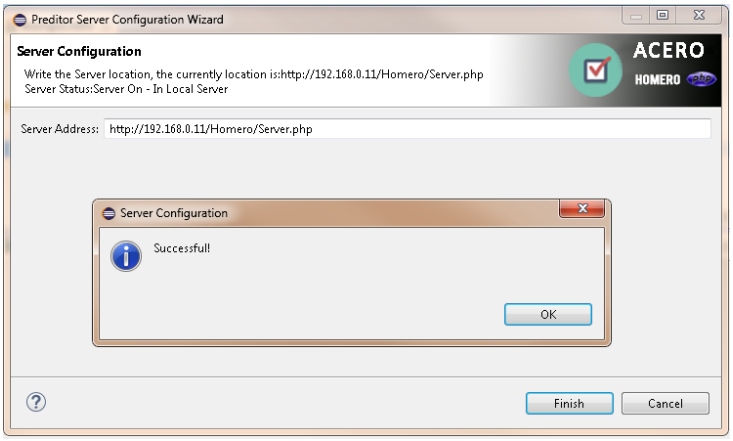

Figura A.35: Exemplo de uma configuração bem sucedida para a comunicação direta com o *framework Homero*