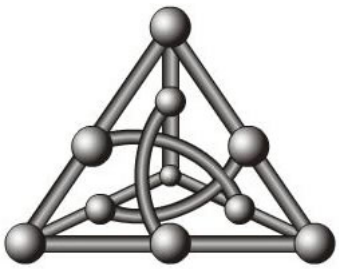
UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL

ACESSIBILIDADE NAS FASES DE ENGENHARIA DE REQUISITOS, PROJETO E CODIFICAÇÃO DE SOFTWARE: UMA FERRAMENTA DE APOIO

Rodrigo Gonçalves de Branco

Dissertação de Mestrado em Ciência da Computação
Área de Concentração - Engenharia de Software

Orientação: Prof^{a}. Dr^{a}. Débora Maria Barroso Paiva



Campo Grande
2013

ACESSIBILIDADE NAS FASES DE ENGENHARIA DE REQUISITOS, PROJETO E CODIFICAÇÃO DE SOFTWARE: UMA FERRAMENTA DE APOIO

Dissertação apresentada como requisito para obtenção do grau de Mestre em Ciência da Computação no curso de Pós-Graduação em Ciência da Computação, Faculdade de Computação, Fundação Universidade Federal de Mato Grosso do Sul.

Orientadora: Prof^a. Dr^a. Débora Maria Barroso Paiva

Campo Grande, 09 de Setembro de 2013.

Agradecimentos

Aos meus pais, que, apesar de nunca terem entendido completamente o que faço em minha área profissional, sempre me apoiaram e continuam me apoiando. Agradeço por me mostrarem desde cedo o valor dos estudos, e por serem exemplos a qual pretendo continuar seguindo.

À minha esposa Juliana, que, no começo deste trabalho era apenas namorada, mas não menos importante por este motivo. Agradeço pela motivação excepcional que sempre me forneceu, por me tolerar e compartilhar os momentos de adversidade que surgiram nesta caminhada, bem como também os momentos felizes. E, principalmente, pelo apoio em todas as decisões que tomei até agora.

Ao meu irmão, Leandro, por ser o companheiro e camarada que ele sempre foi, sempre disposto a me acompanhar, seja lá qual fosse o programa. Agradeço também por ser esta pessoa de extrema paciência, sempre calmo, independente da situação envolvida.

À minha professora e orientadora, professora Débora, por sempre acreditar em meu trabalho. Agradeço pela paciência, orientações e conselhos oferecidos durante essa longa trajetória, me ajudando desde os tempos de minha graduação, não me abandonando nem mesmo quando precisei me mudar para uma cidade muito distante como Porto Velho, me incentivando sempre a continuar em minha jornada no programa de Mestrado.

Ao meu sobrinho Enzo, apenas por existir, e alegrar minha vida com seus sorrisos, sua voz e suas brincadeiras.

Aos meus amigos e colegas de faculdade, que sempre ajudaram a deixar minha vida mais divertida, fazendo-me esquecer, mesmo que por pouco tempo, dos problemas cotidianos. Agradeço também aos meus amigos que também compartilharam de momentos ruins da minha vida e ainda assim estavam lá para me apoiar e consolar.

A todos os meus professores da FACOM e do programa de Mestrado, profissionais que tenho orgulho em dizer que foram meus tutores, cujos ensinamentos carrego comigo até hoje e tento compartilhar com os demais.

Aos meus colegas de trabalho (TJ-MS, IFMS, TRT-RO), por me mostrar a imensidão de opiniões, técnicas e abordagens encontradas em nossa área de atuação, assim como fornecer soluções para os problemas que encontramos.

A todos que participaram direta ou indiretamente para o desenvolvimento deste trabalho.

Resumo

Fornecer produtos acessíveis deixou de ser um diferencial de determinadas empresas. Acessibilidade, nos dias atuais, é um requisito fundamental de qualquer solução desenvolvida, indicando principalmente respeito e cumplicidade com os clientes. Essa afirmação é especialmente verdadeira para os produtos desenvolvidos para a *Internet*, porta de acesso para toda a intercomunicação mundial. A *Internet* se mostrou a tecnologia mais rápida e barata de aquisição de informação, levando tecnologias legadas (serviços bancários, por exemplo) a se adaptarem de forma que pessoas com dificuldades permanentes ou momentâneas consigam interagir com a sociedade. Contudo, fornecer um produto acessível nem sempre é uma tarefa fácil. Além de diversas classes diferentes de deficiências e dificuldades (o que acarreta problemas de acessibilidade diferentes), a falta de treinamento e experiência na área faz com que desenvolvedores cometam erros em vários aspectos, resultando num produto inacessível. Os modelos de processos e *frameworks* de desenvolvimento de *software* ainda não se adaptaram de forma consistente e homogênea, em relação a acessibilidade na fábrica de *software*. A área de Tecnologia da Informação está passando por uma fase de transição entre o HTML 4 e XHTML para o HTML 5, que, entre outras coisas, pretende enfatizar a *web* semântica e tratar dos problemas específicos de acessibilidade. Por fim, as ferramentas disponíveis aos desenvolvedores não conseguem, de maneira eficaz, auxiliar efetivamente os desenvolvedores a entregarem um produto acessível. Neste trabalho considera-se que os requisitos de acessibilidade devem ser levados em conta durante todas as fases do processo de desenvolvimento de *software*, ou seja, devem evoluir desde a fase de análise de requisitos até a fase de teste de *software* para que se obtenha acessibilidade como um atributo de qualidade do produto final de *software*. Assim, buscou-se, sobretudo, criar uma abordagem que pudesse promover a rastreabilidade dos requisitos de acessibilidade desde sua concepção até a fase de codificação. Esta abordagem associou requisitos, modelos UML e técnicas de implementação de acessibilidade, mapeadas em uma ontologia de acessibilidade. Além disso, foi desenvolvido um *plugin* para o *Eclipse* que promoveu a associação das técnicas de implementação de acessibilidade e da matriz de rastreabilidade. Foi criada uma prova de conceito com a proposta de verificar se os objetivos do trabalho foram alcançados. O trabalho demonstrou que é possível realizar, de forma automática, o rastreamento dos requisitos de acessibilidade bem como suas técnicas de implementação, desde a Fase de Engenharia de Requisitos até a Fase de Codificação.

Palavras-Chave: *acessibilidade, rastreabilidade de requisitos, processo de desenvolvimento de software, ferramenta CASE.*

Abstract

Providing accessible products has recently left to be a differential feature of certain companies. Accessibility, today, is a fundamental requirement of any developed solution, indicating primarily respect and care to customers. This statement is especially true for products designed to the Internet which is the gateway of all world intercommunication. The Internet has showed to be the fastest and cheapest technology to acquire information, and has forced legacy technologies (banking services, for example) to adapt itself so that people with permanent or momentary difficulties can be able to interact with society. However, to give an accessible product is not always an easy task. In addition to several different classes of disabilities / difficulties (which leads to different accessibility problems), lack of training and experience in the area makes developers producing code in a wrong way, resulting in an inaccessible product. The process models and software development frameworks have not been adapted in a consistent and homogeneous way, contemplating the accessibility in the software factory. We are going through a transition phase between from the HTML and XHTML 4 to HTML 5, which among other things, aims to deliver a semantic web and to treat specific problems of accessibility, but it's not yet fully consolidated. Finally, the tools available to developers cannot effectively assist developers to deliver an affordable product. In this work it is considered that the accessibility requirements should be taken into account during all phases of software development, ie, must evolve from initial requirements analysis to the phase of software testing in order to obtain accessibility as an attribute of software quality of the final product. Thus, we sought primarily to create an approach that could promote accessibility requirements traceability from conception to the coding phase. This approach has associated Requirements, UML models and implementation techniques for accessibility, mapped in an accessibility ontology. In addition, we developed a plugin for Eclipse that promoted the association of technical implementation of accessibility and traceability matrix. We created a proof of concept with the proposal to assess whether the objectives were achieved. The work showed that it is possible to check, automatically, the traceability of accessibility requirements as well as its implementation techniques, from Requirements Engineering phase to Coding phase.

Keywords: *accessibility, requirement traceability, software development process, CASE tool.*

Sumário

Lista de siglas

ACCLIP Accessibility for LIP. 18

ACCMD AccessForAll Meta-data. 18, 21

AJAX Asynchronous JavaScript and XML. 12

API Application Programming Interface. 37, 40, 41, 44

AR Accessibility Requeriments. 18

AST Abstract Syntax Tree. 47

ATAG Authoring Tool Accessibility Guidelines. 10, 13, 14

BIRT Business Intelligence Reporting Tools. 36

CASE Computer-Aided Software Engineering. 26, 29–34, 36, 37, 49, 62

CLF Common Look and Feel. 12

CMS Content Management System. 13, 14

CSS Cascade Style Sheet. 20

DDA Disability Discrimination Act. 9

DHTML Dynamic HTML. 12

DP Device Profile. 18

DR Design Rationale. 24

e-MAG Modelo de Acessibilidade de Governo Eletrônico. 12

EARL Evaluation and Report Language. 10, 15, 19

EMF Eclipse Modeling Framework. 31, 37, 38, 40, 41, 45, 46, 64

EMFT Eclipse Modeling Framework Technology. 37

HTML HyperText Markup Language. 13, 14, 20, 33

HTTP HyperText Transfer Protocol. 20

HTTPS HyperText Transfer Protocol Secure. 18

ID Identifier. 47

IDE Integrated Development Environment. 37, 38

IETF The Internet Engineering Task Force. 10

IMS IMS Global Learning Consortium. 18

IRI Internationalized Resource Identifier. 41

J2ME Java Plataform Micro Edition. 20

JavaCC Java Compiler Compiler. 19

JDT Eclipse Java development tools. 47

LIP Learner Information Package. 18

MTA Modelo de Tarefas de Acessibilidade. 4, 5, 24, 26–29, 31, 35, 37, 39, 49, 50, 54, 58, 59, 61–65

NDA National Disability Authority. 12

NFR Non-Functional Requeriments. 18

ODS OpenDocument Spreadsheet file format. 44, 60

OMG Object Management Group. 29

OWL Web Ontology Language. 32, 33, 39, 41

PAM Problem Articulation. 23

PM Page Measurement. 22

RDF Resource Description Framework. 15, 20, 31, 41, 47

RMF Requirements Modeling Framework. 38

RSS Really Simple Syndication. 19

RUP Rational Unified Process. 2

SIM Subscriber Identification Module. 20

SMIL Synchronized Multimedia Integration Language. 14, 20, 21

UAAG User Agent Accessibility Guidelines. 10, 14

UGC User-generated Content. 21

UI User Interface. 46

UML Unified Modeling Language. 5, 29–31, 35–42, 44–47, 49, 53–55, 57, 59, 60, 63–65

UP Unified Software Development Process. 2

URI Uniform Resource Identifier. 41

UWEM Unified Web Evaluation Methodology. 22

W3C World Wide Web Consortium. 10, 11, 13, 14, 32

WAB Web Accessibility Barriers. 22

WAI Web Accessibility Initiative. 10, 11, 13, 14

WAI-ARIA Accessible Rich Internet Applications Suite. 10, 12, 13, 18, 19

WAQM Web Accessibility Quantitative Metric. 22

WCAG Web Content Accessibility Guidelines. 10–13, 19, 21, 23, 26, 32, 33, 37, 51–53, 55–57, 59, 64

WWW World Wide Web. 7

WYSIWYG What You See Is What You Get. 3, 14

XHTML eXtensible Hypertext Markup Language. 19

XMI XML Metadata Interchange. 29, 31, 64

XML Extensible Markup Language. 14, 29, 31, 41, 64

XP eXtremming Programming. 2

XPath XML Path Language. 20

XSLT eXtensible Stylesheet Language for Transformation. 20

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contexto e Motivação

A utilização de sistemas para internet é cada vez mais comum. Atividades que antes eram realizadas presencialmente estão migrando para serviços virtuais que podem ser acessados a partir de qualquer dispositivo com acesso à internet.

A Receita Federal Brasileira, por exemplo, só recebe as declarações de Imposto de Renda via internet (IRPF, 2013). As declarações emitidas em papel foram completamente abandonadas no ano de 2011 (Camargo, 2011). Essa decisão foi tomada pois a declaração feita em papel é mais vagarosa em todos os aspectos se comparada ao envio da declaração pela Internet e, além disso, era muito grande o número de declarações preenchidas erroneamente usando os formulários impressos (IRPF, 2011).

Alguns Tribunais já instituiram a emissão de certidões apenas por meio eletrônico. Um exemplo é o Tribunal de Justiça do Estado do Ceará, que permite a emissão de Certidão Negativa Criminal por meio de seu *site* (TJCE, 2011a). Segundo o próprio Tribunal, os motivos para adotar a emissão de Certidão *online* implicam em "rapidez, transparência, amplo acesso, interatividade e significativa redução de custos materiais e humanos, contribuindo para os resultados de excelência que se pretende alcançar na prestação dos serviços do Judiciário à população" (TJCE, 2011b).

O Conselho Monetário Nacional autorizou uma medida que permite que os bancos possam oferecer aos seus clientes uma conta movimentada exclusivamente por meios eletrônicos. Assim, o cliente ficará isento de cobranças de tarifas caso faça movimentações usando canais eletrônicos, como Internet, caixas eletrônicos e celular (NE10, 2011).

Os exemplos citados mostram a importância da Internet na disseminação de informação e fornecimento de diversos serviços. A Internet há muito tempo não é tratada mais como uma ferramenta de luxo ou acessória, mas sim como uma ferramenta essencial nas atividades do cotidiano das pessoas. Há várias vantagens em se usar a Internet para apoiar a comunicação e os negócios das empresas, por exemplo (Oliveira, 2011):

- Disponibilidade de 24 horas por dia;

- Possibilidade de acesso de todas as partes do planeta;
- Necessidade de espaço físico e de infra-estrutura reduzidos (ex: bancos) para realizar as atividades;
- Custo de investimento inicial baixo, etc.

Todos deveriam ter pleno acesso aos recursos fornecidos pela Internet. No entanto, devido a dificuldades e deficiências de determinados grupos de pessoas, o acesso aos recursos pode ficar comprometido, sendo que em algumas situações sequer o acesso é possível. É necessário, portanto, fornecer maneiras de garantir a acessibilidade de informações e serviços da *Internet*.

Sun e Zhang (2009) definem Acessibilidade Digital como o requisito básico para fornecer um acesso equalitário à informação extraída da *Internet*, principalmente e especialmente por ser uma maneira vital de acesso ao conteúdo por grupos vulneráveis (principalmente pessoas com necessidades especiais). Necessidades especiais não se referem apenas a pessoas com deficiência física ou mental, mas também a pessoas com algum tipo de impossibilidade momentânea, como a navegação através de um *browser* textual ou a visualização de um *site* através de um *smartphone*, que possui suas dimensões reduzidas.

Entregar produtos acessíveis não é uma tarefa fácil. O assunto é considerado relativamente recente e está sendo alvo de muitas pesquisas (Lazar *et al.*, 2004; Brajnik, 2006; Parmanto e Zeng, 2005). A acessibilidade na *web* também já é regulamentada em vários países, com diretivas e boas práticas que norteiam sua aplicação. Por isso, o papel da Engenharia de *Software* neste contexto é fundamental. A falta de metodologia e processos bem definidos específicos para acessibilidade podem gerar produtos não acessíveis. Os custos são menores quando a acessibilidade é considerada durante o processo de desenvolvimento do *software* (Groves, 2011). Apesar de existir um custo implícito embutido (por exemplo, contratação de um especialista em acessibilidade, ferramentas próprias para testes em acessibilidade, etc), em geral, os benefícios alcançados pela incorporação da acessibilidade nos processos são maiores do que os custos envolvidos necessários para a entrega do produto, pois o custo de manutenção posterior geralmente é mais alto e há um acréscimo no valor agregado ao produto (Sherman, 2001).

Existem propostas para integrar usabilidade e acessibilidade aos processos de Engenharia de *Software* (Moreno *et al.*, 2009; Maia, 2010). Outras propostas são específicas para determinadas plataformas (Microsoft, 2009). Contudo, não são encontrados na literatura trabalhos que integram acessibilidade a modelos e *frameworks* de desenvolvimento bem conhecidos, como Unified Software Development Process (UP) (Jacobson *et al.*, 1999) (ou sua extensão mais conhecida Rational Unified Process (RUP) (IBM, 2013b)), eXtremming Programming (XP) (Beck, 2000) ou Scrum (DeGrace e Stahl, 1990).

Tão importante quanto integrar acessibilidade no processo de desenvolvimento de *software* são as competências técnicas do Engenheiro de Software, Analista de Sistemas, Projetista e Desenvolvedor. Devido a fatores como a recente exposição do tema nos meios de comunicação e a deficiência no treinamento e formação do desenvolvedores, muitos deles sequer sabem como codificar para tornar seus produtos acessíveis (Kavcic, 2005;

Alves, 2011). As várias classes de problemas de acessibilidade contribuem para tornar o trabalho de codificação ainda mais difícil. A utilização de componentes de *software* com atributos de qualidade embutidos (sistemas de gerenciamento de conteúdo e sistemas *wiki* normalmente possuem um editor What You See Is What You Get (WYSIWYG) que já são acessíveis, como o *TinyMCE* (TinyMCE, 2013)) atenuam mas não eliminam o problema. A conscientização dos desenvolvedores vêm ocorrendo de forma gradual, mas poucas instituições de ensino brasileiras dedicam uma disciplina específica para Acessibilidade, ou dedicam pouco tempo em disciplinas de âmbito maior como Interface Humano-Computador, Hipermídia/Multimídia ou Engenharia de *Software* (Exemplos de grades curriculares: Harvard[1], MIT[2], USP[3], UFMS[4] e IFMS[5]).

A utilização de ferramentas que apóiam os profissionais na construção de boas soluções acessíveis é muito comum, pois aumenta a produtividade, retirando boa parte do esforço necessário e diminuindo passos repetitivos que seriam necessários sem o auxílio de tais ferramentas. Em geral, são utilizadas ferramentas *CASE, IDEs e frameworks*, mas existem algumas ferramentas específicas para o contexto de acessibilidade. Existem várias classes destas ferramentas específicas, entre elas:

- simulador: a ferramenta simula uma deficiência/dificuldade, mostrando ao desenvolvedor o problema do produto (Votis *et al.*, 2009);
- validador: a ferramenta utiliza um conjunto de normas e padrões pré-definidos e, através da validação objetiva, julga se o produto atende ou não aos critérios definidos[6];
- avaliador: a ferramenta utiliza um validador, complementando com informações, dicas e métricas, além de apresentar potenciais problemas que devem ser verificados manualmente[7].

Os avaliadores são ferramentas úteis, utilizados para fiscalização e auditoria de *sites*. Porém, pesquisas mostram que os desenvolvedores não estão satisfeitos com os avaliadores disponíveis (Trewin *et al.*, 2010). As ferramentas nem sempre informam de forma objetiva as mudanças necessárias para fornecer um produto acessível (Groves, 2012). Desenvolvedores iniciantes podem não entender as informações apresentadas na ferramenta. E, principalmente, os desenvolvedores estão insatisfeitos em utilizar ferramentas externas ao seu ambiente de desenvolvimento para efetuar a avaliação (Trewin *et al.*, 2010).

A situação é agravada pela constatação de que há poucos estudos que abordam a engenharia de requisitos e rastreabilidade do ponto de vista do tópico acessibilidade, ou seja, poucos estudos têm indicado como ocorre a evolução dos requisitos de acessibilidade durante o processo de desenvolvimento das aplicações web (Dias *et al.*, 2010).

[1] http://www.seas.harvard.edu/teaching-learning/undergraduate/computer-science/planning-and-courses
[2] http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/#undergrad
[3] http://www.ime.usp.br/dcc/grad/grade
[4] http://www.sien.ufms.br/cursos/grade/1904
[5] http://www.ifms.edu.br/rightsidebar/cursos/graduacao-2/tecnologia-em-sistemas-para-internet/
[6] Exemplo de validador da W3C: http://validator.w3.org/
[7] Exemplo de avaliadores: Hera(http://www.sidar.org/hera/) e daSilva(http://www.dasilva.org.br/)

1.2 Objetivos

Diante do exposto, os objetivos deste trabalho são:

1. Estender o MTA (tratado na seção 3.8), propondo uma metodologia para a rastreabilidade dos requisitos de acessibilidade através do processo de desenvolvimento de *software*;
2. Permitir a associação explícita entre os requisitos de acessibilidade e os artefatos de documentação e, para cada associação, especificar uma ou mais técnicas de implementação de acessibilidade de acordo com o documento de conformidade em acessibilidade escolhido;
3. Implementar uma ferramenta de suporte que seja integrada ao ambiente de desenvolvimento e que implemente os objetivos listados acima.

Busca-se identificar pontos de integração entre as etapas de engenharia de requisitos, projeto e implementação de código e propor uma ferramenta (*plugin* Eclipse) que represente esta integração. Assim, pretende-se que os requisitos de acessibilidade se tornem itens de acessibilidade verificáveis e o desenvolvedor deverá ser informado se a associação requisitos, modelos e código está ou não sendo feita. A ferramenta deverá apresentar sugestões para que os itens de acessibilidade sejam implementados. Com isso, pretende-se desenvolver uma ferramenta de apoio à acessibilidade diferente daquelas apresentadas na literatura, pois terá como foco a acessibilidade durante o processo de desenvolvimento.

Com base na pesquisa de Trewin *et al.* (2010), pretende-se que a ferramenta:

- seja orientada ao desenvolvedor (a apresentação dos resultados nas ferramentas tradicionais são adequadas para avaliação e auditoria de *sites*, e não para desenvolvedores);
- seja integrada ao ambiente de desenvolvimento do desenvolvedor;
- apresente informações objetivas e no momento em que o desenvolvedor desejar visualizar;
- tenha relação direta entre os requisitos e casos de uso com a etapa de codificação;
- permita que seja feita o rastreamento dos requisitos de acessibilidade, desde a sua concepção até a fase de codificação.
- permita que o desenvolvedor consiga verificar, em nível de código, a associação dos requisitos e modelos.

1.3 Metodologia

As atividades necessárias para alcançar os objetivos propostos, conforme mostradas na figura 1.1 são:

1. conhecer e definir o escopo do trabalho;
2. estudar a literatura sobre o assunto;
3. identificar os pontos de integração entre as atividades de engenharia de requisitos, projeto e geração de código;
4. estudar o processo de desenvolvimento de *plugins* para o *Eclipse*;
5. estudar como técnicas de acessibilidade podem ser associadas aos modelos UML;
6. estudar quais tecnologias existentes podem ser usadas para efetuar a associação dos requisitos e modelos às técnicas de acessibilidade;
7. desenvolver a ferramenta;
8. desenvolver uma prova de conceito, criando um projeto utilizando o MTA e a ferramenta proposta;
9. apresentar os resultados e conclusões.

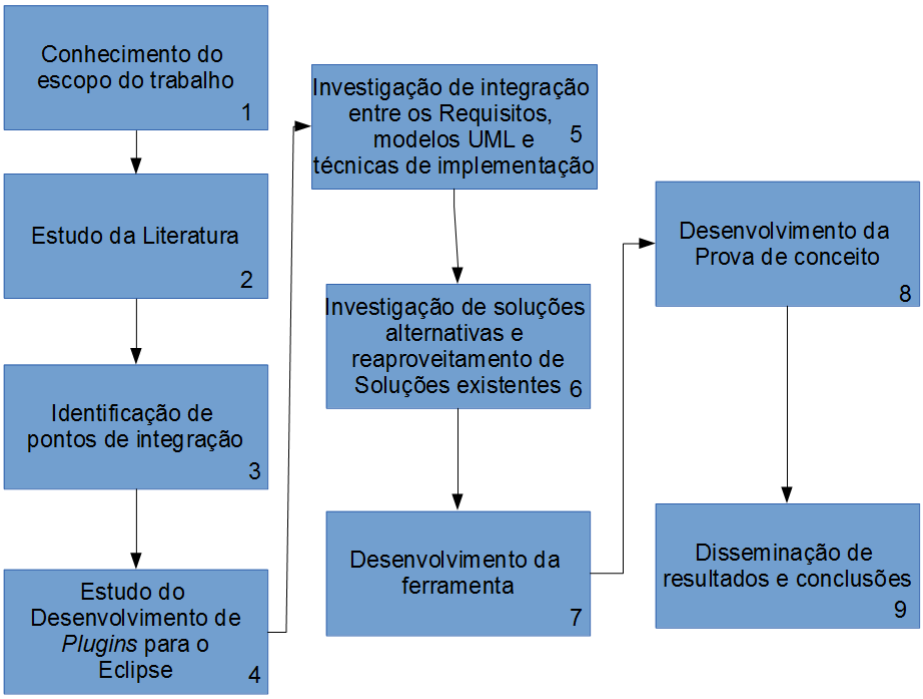


Figura 1.1: Etapas para a realização deste trabalho

1.4 Organização do Trabalho

Este trabalho está organizado em seis capítulos. O primeiro capítulo contextualizou a acessibilidade no desenvolvimento de *sites*, apresentou alguns desafios conhecidos na área e a motivação para o desenvolvimento do mesmo.

O segundo capítulo apresenta os detalhes do domínio de acessibilidade na *Internet*. Esse capítulo apresenta as tecnologias e conceitos que devem ser considerados para entregar produtos acessíveis. Também apresenta as iniciativas e legislações acerca de acessibilidade de produtos *web*. Por fim, são apresentados os principais documentos, referências e diretrizes para que os *sites* sejam acessíveis.

O terceiro capítulo contém o levantamento bibliográfico realizado, com breves descrições de cada trabalho. O capítulo tem por objetivo mostrar quão amplo é o domínio de acessibilidade na *Internet*, bem como as propostas sugeridas e os problemas ainda sem solução.

O quarto capítulo detalha o processo de integração de requisitos de acessibilidade ao processo de desenvolvimento de *software* (MTA), apresentando uma ferramenta para demonstrar como a integração ocorre.

O quinto capítulo apresenta uma prova de conceito, demonstrando a utilização da ferramenta construída para a abordagem escolhida, bem como sua efetividade e limitações.

O sexto e último capítulo apresenta as conclusões gerais e trabalhos futuros.

Capítulo 2

Acessibilidade na Web

O modo de vida das pessoas começou a mudar radicalmente depois que Tim Berners-Lee criou a World Wide Web (WWW), mais conhecida como *Web*. A *Web* é um ambiente de documentos e dados interconectados globalmente através da Internet (McPherson, 2009). Em sua concepção inicial, a *Web* foi explicitamente criada para poder ser utilizada sem o mouse, e até sem usar os olhos, se necessário (Thatcher *et al.*, 2006). Portanto, a maneira como o usuário acessa um recurso na *Web* pouco ou nada pode influenciar na aquisição da informação.

A fim de tornar os *sites* mais atrativos, várias tecnologias surgiram, como *JavaScript* (Goodman, 2001) e *Flash* (Rey, 2002). Todavia, o uso indiscriminado dessas tecnologias podem, ao mesmo tempo, facilitar, inibir ou impedir o acesso aos recursos de alguns usuários, principalmente dos usuários com algum tipo de deficiência.

Obrigados por regulamentações e leis específicas ou não, os desenvolvedores gradualmente estão adaptando seus *sites* e produtos para as necessidades dos usuários. O grande entrave para a disseminação da cultura de acessibilidade na *Web* está na conscientização dos desenvolvedores sobre a importância do tema, e sobre as consequências trazidas pela utilização de tecnologias que se tornam barreiras para o acesso ao conteúdo disponibilizado na Web (Freire, 2008; Alves, 2011).

O objetivo deste capítulo é apresentar os conceitos relevantes que devem ser considerados para que os serviços *Web* entregues sejam acessíveis. Serão apresentadas as tecnologias assistivas que auxiliam pessoas com deficiência, as iniciativas para o desenvolvimento de sistemas *Web* acessíveis, a legislação e leis que asseguram a acessibilidade na *Web*, as diretrizes que guiam o desenvolvimento de produtos *Web* acessíveis e questões referentes a avaliação de acessibilidade de produtos *Web*.

2.1 Tecnologias Assistivas e *Design Universal*

Pesquisadores da Universidade Estadual da Carolina do Norte definem *Design Universal* como o *design* de produtos e ambientes usáveis por todas as pessoas, sem precisar de

adaptações ou *design* especializados. Também propõem sete princípios básicos para o *design* de produtos, de forma que esse objetivo seja alcançado. São eles (NCSU, 1997):

- utilização equalitária: o produto é útil e comercializável às pessoas com diversas habilidades;
- utilização flexível: o produto suporta uma grande variedade de preferências individuais e habilidades;
- utilização simples e intuitiva: o uso do produto é fácil, independentemente da experiência do usuário, nível de conhecimento, competências linguísticas, ou educação;
- informação perceptível: o produto transmite a informação necessária eficazmente para o utilizador, independentemente do ambiente, condições ou habilidades sensoriais do usuário;
- tolerância a erros: o produto minimiza perigos e as consequências adversas de ações involuntárias ou acidentais;
- esforço físico mínimo: o produto pode ser usado de forma eficiente e confortável e com um mínimo de fadiga;
- espaço e tamanho adequado para aproximação e utilização: o tamanho apropriado e o espaço para utilização do produto é suficiente para abordagem, alcance, manipulação e uso independentemente do corpo do usuário; tamanho, postura ou mobilidade.

Esses princípios, obviamente, não são especificamente voltados para o desenvolvimento de sistemas *Web*, mas podem ser facilmente adaptados para tal. Ainda assim, desenvolver sistemas *Web* completamente acessíveis é uma tarefa difícil. Deve-se considerar também que alguns dispositivos são intrinsicamente inacessíveis para certos tipos de usuários, por exemplo, um usuário cego não conseguirá extrair informações de um monitor tradicional.

Diante desse fato, é necessário fornecer aos usuários tecnologias intermediárias para interceptar ou transformar a informação, apresentado-a de uma maneira que o usuário consiga entender. Essas tecnologias são denominadas tecnologias assistivas. Tecnologia assistiva é, portanto, o conjunto de equipamentos, serviços, estratégias e práticas concebidas para atenuar os problemas encontrados pelas pessoas com necessidades especiais (Cook e Hussey, 1995).

Existem diversos tipos de tecnologias assistivas, a escolha depende da necessidade do usuário. É importante considerar que a dificuldade no acesso pode ser permanente ou momentânea. A seguir serão apresentadas algumas tecnologias assistivas para certas deficiências:

- Cegueira

  - Leitor de tela: software que lê o que está na tela do computador e informa a saída, seja em um sintetizador de voz, ou um display *braille*. Ex: *JAWS* (Freedom Scientific, 2013a) e *DOSVOX* (NCE-UFRJ, 2002);
  - Navegador textual: navegador que não carrega imagens. Deve ser usado em conjunto com o Leitor de Tela. Ex: *Lynx* (LYNX, 1998)
  - Navegador com voz: permite a navegação por meio da voz. Ex: *Opera* (Opera Software, 2013) e *Firefox* com o plugin *Text to Voice* (Mozilla, 2012)

- Baixa visão
  - Ampliador de tela: software que amplia o conteúdo da página para facilitar a leitura. Ex: *MAGic Screen Magnification* (Freedom Scientific, 2013b)

- Deficiência física
  - *Eye-tracking*: capta as ações por meio do movimento dos olhos. Ex: *LEA* (Open Source Software, 2009)
  - Teclado alternativo: dependendo da deficiência do usuário ou do dispositivo, um teclado alternativo pode ser inserido no contexto. Para deficientes visuais, teclados *braille* são indicados (Manohar e Parthasarathy, 2009). Para usuários com dispositivos móveis, um teclado virtual pode ser a melhor solução (Sarcar *et al.*, 2010).

- Deficiência auditiva
  - Quando o usuário não perdeu 100% da audição, ele pode usar softwares que melhoram o desempenho do áudio do computador. Mas o mais comum é a utilização de legendas juntamente com as mídias que possuem informações sonoras.

No momento da pesquisa não foram encontradas tecnologias assistivas para outros tipos de deficiências.

2.2 Legislação sobre acessibilidade na *Internet*

A primeira Lei sobre acessibilidade na *Internet* foi a *Section 508*, proposta em 1998 pelo Governo dos Estados Unidos (U.S. Department of Justice, 2011). Segundo essa lei, a tecnologia inacessível interfere na capacidade individual de adquirir e usar a informação de maneira rápida e fácil. A *Section 508* foi decretada para eliminar barreiras na tecnologia da informação, disponibilizando novas oportunidades para as pessoas com necessidades especiais e incentivando o desenvolvimento de tecnologias que as auxiliem a atingir esses objetivos (Freire, 2008).

Após essa iniciativa, países em todo o mundo começaram a incorporar tópicos de acessibilidade no meio digital em suas leis já existentes, ou até mesmo confeccionar novas. Por

exemplo, a Lei Disability Discrimination Act (DDA), instituída pelo governo britânico em 1995 (NIDirect, 1995), foi alterada em 1999 e passou a possuir uma seção especificamente sobre *websites* (Webcredible, 2011). É válido mencionar que o "nível" de acessibilidade do *site* não é mencionado: o DDA requer que sejam realizadas "adaptações razoáveis" para garantir que uma pessoa com deficiência possa acessar o mesmo.

No Brasil, as Leis 10.048/2000 (Planalto, 2000a) e 10.098/2000 (Planalto, 2000b), regulamentadas pelo Decreto 5.296/2004 (Planalto, 2004), tornam obrigatória a acessibilidade nos websites da administração pública para o uso das pessoas com necessidades especiais para garantir o pleno acesso às informações.

2.3 Documentos e padrões

Existem dois grandes órgãos que documentam e fornecem diversas especificações e recomendações para os padrões e novas tendências da *Internet*. Um deles é o The Internet Engineering Task Force (IETF) (IETF, 2013) que tem um caráter mais arquitetural, preocupado principalmente com os protocolos e recursos utilizados na *Internet*. O outro é o W3C (W3C, 2013g) que tem um caráter voltado para *front end* e aplicações. Este último possui uma ramificação denominada WAI (WAI, 2013), que se preocupa com as questões referentes à acessibilidade na *Internet*.

Quando se trata de padrões e recomendações de acessibilidade na *Internet*, a W3C-WAI é referência internacional. Todo tipo de trabalho sobre acessibilidade na *Internet* está relacionado (direta ou indiretamente) aos documentos da W3C-WAI. A Figura 2.1 mostra quais documentos devem ser consultados pelos desenvolvedores para atingir seus objetivos sobre acessibilidade.

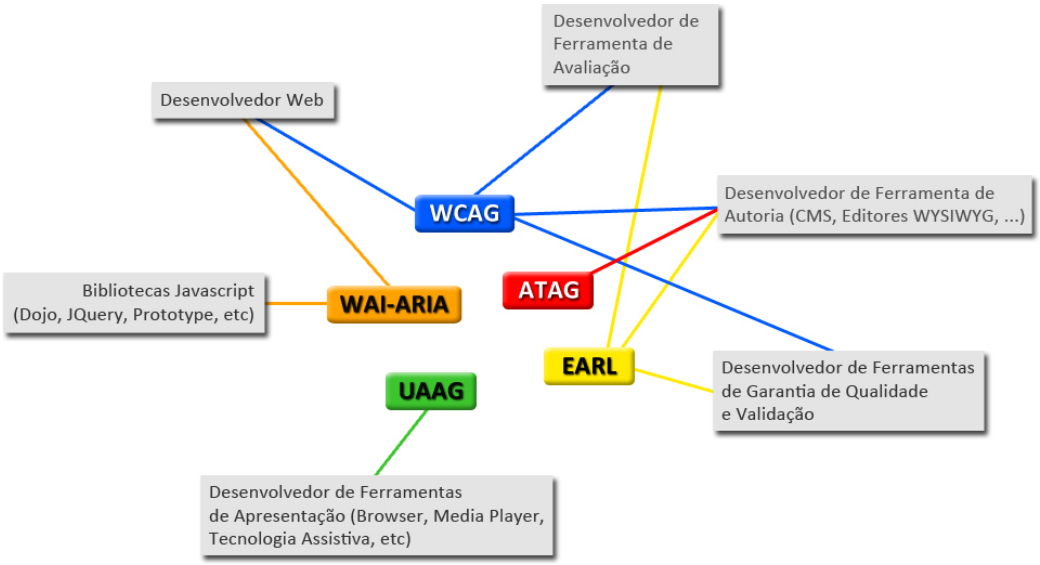


Figura 2.1: Relação de interesses entre os desenvolvedores e a documentação W3C-WAI: WCAG, ATAG, WAI-ARIA, UAAG e EARL

A seguir, serão apresentados os cinco principais documentos de referência publicados pela W3C-WAI.

2.3.1 WCAG

O WCAG é um documento que fornece diretrizes e recomendações para implementação de acessibilidade em *sites* (WCAG, 2013b). A versão 1.0 do WCAG foi publicada em 1999 (W3C, 2013f). O documento é dividido em 14 diretrizes, cada uma com seus respectivos *checkpoints*. Cada *checkpoint* por sua vez possui uma prioridade.

As prioridades são divididas em grupos: as prioridades nível 1 (devem ser atendidas), as prioridades nível 2 (deveriam ser atendidas) e as prioridades nível 3 (poderiam ser atendidas). Se alguma prioridade nível 1 não for atendida, um ou mais grupos de usuários ficarão impossibilitados de acessar as informações do documento. Se alguma prioridade nível 2 não for atendida, um ou mais grupos de usuários terão dificuldades em acessar as informações em qualquer parte documento. Se alguma prioridade nível 3 não for atendida, um ou mais grupos de usuários terão dificuldades em acessar alguma informação em partes específicas do documento.

Se todos os *checkpoints* com prioridade 1 forem atendidos, o documento possui nível de conformidade "A". Se todos os *checkpoints* com prioridade 1 e 2 forem atendidos, o documento possui nível de conformidade "AA". Por fim, se todos os *checkpoints* forem atendidos, o documento possui o nível de prioridade "AAA". A Tabela 2.1 relaciona as informações descritas até agora para os níveis de prioridade do WCAG 1.0.

Prioridade	**Recomendação**	**Nível de Conformidade**	**Impacto (para determinado grupo de usuários)**
Nível 1	devem ser atendidas	A (Nível 1)	impossibilidade de acesso ao documento
Nível 2	deveriam ser atendidas	AA (Nível 1 e 2)	dificuldade no acesso ao documento
Nível 3	poderiam ser atendidas	AAA (Nível 1, 2 e 3)	dificuldade no acesso de partes específicas do documento caso não seja atendida

Tabela 2.1: Níveis de Conformidade do WCAG 1.0

Em 2008, o W3C-WAI liberou a segunda versão do WCAG (WCAG, 2013c). Segundo a *WAI*, "O WCAG 2.0 aplica-se amplamente a tecnologias mais avançadas; é mais fácil de usar e entender; é precisamente validado com testes automáticos e avaliação humana". O documento possui 12 diretrizes, organizadas em 4 princípios: percepção, compreensão, operação e robustez. Cada diretriz possui critérios de sucesso testáveis, nas mesmas prioridades do WCAG 1.0. A *WAI* recomenda a utilização do WCAG 2.0 e, devido às semelhanças com o WCAG 1.0, sabe-se que a maioria dos sites em conformidade com

o WCAG 1.0 não requerem mudanças significativas para estar em conformidade com o WCAG 2.0 e alguns nem mesmo necessitam de mudanças. As principais diferenças podem ser vistas no site da *WAI*, na página específica sobre o assunto (WCAG, 2013a). Branco (2009) fez um breve comparativo dos dois documentos e constatou que a Tabela 2.1 é válida para o WCAG 2.0 e que, principalmente, as orientações de cada *guideline* se tornaram mais claras e fáceis de testar.

É válido observar que nem todos os desenvolvedores concordam com as diretrizes ou recomendações do WCAG 1.0 ou mesmo WCAG 2.0 (Clark, 2006), apesar deste último ter se tornado um padrão internacional (ISO/IEC, 2012). De fato, um grupo fechado de desenvolvedores preparou uma versão de extensão/atualização/correção não oficial do WCAG 1.0, denominada WCAG Samurai (WCAG Samurai, 2006).

Existem diversas diretrizes de acessibilidade, cada qual com suas particularidades e voltadas para realidades específicas. Alguns exemplos são: Common Look and Feel (CLF), substituída pelo documento *Web Standards for the Government of Canada*, do Canadá (Treasury Board of Canada Secretariat, 2013), National Disability Authority (NDA), da Irlanda (NDA, 1999) e as diretrizes de *design* de acessibilidade da *Microsoft* (Microsoft, 2009).

O governo brasileiro possui um padrão próprio, denominado Modelo de Acessibilidade de Governo Eletrônico (e-MAG). Atualmente na sua terceira versão (Governo Eletrônico, 2007), o e-MAG é baseado no WCAG 2.0, mas voltado para a realidade local. Em relação a versão anterior, o documento passou por mudanças sutis, por exemplo, a eliminação das divisões de visão técnica e visão do cidadão. Apesar desse modelo poder ser usado em outras iniciativas, é explicitado que o mesmo deve ser seguido para os *sites* governamentais. Por isso, não existem mais níveis de prioridade (todas as recomendações devem ser cumpridas) e uma seção especial foi incluída com o objetivo de padronizar elementos de acessibilidade que devem existir em todos os *sites* e portais governamentais.

2.3.2 *WAI-ARIA*

Estamos em plena era da *Web 2.0*, com *sites* complexos e dinâmicos e, consequentemente, controles da interface de usuário mais complexos também. As tecnologias assistivas precisam interagir com esses controles para que a experiência de utilização seja positiva para usuários com deficiência. Contudo, as informações que as tecnologias assistivas necessitam para poder manipular esses elementos complexos normalmente não estão presentes na maioria das tecnologias *Web*.

Tecnologias *Web 2.0* podem ser inacessíveis até para usuários que não possuem nenhum tipo de deficiência, mas possuem alguma barreira tecnológica, por exemplo, um menu *drag-and-drop* para o usuário que não possui um *mouse*. A atualização de determinadas regiões do *site* pelas tecnologias citadas também são um grande problema para as tecnologias assistivas (principalmente para o *Screen Reader*). Exemplos de tecnologias desse tipo são: Asynchronous JavaScript and XML (AJAX) (Garrett, 2005) e Dynamic HTML (DHTML) (Teague, 2001).

O documento WAI-ARIA pretende resolver esses desafios de acessibilidade definindo como a informação sobre a funcionalidade do elemento deve ser entregue à tecnologia assistiva, adicionando atributos, técnicas de navegação, marcação de regiões, entre outras (WAI-ARIA, 2013).

O WAI-ARIA é, portanto, um *framework* que indica aos desenvolvedores:

- papéis que descrevem o tipo do elemento apresentado;
- papéis que descrevem a estrutura da página;
- propriedades que descrevem o *status* do elemento;
- propriedades que definem as "regiões vivas" (dinâmicas);
- propriedades para elementos *drag-and-drop*;
- uma maneira para navegar e interagir com os elementos citados.

A documentação não está inteiramente completa. Ao final do ano de 2011, a versão definitiva do documento ainda não havia sido liberada. Como as especificações do HTML 5 e do WAI-ARIA ainda não estão completamente definidas, algumas marcações ficam sobrepostas (Tarr, 2011). Contudo, desenvolvedores influentes afirmam que o padrão HyperText Markup Language (HTML)5 não vai tornar o uso do WAI-ARIA redundante (Irish, 2011; Faulkner, 2010).

Esta documentação é voltada principalmente para desenvolvedores *Web*, desenvolvedores de tecnologias *Web 2.0* e desenvolvedores de *Javascript frameworks* como *jQuery* (Resig, 2006), *Dojo* (Schutta e Asleson, 2006), *MooTools* (Newton, 2008), *Prototype* (Orchard *et al.*, 2010), entre outros.

2.3.3 *ATAG*

Ferramentas de Autoria são *softwares* e serviços que permitem a qualquer pessoa produzir *sites* e conteúdo *web*. Esse tipo de ferramenta geralmente possui uma *engine* que exibe o conteúdo fornecido pelo usuário utilizando padrões e *templates* previamente configurados na ferramenta. Pela facilidade e rapidez de "postar" conteúdo, muitos usúarios (principalmente usuários leigos em computação) utilizam ferramentas desse tipo.

Existem várias ferramentas de autoria e cada uma tem sua particularidade. Existem ferramentas para produção de *blogs*, para produção de portais chamadas Content Management System (CMS), de criação de páginas *wiki*, entre outras. Alguns exemplos de ferramentas são: *Wordpress* (Wordpress, 2013), *Joomla* (Joomla, 2013), *Drupal* (Drupal, 2013), *Plone* (Plone, 2013), Sistemas *Wiki* (Wikipedia, 2013) e *Moodle* (Moodle, 2013).

Não raro, empresas e organizações desenvolvem suas próprias ferramentas de autoria, principalmente quando o tipo de conteúdo a ser postado é muito específico, ou quando a regra de negócios é diferente das ferramentas convencionais. Pesquisadores e estudantes

da Universidade Federal de Mato Grosso do Sul (UFMS) desenvolveram ferramentas de autoria própria, chamadas *Titan* (Carromeu *et al.*, 2010), *Pantaneiro* (Sandim, 2009) e *Pantaneiro acessível* (Maia, 2010).

É um desafio prover total acessibilidade para o conteúdo gerado por essas ferramentas, visto que este é fornecido pelo usuário que normalmente é leigo no assunto. A W3C-WAI publicou um documento destinado a produção de ferramentas de autoria, denominado ATAG (W3C, 2013a) e o objetivo deste documento é definir como as ferramentas de autoria deveriam ajudar os desenvolvedores a fornecer conteúdo em conformidade com o WCAG. Mais do que isso, este documento explica como tornar as ferramentas de autoria acessíveis para que pessoas com deficiência possam produzir seu próprio conteúdo.

O documento ATAG, assim como o WCAG, possui diretrizes e *checkpoints* para alcançar o resultado previsto. A versão 1.0, de Fevereiro de 2000 (ATAG, 2013a), possui 28 *checkpoints* e norteia o desenvolvimento de ferramentas de autoria levando em consideração:

- produzir o *site* em conformidade com as diretrizes de acessibilidade;
- ajudar o usuário da ferramenta de autoria com informações relativas a acessibilidade do *site*;
- prover maneiras de checar e corrigir conteúdo inacessível;
- integrar acessibilidade com ajuda, documentação e percepção; e
- tornar a própria ferramenta de autoria acessível para deficientes.

A versão 2.0 do documento ainda está em desenvolvimento (ATAG, 2013b). A novidade é que na nova versão, cada diretriz está relacionada a um princípio, perceptível ao desenvolvedor.

O documento ATAG é voltado principalmente para desenvolvedores de ferramentas de autoria, incluindo:

- ferramentas de edição WYSIWYG; editores HTML e Extensible Markup Language (XML));
- ferramentas que transformam documentos em formatos *web* (por exemplo, transformação *.doc* para *.html*);
- ferramenta de produção multimídia web (ferramentas de autoria Synchronized Multimedia Integration Language (SMIL)) (SMIL, 2013);
- ferramentas de publicação e CMS;
- ferramentas de gerenciamento de *layout*;
- qualquer ferramenta que permite aos usuários publicar e compartilhar conteúdo.

2.3.4 *UAAG*

As ferramentas que permitem ao usuário acessar o conteúdo *Web* são chamadas *User Agents*, ou Agentes de Usuário. Essas ferramentas incluem navegadores, *media players* e tecnologias assistivas.

A W3C-WAI possui um documento voltado para os desenvolvedores de *User Agents* chamado UAAG (UAAG, 2013c). A versão 1.0 do documento, de Dezembro de 2002 (UAAG, 2013a), possui 12 diretrizes de acessibilidade. A versão 2.0 (UAAG, 2013b) ainda não foi finalizada.

O conjunto de *checkpoints* que o UAAG 1.0 cobre são:

- acesso a todo o conteúdo, incluindo conteúdo amarrado a eventos disparados por ações de teclado e *mouse*;
- controle de usuário sobre a maneira como o conteúdo é renderizado[1];
- controle de usuário sobre a interface, com informações sobre as características de acessibilidade;
- interface de programação padrão, para prover interação com tecnologias assistivas.

2.3.5 *EARL*

O documento EARL é na verdade um formato definido para expressar resultados de testes, principalmente os resultados gerados automaticamente por ferramentas de avaliação (EARL, 2013b). Este documento utiliza Resource Description Framework (RDF) (W3C, 2013d) para definir os termos que expressarão os resultados dos testes.

A versão 1.0 do documento ainda está em desenvolvimento (EARL, 2013a) e tem o objetivo de tratar os resultados de testes das seguintes ferramentas:

- ferramentas de avaliação de acessibilidade;
- ferramentas de garantia de qualidade e validação;
- ferramentas de autoria e desenvolvimento; e
- descrição de conteúdo Web.

[1]Renderização é o processo de geração de uma imagem por meio do processamento de um modelo computacional

2.4 Considerações Finais

Neste capítulo foram apresentados a legislação, documentos e tecnologias de maior relevância para o contexto de acessibilidade na *Web*. O próximo capítulo apresenta a pesquisa bibliográfica realizada neste trabalho.

Capítulo 3

Pesquisa Bibliográfica

Para realizar este trabalho, uma pesquisa bibliográfica foi executada com o intuito de verificar as principais áreas de interesse no assunto, bem como a motivação dos pesquisadores e os resultados alcançados. Os artigos estudados levaram em conta o assunto de acessibilidade no contexto de Engenharia de *Software*, destacando-se algumas fases e atividades do processo de desenvolvimento de *software* indicados a seguir:

- Requisitos
- Arquitetura
- Navegação
- Interface
- Conteúdo
- Avaliação
- Processo de Desenvolvimento

Foram analisadas também *surveys* que indicavam o estado da arte e da prática em relação ao assunto.

Os resultados da pesquisa bibliográfica são apresentados a seguir. Para melhor entendimento, os artigos estão organizados em seções de acordo com os itens descritos anteriormente (alguns artigos poderiam ser encaixados em mais de um item, dessa forma optou-se por deixar no item de maior relação com este trabalho).

3.1 Requisitos

Trewin *et al.* (2010) realizaram uma pesquisa com os desenvolvedores da *IBM* em que o objetivo foi verificar como as ferramentas de apoio a acessibilidade podem ajudar os desenvolvedores. Algumas métricas relacionadas ao tempo foram coletadas em projetos que

envolviam acessibilidade. Como resultado, os desenvolvedores identificaram que as ferramentas atuais de apoio à acessibilidade tornam o processo de desenvolvimento moroso. Questões importantes sobre as ferramentas de apoio foram levantadas, tais como:

- Quais funcionalidades a ferramenta de apoio deve oferecer?
- A ferramenta de apoio deve ou não ser integrada à ferramenta de desenvolvimento?

A pesquisa constatou que o *design*, os testes e a identificação de soluções tecnológicas para o problema são os aspectos mais difíceis para se produzir uma aplicação *web* acessível. A pesquisa mostrou também que os desenvolvedores não confiam completamente nos resultados fornecidos pelas ferramentas de testes, por causa dos falsos positivos, além de mencionar que as explicações fornecidas pelas ferramentas não são suficientemente detalhadas.

Dias *et al.* (2010) estudaram como ocorre a inserção de acessibilidade nas etapas de desenvolvimento de *software*. Para isso, foram buscados artigos nos portais *Springer, ACM, IEEE, Elsevier, Wiley e Scielo*.

Os artigos relevantes foram classificados de acordo com a norma *ISO/IEC 12207:1998* (Requisitos, Projeto, Construção, etc) (ISO/IEC, 1998). Uma das observações dos autores é que a maioria das pesquisas se refere a testes com usuários, enquanto nenhuma contempla instalação de *software*. Esta é uma constatação interessante, visto que usuários com deficiência visual conseguem instalar completamente um sistema operacional *Linux*[1], mas há vários requisitos anteriores que devem ser considerados para que o usuário consiga executar a tarefa de forma adequada (por exemplo, baixar a imagem do Sistema Operacional e gravar em um CD/DVD ou *pen drive*). Nada foi encontrado na literatura sobre a instalação de outros sistemas operacionais (por exemplo, *Windows*) por deficientes visuais.

Masuwa-Morgan (2008) propôs uma ferramenta para especificação de requisitos de acessibilidade. Sua proposta é unir os requisitos das diretrizes de acessibilidade com os requisitos da engenharia de *software* tradicional, incluindo requisitos específicos de acessibilidade no Documento de Requisitos. O trabalho é baseado nos *checklists* e *checkpoints* das diretrizes de acessibilidade mais conhecidas.

A ferramenta, nomeada *AccessOnto*, aproveita a saída de algumas ferramentas *CASE* tais como *SELECT* (Select, 2013) e prevê integração com o *IBM Rational Policy Tester Accessibility* (IBM, 2011). Um *XML Schema* provê as descrições, regras e classes, descrevendo as diretrizes, características do usuário e objetos no ambiente de interação. A ferramenta é usada para intervir em diagramas de caso de uso, diagramas de classes e diagramas de transição de estados.

Mikic *et al.* (2007) discutiram as iniciativas de padronização de acessibilidade no domínio *e-Learning*. De acordo com os autores, há muitas instituições trabalhando na padronização de tecnologias *e-Learning*, nos seguintes subdomínios: metadados, agregação de conteúdo, informações do estudante, acessibilidade e *Runtime*.

[1]https://help.ubuntu.com/community/Accessibility e http://www.linuxacessivel.org/

Os autores aprofundaram seus estudos nos padrões Learner Information Package (LIP), Accessibility for LIP (ACCLIP) e AccessForAll Meta-data (ACCMD) (IBM, 2013a), propostos pela IMS Global Learning Consortium (IMS) (IMS, 2013), e explicaram a proposta de Device Profile (DP), estendendo os padrões mencionados.

Sun e Zhang (2009) estudaram os princípios e conceitos de acessibilidade em *sites* educacionais. Os autores descrevem a heteogeneidade dos estudantes (questões como deficiências, equipamentos, habilidades, idade, entre outras). Os autores sugerem que alguns princípios de *design* devem ser assumidos (flexibilidade, simplicidade e tolerância a erros. Também sugerem aplicar um processo de desenvolvimento *top-down*, partindo da etapa de Definição e passando pelas etapas de Análise, Protótipo, Testes, Integração, Liberação e Manutenção, para portanto atingir os objetivos e prover um produto acessível.

Baguma *et al.* (2009) estudaram como incorporar acessibilidade junto a análise de requisitos. Uma pesquisa inicial foi feita para verificar em que estágio de desenvolvimento a acessibilidade é considerada e quanto tempo é dedicado a esse fim. Há uma discussão se requisitos de acessibilidade devem ser considerados requisitos funcionais ou requisitos não funcionais, já que alguns autores consideram acessibilidade como uma sub-área de usabilidade.

Os autores trataram requisitos de acessibilidade como requisitos não funcionais, utilizando como base o *NRF Framework*, que representa o requisito não funcional através de operacionalizações estáticas e dinâmicas, transformando a representação em um grafo (Cysneiros e do Prado Leite, 2001) e tratando os requisitos como objetivos (*goal graph*), que precisam ser satisfeitos. No trabalho de Baguma *et al.* (2009), alterações no Non-Functional Requeriments (NFR) *goal graph* foram feitas, transformando-o em um Accessibility Requeriments (AR) *graph*. O diagrama de casos de uso é alterado para comportar a acessibilidade.

Akhter *et al.* (2009) propuseram um *framework* conceitual para aumentar a confiança de *sites* por usuários que utilizam leitores de tela. Algumas diretrizes são propostas, tais como:

- informar que o usuário está em uma página segura (HyperText Transfer Protocol Secure (HTTPS));
- apresentar, assim que possível, informações sobre certificados e, se necessário, duplicar a informação;
- informar imediatamente se uma página é recarregada, possível através de regiões WAI-ARIA.

3.2 Arquitetura

Bigham *et al.* (2010) propuseram uma ferramenta para que usuários finais apontem problemas de acessibilidade. A ferramenta é uma extensão do leitor de tela *WebAnywhere* (WebAnywhere, 2013). Um script *server-side* guarda o conjunto de ações de usuário,

permitindo ao desenvolvedor reproduzi-las posteriormente. Essa ferramenta é útil para compensar a deficiência das ferramentas de avaliação automática, que não conseguem resolver completamente os problemas presentes no *site*.

Giakoumis *et al.* (2010) estudaram acessibilidade envolvendo *web services*. O estudo é baseado na forma como os *web services* se comunicam, através do protocolo *SOAP* (SOAP, 2013). Antes de entregar o conteúdo, o *web service* "desempacota" o mesmo e analisa utilizando as diretrizes WCAG. Por esse motivo, as funcionalidades são definidas em classes, assim como os níveis de conformidade do WCAG. Um módulo de validação foi desenvolvido, gerando relatórios EARL.

van Schaik e Ling (2008) estudaram a percepção do usuário em relação à qualidade de *sites* na internet. Foi realizada uma pesquisa para verificar a percepção do usuário com alguns *sites* pré-definidos. Os elementos do *site* eram reordenados e seu layout foi modificado. Dessa maneira, os usuários davam seu parecer antes e depois da utilização do mesmo.

A pesquisa considerou alguns princípios básicos, como beleza e qualidade, tratando-os como variáveis aleatórias para proceder com o estudo estatístico. O estudo mostrou que a variável aleatória *beleza* é mais estável do que, por exemplo, a percepção do usuário sobre qualidade. A base de conhecimento resultante pode ser usada para produzir um guia de *design* na forma de padrões de *design* para facilitar a compreensão e aplicação.

Buzzi *et al.* (2008) propuseram modificações na *Wikipedia* de forma que a mesma se torne acessível para contribuição de usuários cegos.

Os objetivos foram alcançados através das seguintes alterações:

- utilização de regiões WAI-ARIA, principalmente para questões como foco nos elementos;
- alteração da barra de ferramentas (de *javascript* para eXtensible Hypertext Markup Language (XHTML));
- mudança na forma de escolha de símbolos especiais.

3.3 Navegação

Votis *et al.* (2009) construíram um *plugin* para *NetBeans* que simula algumas deficiências visuais, para testar principalmente aplicações *Java Swing*. Os autores mencionam que existem vários simuladores de deficiências físicas na literatura, sendo que o simulador proposto por eles cobre vários tipos de deficiência visual (tais como catarata, hipermetropia, entre outras). Os resultados obtidos foram comparados com outros simuladores da mesma categoria. A implementação foi projetada usando um design centrado no usuário, justamente para entender suas maiores dificuldades.

Encelle e Baptiste-Jessel (2007a) propuseram a personalização da interface do usuário, tornando assim o conteúdo acessível. A proposta permite que os usuários definam como

o conteúdo deve ser apresentado. Os requisitos que devem ser considerados para alcançar esse grau de flexibilidade são: apresentação do conteúdo, navegação e busca. Como exemplo, os autores demonstraram as possibilidades da abordagem montando uma linguagem descritiva em Java Compiler Compiler (JavaCC) (JavaCC, 2011) para mídia Really Simple Syndication (RSS) (RSS, 2011). Os autores detalharam como a transformação ocorre (Encelle e Baptiste-Jessel, 2007b), usando SMIL SMIL (2013), XML Path Language (XPath) e eXtensible Stylesheet Language for Transformation (XSLT) (Tennison, 2001). Segundo os autores, é relativamente simples utilizar a proposta sugerida, bastando apenas três utilizações para o usuário conseguir aprender a usar ferramenta e personalizar a interface que estiver usando.

Ferres *et al.* (2007) propuseram um método para tornar gráficos acessíveis. A acessibilidade é alcançada da seguinte maneira: os gráficos são confeccionados no *Excel*, por meio de um *plugin* feito para esse fim. Depois de publicado, um *plugin* instalado no navegador reconhece o gráfico acessível, fornecendo a interação necessária para que o usuário consiga extrair toda a informação necessária.

A navegação no gráfico foi inspirada em leitores de tela, portanto é feita por meio do teclado. Além disso, o *software* de leitura do gráfico possui um sintetizador de voz embutido.

Michail e Christos (2007) propuseram a personalização da navegação nativa dos *sites*, de acordo com as preferências do usuário, por meio de um *framework* que estende o *SeEBrowser* (SeEBrowser, 2011). O framework permite guardar anotações de acessibilidade através de comandos *POST* e *GET* do protocolo HyperText Transfer Protocol (HTTP). As anotações são armazenadas no formato RDF.

O *software* possui algumas métricas para avaliar o nível dos atalhos fornecidos ao usuário, contando pressionamento das teclas, tempo gasto em cada página, entre outras. A navegação é reordenada baseada na relevância das informações previamente cadastradas, através das estatísticas coletadas.

3.4 Interface

Thinyane e Thinyane (2009) propuseram a possibilidade de customização das interfaces de aplicativos *web* (baseados em HTML e Cascade Style Sheet (CSS)) de celulares de acordo com as preferências do usuário. O grande diferencial dessa proposta é que, utilizando *Java Card* e Java Plataform Micro Edition (J2ME), as preferências são gravadas no Subscriber Identification Module (SIM) *chip*, e não na memória do celular, como a maioria das aplicações fazem. Dessa forma, é possível alterar o aparelho físico, mas as informações da interface não são perdidas.

Pauwels *et al.* (2009) questionaram a melhor forma de evidenciar campos obrigatórios em formulários. Vários pesquisadores afirmam que campos obrigatórios de um formulário devem aparecer logo no início, destacados. A *Apple* prefere a abordagem de usar asteriscos, e só destacar os campos que o usuário errou após o usuário submeter a ação.

Em termos de acessibilidade, a pesquisa mostra que a abordagem de campos coloridos é melhor do que a de asterisco, mas leitores de tela não são sensíveis a esse contexto. Aparentemente, é bom utilizar as duas abordagens. Apesar disso, os autores não conseguiram concluir qual cor deve ser utilizada, informando apenas que a cor utilizada para seus testes foi a amarela.

Halbach (2010) tratou da deficiência cognitiva (problemas com foco, dificuldade de leitura, entendimento, memória, entre outros). Os documentos WCAG 1.0 e 2.0 cobrem uma área limitada dos problemas que a deficiência cognitiva pode trazer. Portanto, o autor propõe um conjunto de princípios de *design* para cada sub-área e desenvolveu uma solução baseada no estudo de caso do *site* de serviços públicos da Noruega. Nesse *site*, 33% dos chamados do *Help Desk* eram relativos a tela de *login*. A solução apresentada é interessante pois usa padrões de *design* pouco usuais em desenvolvimento de *sites*, por exemplo, usar uma área para informar o que o usuário está fazendo no momento, e quais são os próximos passos. Isso auxilia pessoas com problemas de mémoria, principalmente idosos.

3.5 Conteúdo

Ferretti *et al.* (2008) propuseram, por meio de objetos ACCMD, uma forma compartilhada para prover acessibilidade em ambientes *e-Learning 2.0*, principalmente para objetos multimídia SMIL.

No ambiente virtual, o tutor disponibiliza o conteúdo, que depois pode ser acrescido de recursos pelas partes interessadas (tutor, estudantes, etc). A edição colaborativa é feita por uma interface *wiki-like*, na qual as "camadas" de acessibilidade são adicionadas aos objetos de aprendizagem, sob supervisão do tutor.

Martín García *et al.* (2009) estudaram a relação entre os *Prosumers* (usuários geralmente leigos que utilizam ferramentas de autoria para publicar informações *online*) e acessibilidade do conteúdo por eles disponibilizado. Deve ser levado em conta que *Prosumers* não são profissionais em construção de *sites*, muito menos em acessibilidade.

A inclusão de acessibilidade no User-generated Content (UGC) deve ser feita, portanto, de forma transparente ao *prosumer*. Os autores explicitam algumas formas de tornar esse processo possível:

- acessibilidade orientada a plataforma (restrição na geração, utilização de *templates*, etc);
- acessibilidade dependente do criador (permitir a ferramenta "sugerir" as questões de acessibilidade);
- acessibilidade auxiliada pela comunidade (treinamentos, compartilhamento de autoria, moderação, etc).

3.6 Avaliação

Fuertes *et al.* (2011) apresentaram o projeto da ferramenta *Hera-FFX*, um plugin para o navegador *Firefox*. O projeto tem por objetivo eliminar as limitações da ferramenta *Hera 1.0* (Sidar, 2003) e estender sua funcionalidade, realizando as avaliações utilizando o WCAG 2.0.

O trabalho é interessante por fazer comparações com outras ferramentas de avaliação, e mostrar quais características seriam desejáveis que uma ferramenta desse tipo contivesse. Dentre estas características, podem ser citadas:

- avaliação preliminar automática;
- suporte para preenchimento manual;
- modificação da página de apresentação;
- exibição do código anotado;
- avaliação de páginas locais;
- avaliação de páginas com acesso restrito;
- avaliação de renderização de páginas;
- geração de relatórios;
- suporte para treinamento;
- capacidade de multi-sessão;
- flexibilidade.

Vigo e Brajnik (2011) trataram de questões relativas a métricas de acessibilidade. Existem muitas métricas propostas na literatura, tais como Unified Web Evaluation Methodology (UWEM) (WAB Cluster, 2009), Web Accessibility Quantitative Metric (WAQM) (Vigo *et al.*, 2007), *A3* (Buhler *et al.*, 2006), T^1 (Sirithumgul *et al.*, 2009), Web Accessibility Barriers (WAB) (Martínez *et al.*, 2009), Page Measurement (PM) (Bailey e Burd, 2007), entre outras. Os autores questionam qual é a "qualidade" de cada uma delas. Por qualidade, pode-se considerar a validade, confiabilidade, sensibilidade, complexidade, entre outros parâmetros. Esses parâmetros devem ser considerados para determinar em que cenário elas devem ser usadas (validadores desktop, validadores *online*, *engine* de busca, etc). O motor de busca do *Google* já utiliza métricas de acessibilidade para "rankear" os sites (essa métrica não é de conhecimento público) (Google, 2011).

Algumas métricas são difíceis de implementar e testar por ferramentas automáticas. Os autores reforçam que há vários aspectos que devem ser levados em conta ao realizar um estudo comparativo entre as métricas. Contudo, mesmo estando abaixo de um comportamento ótimo, as métricas apresentadas na literatura que tiveram melhor comportamento

em relação ao critério *validez* foram: WAQM, PM e WAB. A métrica PM não teve um bom desempenho em relação ao critério *adequação*.

Brajnik (2009) questionou a validade das diretrizes de acessibilidade. O principal questionamento do trabalho é: um mesmo conjunto de diretrizes de acessibilidade produzirá os mesmos resultados, se conduzido por pessoas ou ferramentas diferentes?

Mesmo que a resposta para a pergunta anterior seja positiva, pode-se perguntar qual é a qualidade dos métodos de avaliação utilizados e o quão "testáveis" são as diretrizes de acessibilidade.

O autor concluiu que as diretrizes do WCAG 1.0 são mais confiáveis do que as diretrizes do WCAG 2.0. Ainda assim, o nível de confiabilidade das diretrizes do WCAG 1.0 e 2.0 não pode ser considerada alta, já que este nível não ultrapassou 80%.

Freire (2012) promoveu um estudo com um grupo específico de usuários com deficiência (cegos, parcialmente cegos e dislexos), verificando quais os tipos de dificuldades os mesmos possuem ao acessar os recursos da *internet*, como os usuários são afetados por estes problemas e quais são suas causas técnicas. As principais perguntas efetuadas pelo autor foram:

- quais são as características principais dos problemas de acessibilidade encontrados pelo grupo de usuários supracitado quando tentam acessar os *sites*;
- qual a relação entre utilizar uma medição de acessibilidade utilizando uma abordagem baseada no usuário e uma medição técnica baseada nas diretrizes do WCAG 1.0 e 2.0.

Para isto, foram escolhidos alguns *sites* com diferentes níveis de conformidade com o WCAG 1.0 e medidos por ferramentas de avaliação de acessibilidade, sendo estes dados confrontados com a medição baseada no usuário utilizando os usuários com necessidades especiais, mapeando os problemas dos usuários em diretrizes técnicas.

O estudo mostrou que os *sites* possuem variados níveis de barreiras, dependendo da deficiência que o usuário possui. Além disso, não há uma correlação significativa entre as classificações das gravidades de problemas de usuários e os níveis de prioridades associadas aos *checkpoints* e critérios de sucesso do WCAG 1.0. E, principalmente, *sites* com altos níveis de conformidade técnica não implicam em *sites* isentos de problemas encontrados pelo usuário.

3.7 Processo de Desenvolvimento

Bonacin *et al.* (2010) propuseram um processo de desenvolvimento inclusivo para aplicações eGov, utilizando semióticas organizacionais. Utilizando o estudo de caso do projeto Cidade Digital (Bonacin *et al.*, 2006), os autores adotaram o conjunto de métodos conhecido como Problem Articulation (PAM), para coletar os dados e construir os artefatos. Como

resultado, os autores delinearam um modelo de processo para o desenvolvimento de sistemas e-Gov inclusivos, possuindo este cinco princípios gerais:

- promover a participação dos cidadãos e outros *stakeholders* no acesso universal e valores de *design*;
- compartilhar o entendimento do contexto social da cidade;
- incluir mais que apenas detalhes técnicos no desenvolvimento de serviços e-Gov;
- promover a interação e interoperatividade entre agências governamentais;
- promover a qualidade de serviço, que incorpora uma ampla definição de acessibilidade e ambientes inclusivos.

Como lições aprendidas, os autores elencaram:

- a identificação de *stakeholders* se torna um problema quando o público alvo é uma população inteira;
- a identificação de *stakeholders* não é suficiente. Estratégias e técnicas precisam ser definidas para promover a participação de pelo menos alguns *stakeholders* considerados chave durante todo o processo;
- discussões exigem estruturação cuidadosa, devido à natureza complexa do e-Gov, pois acessibilidade nesse contexto normalmente envolve muitos problemas complexos, incluindo tecnológicos, políticos, legais e sociais.

Bittar *et al.* (2013) desenvolveram uma abordagem para prover suporte e facilitar o uso de acessibilidade durante o desenvolvimento de aplicações *web*. A abordagem foi proposta para solucionar os problemas identificados pela falta de consideração de questões relacionadas a acessibilidade para *web* e foi baseada em princípios teóricos (conceitos envolvidos) e em resultados de estudos práticos (estudos empíricos), utilizando Design Rationale (DR) como suporte. A abordagem é estruturada em três eixos, onde atividades relacionadas são agrupadas. Os eixos são: (1) Treinamento em acessibilidade, (2) Gerenciamento de decisões e (3) Desenvolvimento e Ferramentas.

Os autores efetuaram um estudo experimental com seis grupos, usando a abordagem sugerida. A pesquisa demonstrou que os grupos usando a aborgagem apresentaram melhores resultados que os grupos que não utilizaram a abordagem. Contudo, não usar a abordagem não implica necessariamente em um mau desenvolvimento.

3.8 Acessibilidade no Processo de Desenvolvimento

Melo e Baranauskas (2006) consideram importante apoiar o desenvolvimento de sistemas acessíveis utilizando métodos e técnicas que possam explicitá-los e representá-los. Com

o objetivo de amparar o desenvolvimento de *software* seguindo padrões e *frameworks* de desenvolvimento já existentes, Maia (2010) propôs o MTA.

O modelo proposto sugere tarefas de acessibilidade a serem empregadas nos Subprocessos do Processo de Desenvolvimento da Norma ISO/IEC 12207 (ISO/IEC, 1998). A proposta apresentada é que, alterando e adaptando esses Subprocessos específicos, a acessibilidade do produto final seja melhorada.

3.8.1 ISO/IEC 12207

A norma ISO/IEC 12207 (IEEE/EIA, 1998) tem como objetivo prover um padrão para desenvolvimento e acompanhamento dos processos do ciclo de vida de softwares. Os processos são separados por fundamentais, de apoio, organizacionais e de adaptação.

Os processos são divididos em atividades ou Subprocessos e esses, por sua vez, são divididos em tarefas. É importante mencionar que a norma não descreve as técnicas específicas utilizadas para a realização das atividades, mas sim um *framework* que permite planejar e entender o processo de desenvolvimento do software.

Apesar de a norma poder ser utilizada parcialmente, algumas atividades dos processos se relacionam com outros processos. Pode-se citar, por exemplo, tarefas do Processo de Desenvolvimento que requeiram a documentação de suas saídas e, portanto, o Processo de Documentação deveria ser utilizado (Maia, 2010).

O MTA foi desenvolvido focado no *Processo de Desenvolvimento* da norma ISO/IEC 12207. Neste processo, encontram-se os seguintes Subprocessos:

1. Elicitação dos requisitos do sistema;
2. Análise dos requisitos do sistema;
3. Projeto Arquitetural do sistema;
4. Análise de Requisitos do *software*;
5. Projeto de *software*;
6. Construção do *software* (código e teste de unidade);
7. Integração do *software*;
8. Teste do *software*;
9. Integração do sistema; e
10. Teste do sistema.

É importante ressaltar a diferença de sistema e *software*, que constam na descrição dos subprocessos. Sistema trata do escopo da solução, incluíndo estímulos externos, variantes

tecnológicas, entre outros. Esse sistema é dividido em partes menores (componentes de *software*), onde as funcionalidades da solução são especificadas e construídas. Por fim, estes componentes de *software* são reunidos para construírem a solução, novamente referenciada aqui como sistema.

3.8.2 MTA integrado ao Processo de Desenvolvimento - ISO/IEC 12207

Em cada Subprocesso do Processo de Desenvolvimento, foram inseridas uma ou mais tarefas de acessibilidade. A Tabela 3.1 relaciona os Subprocessos e as tarefas de acessibilidade relacionadas. Maia (2010) descreve os Subprocessos e como as tarefas de acessibilidade devem ser desenvolvidas.

O MTA foi idealizado com o objetivo de guiar o processo de desenvolvimento desde as fases iniciais para que a aplicação que está sendo desenvolvida seja acessível, de forma a evitar o retrabalho ocasionado pelas correções de acessibilidade realizadas somente na fase de testes, como acontece tradicionalmente. Segundo Maia (2010), é possivel resumir o MTA como um modelo de desenvolvimento, baseado na norma ISO/IEC 12207, aplicável ao WCAG, que permite o apoio ferramental ao modelo.

3.9 Considerações Finais

A pesquisa bibliográfica mostrou que o tema acessibilidade, inserido no contexto de desenvolvimento *web*, possui várias possibilidades de pesquisa. O tema é amplo, novo e sua importância tem sido reforçada a cada dia.

As áreas de pesquisa podem ser desenvolvidas em toda as etapas de desenvolvimento do produto (desde a análise de requisitos até a manutenção do sistema). As pesquisas podem ser aplicadas em nichos específicos, como *hardware*, tecnologias assistivas e, principalmente, projetos de arquitetura, navegação e interface dos produtos. Contudo, notou-se que as pesquisas são esparsas, e tentam endereçar problemas específicos, havendo pouca integração entre os trabalhos pesquisados.

Enquanto outras áreas já adaptam *frameworks* e processos de desenvolvimento de *software* com requisitos específicos (por exemplo, segurança da informação (Belani *et al.*, 2009)), os requisitos de acessibilidade ainda são negligenciados no processo de desenvolvimento de sistemas e no escopo de Engenharia de *Software*.

No próximo capítulo serão apresentadas abordagens para permitir a integração de requisitos de acessibilidade ao processo de desenvolvimento de *software*, especificando como a rastreabilidade dos requisitos é alcançada e como as técnicas de implementação são vinculadas, e dentre as abordagens possíveis, quais foram as escolhidas para a construção da ferramenta Computer-Aided Software Engineering (CASE) proposta por este trabalho.

Subprocessos	**Tarefas de acessibilidade**
1. Elicitação dos requisitos do sistema	1.1 Identificar os requisitos de acessibilidade do sistema
2. Análise de requisitos do sistema	2.1 Especificar os requisitos de acessibilidade do sistema
	2.2 Avaliar os requisitos de acessibilidade do sistema
3. Projeto arquitetural do sistema	3.1 Alocar os requisitos de acessibilidade aos elementos do sistema
	3.2 Avaliar o projeto arquitetural do sistema com relação aos requisitos de acessibilidade
4. Análise de requisitos do software	4.1 Estabelecer os requisitos de acessibilidade do software
	4.2 Avaliar os requisitos de acessibilidade do software
5. Projeto de software	5.1 Projetar as interfaces externas acessíveis
	5.2 Realizar o projeto navegacional acessível
	5.3 Avaliar acessibilidade do projeto de software
6. Construção do Software (código e teste de unidade)	6.1 Especificar técnicas para implementação da acessibilidade da interface e do código
	6.2 Codificar cada unidade de software de acordo com as técnicas de acessibilidade
	6.3 Planejar teste de acessibilidade para cada unidade de software
	6.4 Executar testes de acessibilidade de cada unidade de software
7. Integração do software	7.1 Planejar teste de acessibilidade do software integrado
8. Teste do software	8.1 Conduzir testes de acessibilidade do software
	8.2 Avaliar o resultado do teste de acessibilidade
9. Integração do sistema	9.1 Realizar testes de acessibilidade no sistema
	9.2 Avaliar os resultados dos testes de acessibilidade do sistema
10. Teste do sistema	10.1 Certificar a conformidade com os requisitos do sistema

Tabela 3.1: Subprocessos e tarefas de acessibilidade do MTA Maia (2010)

Capítulo 4

Integração de Requisitos de Acessibilidade ao Processo de Desenvolvimento de *Software*

O MTA, como extensão de um processo de desenvolvimento de *software*, não especifica qual é a maneira correta de executar as tarefas e atividades propostas. É um desafio partir dos requisitos de acessibilidade e conseguir entregar um produto acessível. Por isso, é fundamental haver um método de rastreabilidade desses requisitos, de forma que exista uma garantia de que a implementação está sendo feito de forma correta.

Os requisitos de acessibilidade devem ser propagados desde o início do processo (levantamento e confecção do documento de requisitos) até as etapas finais de codificação e testes de forma consistente.

Para o trabalho proposto, foi definido como escopo apenas os Subprocessos 4 (Análise de Requisitos de Software), 5 (Projeto de Software) e 6 (Construção do Software) e suas respectivas tarefas de acessibilidade propostas no MTA. Essa abordagem foi adotada pois, embora as definições de acessibilidade já comecem no Subprocesso 1, é no Subprocesso 4 que os requisitos são "lapidados" e o artefato de entrada das fases de projeto é gerado.

Como o escopo deste trabalho diz respeito a rastreabilidade dos requisitos de acessibilidade e suas técnicas de implementação, as tarefas de testes, presentes no Subprocesso 6, não serão levadas em consideração.

A seguir, serão apresentados os relacionamentos entre os Subprocessos citados com o rastreio dos requisitos de acessibilidade.

4.1 Subprocesso 4 - Análise de Requisitos de Software

O especialista em acessibilidade é fundamental em todas as etapas descritas a seguir, como já reforça o modelo MTA. Ele deve conseguir identificar requisitos funcionais e não funcionais de acessibilidade. O requisito de acessibilidade pode não ser encontrado explicitamente neste momento, mas a identificação posterior exigirá um refatoramento do modelo.

Requisitos funcionais e não funcionais são abordados no Subprocesso 4, ponto de partida deste trabalho no modelo MTA. O objetivo do Subprocesso 4 é estabelecer os requisitos dos elementos de software do sistema. Elementos aqui são considerados interface e código (Maia, 2010).

O MTA insere duas tarefas de acessibilidade no Subprocesso Análise de Requisitos de *Software*, conforme mostrado na Figura 4.1.

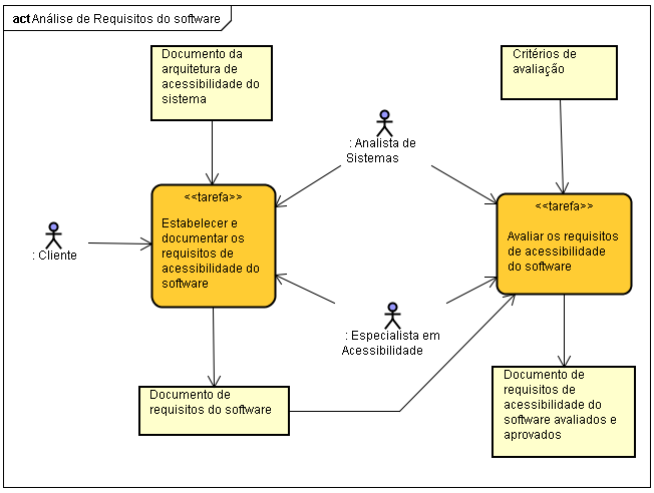


Figura 4.1: Tarefas para o subprocesso de análise de requisitos do software (Maia, 2010)

É importante ressaltar que os requisitos já são parcialmente coletados em Subprocessos anteriores, mas é apenas nesse Subprocesso que os requisitos de interface e código são considerados. É nesse Subprocesso também que irá originar o artefato para as fases de projeto e codificação.

A saída da tarefa **Avaliar os requisitos de acessibilidade de software** é processada, gerando um novo artefato (Documento de requisitos de acessibilidade do software avaliados e aprovados). Neste trabalho, propõe-se que mais um artefato seja gerado, sendo um documento de requisitos em um formato legível por máquina, não ambíguo e que possa ser gerenciado por uma ferramenta CASE, que será utilizado nos passos subsequentes para rastreamento dos requisitos de acessibilidade. Foi adotado neste trabalho o formato XML Metadata Interchange (XMI)/XML, padrão da Object Management Group (OMG) definido para troca de informações e metadados entre as ferramentas de modelagem, principalmente por ser o principal formato para descrever elementos UML, apesar

de ser possível utilizar outros formatos para atingir o mesmo objetivo.

É importante definir como os requisitos serão rastreados dentro do processo de desenvolvimento. O método de rastreabilidade utilizado será uma matriz de rastreabiliade de requisitos (guo *et al.*, 2009a). A matriz é um artefato bidimensional que relaciona dois elementos distintos através de seu produto cartesiano e os pontos de intersecção indicam quais elementos estão relacionados. A Figura 4.2, por exemplo, relaciona Requisitos *versus* Casos de Teste. Outros relacionamentos podem ser usados na matriz de rastreabilidade, como será demonstrado posteriormente neste trabalho.

Sample traceability matrix

Requirement Identifiers	Reqs Tested	REQ1 UC 1.1	REQ1 UC 1.2	REQ1 UC 1.3	REQ1 UC 2.1	REQ1 UC 2.2	REQ1 UC 2.3.1	REQ1 UC 2.3.2	REQ1 UC 2.3.3	REQ1 UC 2.4	REQ1 UC 3.1	REQ1 UC 3.2	REQ1 TECH 1.1	REQ1 TECH 1.2	REQ1 TECH 1.3
Test Cases	321	3	2	3	1	1	1	1	1	1	2	3	1	1	1
Tested Implicitly	77														
TC1.1.1	1	x													
TC1.1.2	2		x	x											
TC1.1.3	2	x											x		
TC1.1.4	1			x											
TC1.1.5	2	x												x	
TC1.1.6	1		x												
TC1.1.7	1			x											
TC1.2.1	2				x		x								
TC1.2.2	2					x		x							
TC1.2.3	2							x	x						
TC1.3.1	1										x				
TC1.3.2	1										x				
TC1.3.3	1											x			
TC1.3.4	1											x			
TC1.3.5	1											x			
etc															
TC5.6.2	1														x

Figura 4.2: Exemplo de matriz de rastreabilidade de Requisitos *versus* Casos de Teste (Jain, 2013)

Conforme os artefatos vão sendo gerados/atualizados, a matriz também é gerada e atualizada para se adequar à evolução do projeto. A ferramenta CASE pode gerar o esqueleto da matriz de rastreabilidade para os requisitos de acessibilidade.

O método proposto pode ser classificado como um método estático de rastreabilidade. Embora existam métodos dinâmicos (Cleland-Huang *et al.*, 2005), para estes métodos, mudanças no artefatos refletem mudanças na matriz. Como neste trabalho nem todos os artefatos são utilizados, e é necessário gerar explicitamente a matriz através de um comando do *Eclipse*, métodos dinâmicos não serão considerados.

A UML normalmente é utilizada para discriminar os artefatos posteriores ao documento de requisitos, como casos de uso, diagramas de classe, diagramas de sequência, diagramas de atividade, entre outros. É comum que tais diagramas sejam descritos, através de ferramentas CASE, em XML. Portanto, é possível utilizar o estereótipo textual *Nota* da UML para associar os requisitos aos elementos dos artefatos UML. As notas não possuem valor semântico real para o modelo, mas servem para adicionar mais informações sobre um objeto ou situação específica e podem ser ancoradas a um elemento UML para mostrar que tal nota está associada a um contexto específico. É possível verificar a utilização de notas UML para rastreabilidade no trabalho de Lee *et al.* (2009), apesar do contexto ser diferente.

Definindo uma sintaxe para as notas é possível associar os elementos UML dos artefatos

aos requisitos. A princípio, isso pode ser feito de duas formas:

1. A ferramenta CASE leria os modelos (XML), efetuaria o *parser* e incluiria as notas no arquivo XML;
2. O especialista em acessibilidade, através de ferramentas próprias de modelagem utilizadas no projeto, incluiria as notas, que posteriormente seriam associados utilizando a ferramenta CASE descrita anteriormente.

O principal problema da primeira abordagem é que as ferramentas de modelagem geram modelos UML com formato XML variado e quase sempre incompatíveis, isso quando permitem a exportação do modelo para o formato XML (Os arquivos analisados durante o desenvolvimento deste trabalho para se obter esta conclusão foram gerados pelas ferramentas *Astash Professional* (Astah, 2013), Eclipse Modeling Framework (EMF) (Eclipse, 2013b) e *Visual Paradigm* (Visual Paradigm, 2013)). Assim, seria preciso considerar uma ou mais ferramentas de modelagem específicas devido a quantidade de ferramentas de modelagem disponíveis no mercado. O problema da segunda abordagem é que o especialista em acessibilidade precisa utilizar a ferramenta CASE de modelagem e incluir as notas na sintaxe específica.

Uma alternativa à utilização de notas é a utilização de um arquivo externo de mapeamento entre os requisitos e os modelos. É uma abordagem relativamente simples se a estrutura dos arquivos de requisitos e dos modelos for previamente conhecida. Assim, é possível associar os requisitos utilizando os identificadores da estrutura (por exemplo, se os arquivos forem RDF, é possível utilizar o atributo *rdf:ID*). É possível também que a ferramenta de gerenciamento de requisitos permita que se faça a associação dos requisitos e dos modelos, caso a ferramenta conheça a descrição interna dos modelos. Esta abordagem foi utilizada neste trabalho: os *plugins* utilizados para o gerenciamento de requisitos e modelagem utilizam a formatação de arquivos do EMF, que por sua vez são baseados no RDF, sendo portanto compatíveis entre si.

4.2 Subprocesso 5 - Projeto de Software

O objetivo do Subprocesso 5 é fornecer um projeto onde os requisitos do software possam ser implementados e verificados (Maia, 2010).

Neste subprocesso, o MTA insere três tarefas de acessibilidade, conforme mostrado na Figura 4.3.

As três tarefas deste Subprocesso são de extrema importância e impactam diretamente na acessibilidade do produto final. O artefato *Documento de requisitos de acessibilidade do software avaliados e aprovados* gerado pela ferramenta CASE no formato XML/XMI, ainda sem as especificações de técnicas de implementação de acessibilidade, obtido no Subprocesso anterior passará por cada tarefa deste Subprocesso, de forma que, além da saída da tarefa **Avaliar a acessibilidade do projeto de software** e gerar o artefato

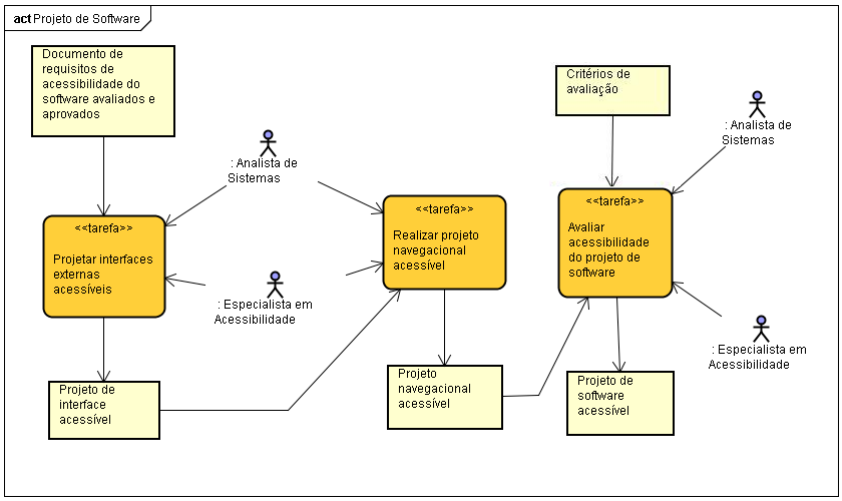


Figura 4.3: Tarefas para o Subprocesso de projeto de software (Maia, 2010)

Projeto de software acessível, esta tarefa também gerará um artefato de projeto de *software* no formato XML/XMI, com elementos de acessibilidade embutidos.

A proposta deste trabalho enseja que as Subtarefas das tarefas pertencentes a esse Subprocesso já comecem a contar com critérios de sucesso do documento de diretrizes de acessibilidade escolhido (por exemplo, o WCAG). Desta forma, torna-se mais fácil relacionar os pontos críticos de acessibilidade do projeto de interface e navegação nos passos posteriores, principalmente porque um software automatizado não consegue fazer algumas verificações levando em consideração o contexto semântico (por exemplo, ordem de navegação), e esses pontos críticos devem ser revisados posteriormente.

Além disso, é necessário esclarecer que o padrão de conformidade para o projeto é definido na Tarefa do Subprocesso 1 (Elicitação dos Requisitos do Sistema) (Maia, 2010). Os padrões e ferramentas disponíveis levam em consideração o reconhecido padrão WCAG 2.0. Contudo, a necessidade de conformidade com outros padrões de acessibilidade leva a necessidade de um monitoramento constante por parte do especialista em acessibilidade, pois nem sempre as técnicas para a implementação do que é cobrado no padrão estão disponíveis (por exemplo, é possível que um *software* precise estar em conformidade com o modelo *Section 508* (United States Government, 2009), mas esse padrão não explicita o que deve ser feito para que o nível de acessibilidade pretendido seja alcançado).

O especialista em acessibilidade deve associar os elementos de interface e navegação utilizando a ferramenta CASE aos requisitos provenientes da etapa anterior. Além disso é possível mapear, se aplicável, técnicas de implementação de acessibilidade. Um projeto útil para essa finalidade é *AEGIS Ontology* (Aegis, 2013), que define uma ontologia de acessibilidade na *Web*. Ontologias são utilizadas para permitir o compartilhamento e reutilização de informações de um determinado domínio, definindo um vocabulário comum em que a informação é compartilhada e representada (Gruber *et al.*, 1993). A ontologia descrita define o domínio de acessibilidade na Internet, mapeando os conceitos de acessibilidade e especificando como estes conceitos podem ser utilizandos em um cenário de acessibilidade.

Este projeto é apoiado pelo *Accessible Consortium* (Accessible Consortium, 2013a), que disponibiliza uma página para a consulta das novidades da iniciativa (Accessible Consortium, 2013b).

A ontologia versão 1.0 é descrita no formato Web Ontology Language (OWL), padrão definido pela W3C em 2004 (W3C, 2013c). O formato OWL já possui uma versão candidata à 2.0, lançada em 2012 (W3C, 2013b), que até o momento de escrita deste trabalho não foi oficializada como versão oficial. As versões aqui descritas em nada influem na versão do WCAG mapeado na ontologia utilizada, que é a versão 2.0 do mesmo.

O projeto vai além de realizar o mapeamento dos conceitos de acessibilidade e cenários. Os arquivos disponibilizados mapeiam o modelo WCAG 2.0 (incluindo técnicas de implementação, critérios de sucesso e falha, etc), leitores de tela, navegadores textuais, lentes de aumento, WAI/ARIA, entre outros. É possível navegar pela ontologia via navegador [1], ou explorar os arquivos confortavelmente através do software *Protegé* (Noy *et al.*, 2001).

Ao final deste Subprocesso, a ferramenta CASE já deve ter as associações dos requisitos de acessibilidade com os itens de interface e navegação (já incluindo a associação com o WCAG 2.0), bem como os outros artefatos modelados no Subprocesso anterior. A matriz de rastreabilidade pode ser gerada novamente, dessa vez mais completa, pois mais dados estão disponíveis neste ponto do processo.

4.3 Subprocesso 6 - Construção do Software

O objetivo do Subprocesso 6 é produzir unidades de software executáveis que apropriadamente refletem o projeto de software (Maia, 2010).

O MTA insere quatro tarefas de acessibilidade neste subprocesso, conforme mostrado na Figura 4.4.

A Tarefa 6.1 (Especificar Técnicas para Implementação da Acessibilidade da Interface e do Código) visa explicitar as técnicas de implementação de acessibilidade, refletidas no projeto de software. Como dito anteriormente, o padrão de conformidade já deve ter sido escolhido no Subprocesso 1. Apesar dos padrões de acessibilidade serem diferentes em essência, existe uma grande intersecção no que se refere as técnicas utilizadas para implementação efetiva da acessibilidade no produto. As regras para implementação do padrão e-Mag 3.0 são muito parecidas com as do padrão WCAG 2.0, e a tendência é que haja uma homogeneização dos esforços em desenvolver produtos acessíveis, principalmente com a evolução do HTML 5.

Os modelos WCAG 2.0 e e-Mag 3.0 já possuem técnicas de acessibilidade presentes em seus respectivos documentos. Esse é o momento de relacionar as especificidades do projeto de software com as técnicas de acessibilidade do modelo escolhido. Caso nenhum modelo conhecido ou algum modelo não usual tenha sido escolhido, as técnicas de implementação de acessibilidade deverão ser explicitamente relacionadas no projeto e, para isso, é fundamental o auxílio de um especialista em acessibilidade.

[1] http://160.40.50.89/Accessible_Ontology/Version5.1/AccessibleOntologyOWLDoc/index.html

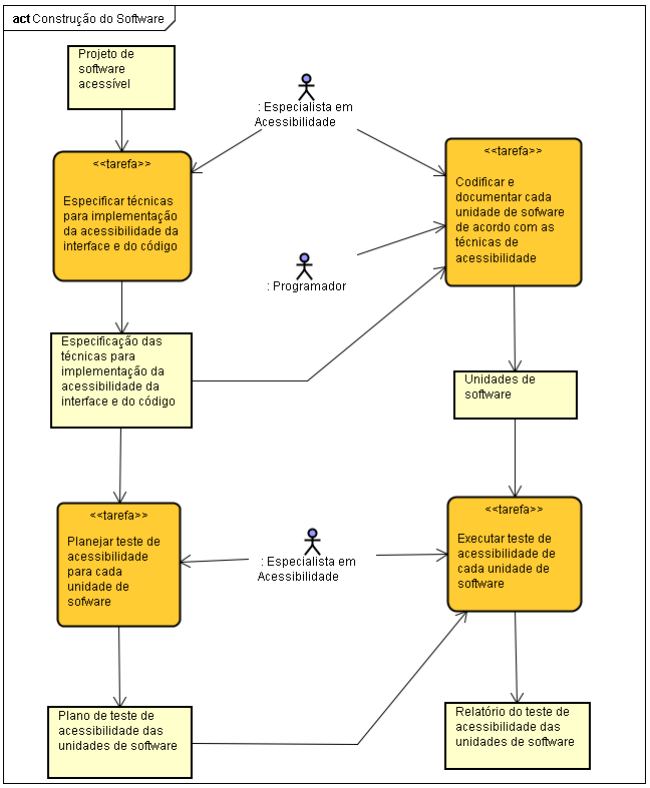


Figura 4.4: Tarefas para o subprocesso de construção do software (Maia, 2010)

Por conveniência, as técnicas utilizadas serão as do WCAG 2.0, pelo fato de já estarem mapeadas nos arquivos OWL descritos anteriormente.

Devido ao escopo do trabalho, apesar de serem extremamente importantes, as tarefas **Planejar teste de acessibilidade para cada unidade de software** e **Executar teste de acessibilidade de cada unidade de software** não foram consideradas.

A Tarefa 6.2 (Codificar e documentar cada unidade de software de acordo com as técnicas de acessibilidade) recebe os resultados dos passos anteriores, pois o projeto já contém os pontos críticos de acessibilidade explicitados. Ferramentas CASE, neste ponto, comumente geram código (*stub*), utilizando artefatos como diagramas de classe, de sequência e casos de uso. O código gerado já poderia conter traços de documentação associados aos requisitos de acessibilidade.

Uma maneira imediata de documentar código é adicionando comentários padronizados, de forma que os requisitos de acessibilidade possam ser localizados. Dependendo da linguagem de programação utilizada, podem ser utilizadas anotações para referenciar os modelos (a linguagem *Java* permite o uso de anotações), mas como este método é dependente de uma linguagem de programação que forneça tal funcionalidade e devido a complexidade envolvida neste processo, esta abordagem não será utilizada.

Para incorporar a rastreabilidade dos requisitos no código gerado pelas ferramentas em forma de comentário de código, tem-se as seguintes abordagens:

1. a ferramenta de modelagem gera os códigos, e posteriormente, a ferramenta CASE proposta incluiria os comentários de rastreabilidade;

2. a ferramenta CASE proposta substituiria o gerador de código, realizando todo o trabalho e gerando o código já comentado.

3. a gerador de código é customizado para que gere o código e os comentários personalizados no mesmo instante.

A primeira abordagem implica em avaliar todo os arquivos *Java* para encontrar o local exato onde o comentário deve ser posicionado. Além de ser necessário construir um *parser* complexo para essa situação, pode ser difícil encontrar o caminho de retorno (código *Java* para o modelo UML).

A segunda abordagem implica na construção total de uma ferramenta complexa que está fora do escopo deste trabalho.

A terceira abordagem, que por sua vez foi a escolhida, foi customizar o gerador de código, pois o código fonte deste gerador estava disponível e o gerador de código usa *templates* de construção. Desta forma, as intervenções na ferramenta foram mínimas, e pouco código adicional precisou ser incorporado.

Na Figura 4.5 é mostrada a abordagem escolhida que, partindo da Engenharia de Requisitos e chegando na construção do *software*, proverá a rastreabilidade dos requisitos e das técnicas de implementação de acessibilidade. A próxima seção explicitará as ferramentas, técnicas e métodos envolvidos para construir a ferramenta de apoio proposta por este trabalho.

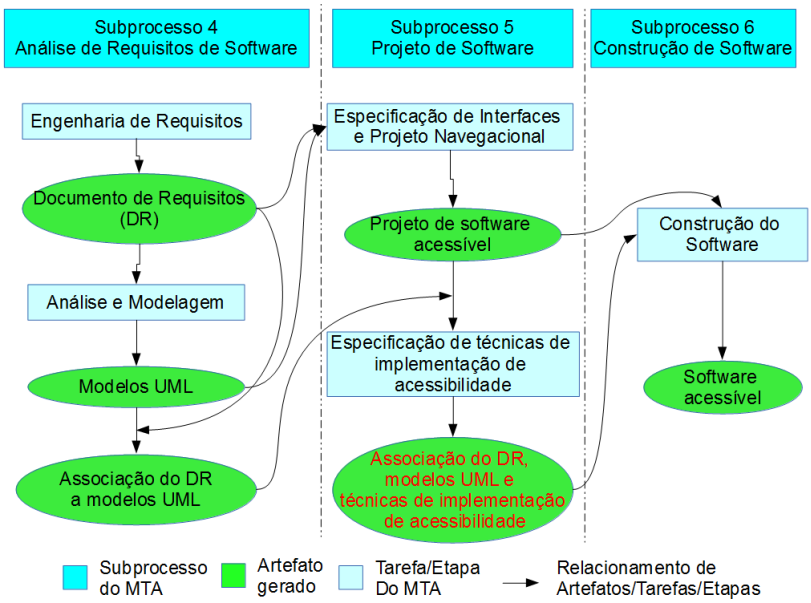


Figura 4.5: Detalhamento dos Subprocessos do MTA para prover a rastreabilidade dos requisitos de acessibilidade de acordo com a abordagem adotada neste trabalho

4.4 Requisitos da ferramenta de rastreabilidade de requisitos de acessibilidade

É desejável que uma ferramenta CASE que apóie a rastreabilidade dos requisitos de acessibilidade permita:

- Associar os requisitos de acessibilidade e os modelos UML. A ferramenta deve permitir a associação bidirecional, ou seja, deve ser possível partir dos requisitos e chegar nos modelos UML e vice-versa;
- Visualizar e gerenciar as associações descritas anteriormente. A ferramenta deve permitir que as associações sejam gerenciadas, ou seja, que tais associações sejam inseridas, atualizadas e removidas;
- Vincular as técnicas de implementação para os elementos associados anteriormente. Para cada vinculação de requisito e modelo glsuml, a ferramenta deve permitir a associação de uma ou mais técnica de implementação, com informações relevantes que serão recuperadas posteriormente;
- Gerar a matriz de rastreabilidade dos elementos associados. A ferramenta deve permitir a geração da matriz de rastreabilidade, de forma automática, mostrando de forma explícita o relacionamento entre os requisitos, modelos UML e técnicas de implementação de acessibilidade;
- Gerar código fonte com as informações necessárias que permita efetuar a rastreabilidade dos requisitos, modelos e técnicas de implementação de acessibilidade. A ferramenta deve, partindo as associações entre requisitos, modelos UML e técnicas de implementação, gerar códigos *stub* preparados para que estas informações sejam recuperadas posteriormente;
- Recuperar as informações de rastreabilidade entre as fases de engenharia de requisitos e codificação. O código fonte precisa ter informações embutidas que permitam referenciar os requisitos, modelos UML e técnicas de implementação de acessibilidade, e uma vez de posse dessas informações

4.5 Alternativas disponíveis

Já existem iniciativas, principalmente corporativas, que permitem agregar os requisitos levantados aos artefatos do processo de desenvolvimento. Hovater (2008) mostra como construir relatórios de rastreabilidade usando os programas *IBM Rational Software Architect* (IBM, 2013d), *IBM Rational RequisitePro* (IBM, 2013c) e Business Intelligence Reporting Tools (BIRT) para *WebSphere* (IBM, 2013e).

O software *Enterprise Architect* da empresa *Sparx Systems* (Sparx Systems, 2013) permite utilizar diagramas de requisitos[2], que são extensões dos diagramas tradicionais da UML, permitindo a rastreabilidade do modelo[3]. Contudo, não foi encontrado na literatura trabalhos que tratem especificamente da rastreabilidade dos requisitos de acessibilidade dentro do processo de desenvolvimento de *software*. Além disso, não foi encontrado como essas alternativas permitem, partindo do código fonte da solução, recuperar as informações de rastreabilidade dos requisitos.

4.6 Escolha das Ferramentas e Tecnologias

A proposta deste trabalho é demonstrar a rastreabilidade dos requisitos de acessibilidade no processo de desenvolvimento de *software*. Dessa forma, para implementar a abordagem apresentada na Figura 4.5, foi construída uma ferramenta CASE, nos moldes descritos na seções 4.1, 4.2 e 4.3. A seguir são elencados os principais elementos usados para a demostração desta proposta:

- MTA - Processo de Desenvolvimento de *Software* com tarefas de acessibilidade (Maia, 2010);
- *Eclipse Juno* - Integrated Development Environment (IDE);
- *Requirement Designer v0.8.0* (Eclipse Marketplace, 2013a) - *Plugin* de gerenciamento de requisitos;
- *UML Designer v2.1.0* (Eclipse Marketplace, 2013b) - *Plugin* de modelagem UML;
- *UML to Java Generator v1.0.2* (Eclipse Marketplace, 2013c) - *Plugin* de geração de código;
- *Java JRE7 e JDK1.7* (Deitel e Deitel, 2001) - Linguagem para desenvolvimento de *plugins* e código final do produto;
- ontologia para implementação das diretrizes do WCAG 2.0 Aegis (2013).

O *Eclipse* é uma plataforma madura e foi escolhido como IDE por vários motivos. Ele serve como base para diversos produtos e tecnologias baseadas em uma IDE, provendo uma Application Programming Interface (API) para facilitar a integração (desRivieres e Wiegand, 2004). O desenvolvimento de *plugins* para o *Eclipse* é feito diretamente na IDE, de forma prática e transparente, tendo ampla documentação a respeito. A linguagem utilizada para o desenvolvimento do *plugin* é a linguagem *Java*, assim como o código gerado pelo *plugin* de exportação.

[2] http://www.sparxsystems.com/enterprise_architect_user_guide/modeling_languages/requirements_diagram.html

[3] http://www.sparxsystems.com/enterprise_architect_user_guide/navigate_search_and_trace/traceability.html

O pacote de modelagem inicialmente instalado foi composto pelos *plugins* EMF propostos pelo *Eclipse*, notadamente o *Ecore Tools* (Eclipse, 2013d), parte integrante do Eclipse Modeling Framework Technology (EMFT). Contudo, os diagramas fornecidos por *default* não satisfizeram aos anseios de modelagem para o desenvolvimento deste trabalho, pois não contemplavam diagramas de sequência ou de casos de uso, por exemplo. Por esse motivo, o *plugin* escolhido foi o *UML Designer*, que utiliza como base o pacote EMF (portanto gera modelos EMF), mas estende os modelos existentes para que se adequem aos modelos UML na sua versão 2.4.

O *plugin* inicialmente escolhido para o gerenciamento de requisitos foi o *ProR* (Eclipse, 2013f), que por sua vez é parte do projeto Requirements Modeling Framework (RMF) e EMF do *Eclipse*. O objetivo do projeto RMF é implementar o padrão *OMG ReqIF* (OMG, 2013) em forma de modelos EMF. Contudo, o RMF se encontra atualmente na versão 0.7.1 e o *plugin* não fornecia a funcionalidade de ligação entre os modelos e os requisitos. Assim, modos alternativos de relacionamento entre os modelos e os requisitos deveriam ser implementados, tornando inviável o desenvolvimento do trabalho. Portanto, o *plugin* escolhido para o gerenciamento de requisitos foi o *Requirement Designer*, que permite realizar a associação dos requisitos com qualquer modelo EMF.

Os *plugins* de modelagem e gerenciamento de requisitos são desenvolvidos pela *Obeo* (Eclipse, 2013e), portanto, para permitir uma integração suave entre as ferramentas, o *plugin* escolhido para geração de código foi o *UML to Java Generator*, desenvolvido pela mesma empresa. Os *plugins* são disponibilizados sob a licença *Eclipse Public License v 1.0* (Eclipse, 2013c), assim como a própria IDE. Dessa forma, é possível estudar, alterar e customizar seus componentes para atingir os objetivos do trabalho.

A Figura 4.6 mostra o comportamento e relacionamento das ferramentas, tecnologias e atores envolvidos no desenvolvimento do produto.

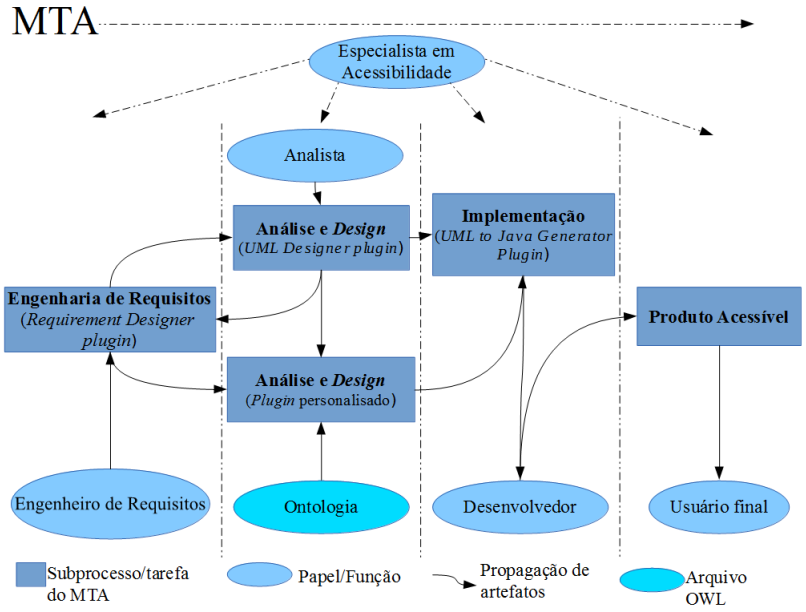


Figura 4.6: Associação das ferramentas e atores no contexto do trabalho

A interpretação da Figura deve ser feita da seguinte forma:

- O MTA é o processo de desenvolvimento e deve permear todas as fases do processo;
- O especialista em acessibilidade (*Accessibility Expert*) deve participar das fases de desenvolvimento especificadas no MTA;
- Os requisitos devem ser coletados e informados na fase de engenharia de requisitos (utilizando a ferramenta *Requirement Designer*) e os modelos e artefatos UML devem ser gerados (utilizando a ferramenta *UML Designer*). O especialista em acessibilidade deve auxiliar a alimentação destes dados, filtrando os requisitos de acessibilidade para que eles sejam associados aos modelos UML (utilizando a ferramenta *Requirement Designer*);
- A ferramenta proposta recupera as associações dos requisitos e modelos UML (apenas aos que dizem respeito à acessibilidade), permitindo que o especialista especifique as técnicas de implementação de acessibilidade para cada uma das associações. As técnicas, diretrizes, abordagens, dentre outros estão armazenadas em arquivos OWL embutidos na ferramenta;
- A ferramenta de geração de código produzirá o código *stub* a partir dos modelos UML, adicionando as referências às associações de acessibilidade quando necessário;
- Os desenvolvedores refinam o código até que o mesmo esteja apto a se tornar o produto acessível que será entregue.

4.7 Construção da Ferramenta

O *plugin* construído utiliza quatro elementos externos que precisam ser detalhados:

- O repositório dos requisitos
- O requisito
- O modelo UML
- A técnica de acessibilidade, representado pela ontologia

A Figura 4.7 mostra a execução da ferramenta *Requirement Designer* e o cadastro de algumas categorias e requisitos como exemplo.

O diagrama de classes visualizado na Figura 4.8 demonstra o núcleo de funcionamento do *plugin Requirement Designer*, que possui os dois primeiros elementos listados anteriormente (repositório dos requisitos e os requisitos). Existe um repositório (*Repository*), que pode conter zero ou mais categorias principais (*Category*). Cada categoria principal pode conter 0 ou mais requisitos (*Requirement*), e também pode conter zero ou mais

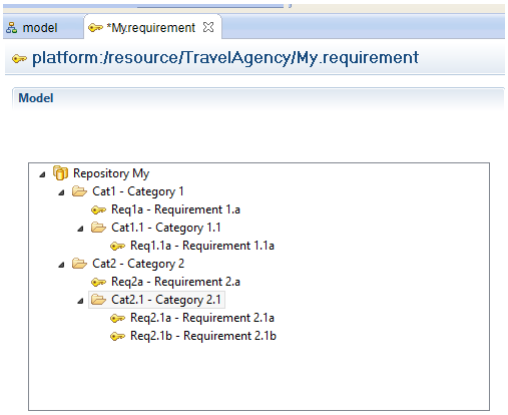


Figura 4.7: Execução da ferramenta *Requirement Designer*

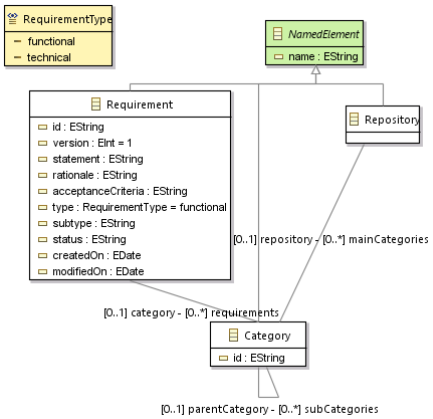


Figura 4.8: Diagrama de classes do *plugin* Requirement Designer (Eclipse Marketplace, 2013a)

subcategorias (*Category*), recursivamente. Os requisitos podem ser do tipo funcionais ou não-funcionais (neste *plugin* tratados como requisitos técnicos).

Os desenvolvedores do *plugin Requirement Designer* construíram o diagrama de classes do *plugin* com uma ferramenta baseada no modelo *Ecore* EMF (provavelmente foi utilizado o *plugin UML Designer*). Com isso, é possível construir as classes *Java* representativas do modelo, utilizando uma ferramenta de transformação, como a *Ecore Generator Model*. Na prática, significa dizer que, uma vez tendo o diagrama de classes no modelo *Ecore*[4], será possível utilizar as classes *Java* resultantes e, principalmente, que o arquivo de saída da ferramenta *Requirement Designer* possa ser lido e interpretado de forma simplificada como os objetos corretos do *plugin*.

A ferramenta para gerar os artefatos UML é a *UML Designer*. A ferramenta usa a API do *Eclipse (org.eclipse.uml2.uml)*, que por sua vez é persistida usando EMF. Por esse motivo, a ferramenta de gerenciamento de requisitos consegue gerar uma associação entre

[4]O código fonte e os modelos/diagramas pode ser baixados em https://github.com/ObeoNetwork/InformationSystem/tree/master/designs/requirement

os requisitos e qualquer elemento presente na API.

A Figura 4.9 mostra um diagrama de casos de uso de exemplo gerado pela ferramenta *UML Designer*.

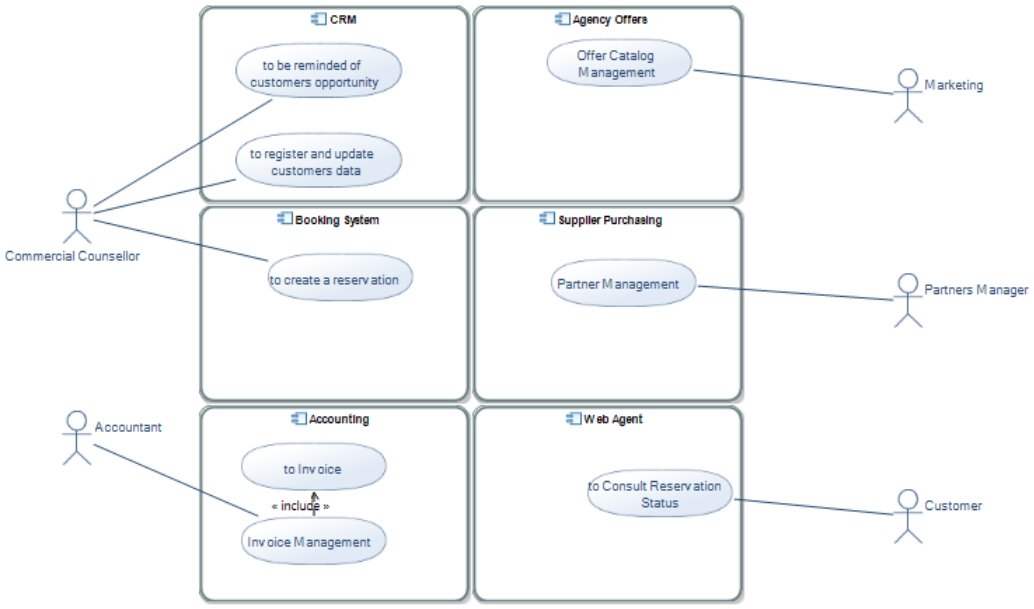


Figura 4.9: Exemplo de um diagrama de casos de uso gerado pela ferramenta *UML Designer* (Modelo UML de exemplo gerado pelo *Eclipse, projeto TravelAgency*)

A ferramenta *Requirement Designer* utiliza referências EMF (elementos *proxy*) para associar os modelos UML gerados pela ferramenta *UML Designer*, significando que os objetos referenciados não são escritos no arquivo XML de destino. Assim, o modelo pode ser alterado, e o objeto sempre estará atualizado no arquivo referenciado. Essa mesma técnica será usada no modelo de dados a ser construído pela ferramenta proposta, com exceção dos objetos referentes à ontologia de acessibilidade.

A ontologia de acessibilidade é descrita no padrão OWL, que tem por base o formato RDF. Para efetuar a leitura dos arquivos OWL, foi utilizada a *OWL API*, que por sua vez é utilizado pelo *software Protégé*, *software* este muito útil para visualizar de forma gráfica os elementos e relacionamentos descritos na ontologia. Este *software* foi usado diversas vezes para descobrir quais os elementos da ontologia e como eles deveriam ser acessados através da API. A Figura 4.10 mostra um dos arquivos da ontologia usado no trabalho, aberto para visualização na ferramenta *Protégé*.

Diante do exposto, é possível apresentar o diagrama de classes da ferramenta proposta neste trabalho, aqui denominada como *AccTrace*, visível na Figura 4.11.

O diagrama de classes deve ser interpretado da seguinte forma: a classe principal é a *AccTraceModel*, que armazena as referências para os repositórios dos requisitos (*Repository*), e também objetos de referência às associações dos requisitos, modelos e técnicas de implementação de acessibilidade (*Reference*). Esse objeto referencia um requisito (*Requirement*), um diagrama UML (*EObject*) e uma ou mais técnicas de implementação de

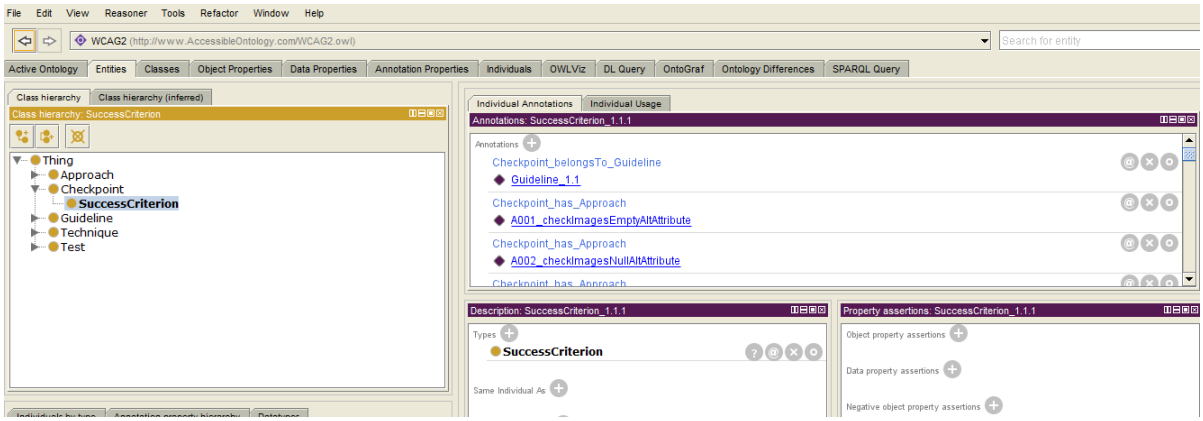


Figura 4.10: Exemplo do arquivo WCAG2.owl aberto no *software Protégé*

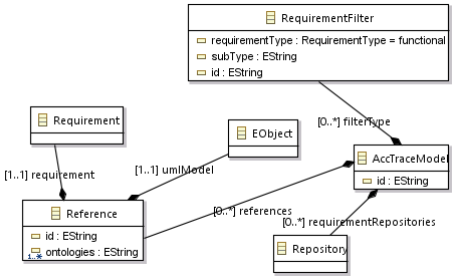


Figura 4.11: Diagrama de classes da ferramenta proposta (*AccTrace*)

acessibilidade, representadas aqui pela seleção da ontologia disponível. As referências à ontologia são persistidas através de sua Internationalized Resource Identifier (IRI) (uma generalização de Uniform Resource Identifier (URI)) em forma de *String*. Além disso, devido a possibilidade de existência de inúmeros requisitos no projeto e considerando o fato de que apenas os requisitos de acessibilidade sejam importantes para a associação à técnicas de implementação de acessibilidade, é previsto no modelo também a inclusão de filtros dos requisitos (*RequirementFilter*), para não poluir a visualização dos requisitos na ferramenta.

A ferramenta possui três visões principais, de acordo com a Figura 4.12.

No editor (*AccTrace Editor*) é possível alterar os repositórios dos requisitos e gerar as associações entre os modelos UML, requisitos e técnicas de implementação. Na visão dos requisitos (*Requirement Associations*) é possível visualizar quais requisitos associados ao modelo UML foram selecionados no editor. Na visão das técnicas já vinculadas (*Accessibility Specifications View*) é possível visualizar as técnicas de implementação já associadas, de acordo com o modelo UML selecionado no editor e o requisito de acessibilidade selecionado na visão dos requisitos. Além disso, é possível remover as técnicas de implementação associadas. As três visões são importantes para o correto funcionamento da ferramenta.

Uma vez selecionado o modelo UML e o requisito, é possível efetuar a associação da técnica de implementação de acessibilidade, clicando com o botão direito do mouse em

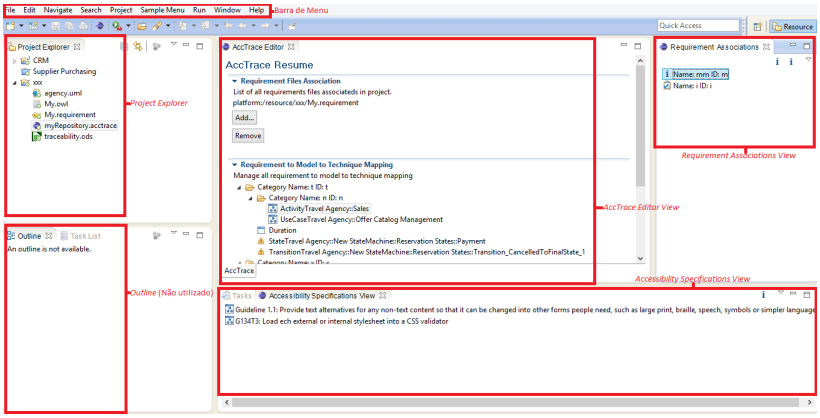


Figura 4.12: Visualização da ferramenta *AccTrace* na tela principal do *Eclipse*

cima do modelo UML, conforme demonstrado na Figura 4.13.

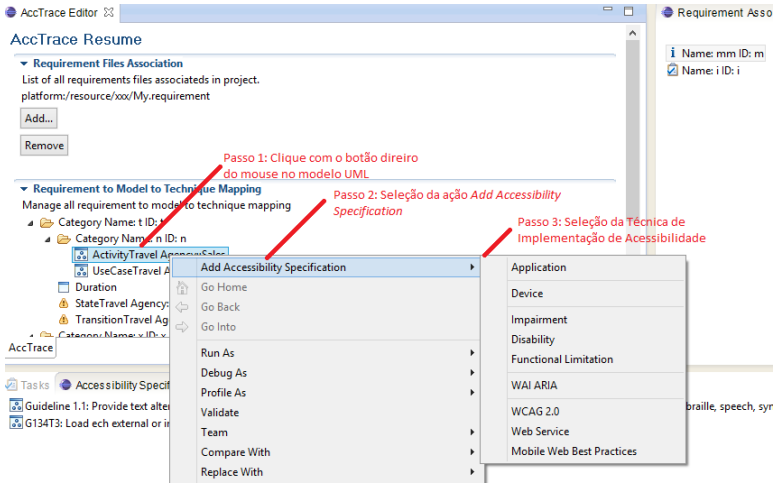


Figura 4.13: Procedimento para efetuar a associação da técnica de implementação de acessibilidade

O menu mostrado na Figura 4.13 foi projetado levando em conta a análise e o estudo da ontologia disponível, efetuado na ferramenta *Protégé*. É possível entender o relacionamento dos elementos da ontologia observando a Figura 4.14.

Para este trabalho, um dos elementos mais importantes da ontologia é o elemento denominado *WCAG 2.0*, pois é este elemento que comporta as técnicas de implementação de acessibilidade que efetivamente serão utilizadas. Os outros elementos, apesar de acessórios, podem ser utilizados para reforçar as técnicas já referenciadas.

O elemento *WCAG 2.0* presente na ontologia possui 5 grupos que podem ser escolhidos, conforme mostra a Figura 4.15: abordagens, testes, critérios de sucesso, diretrizes ou técnicas. Cada um desses grupos contém os elementos que efetivamente devem ser selecionados para que a associação seja concluída.

Tomando como base a escolha do subgrupo critérios de sucesso, podemos ver as opções disponíveis mostradas na Figura 4.16. Os elementos internos da ontologia são diferentes,

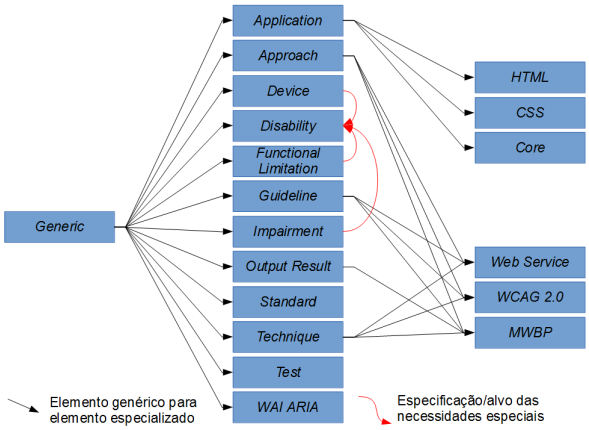


Figura 4.14: Relacionamento entre os elementos da ontologia (Elementos retirados da ontologia)

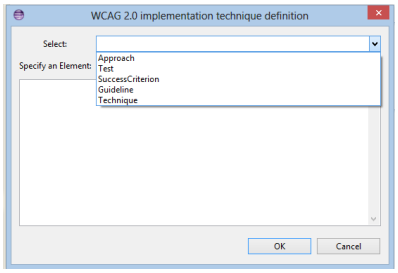


Figura 4.15: Grupos a serem selecionados a partir da ontologia *WCAG 2.0*

dependendo do seu tipo e objetivo.

A Figura 4.17 mostra os elementos da ontologia escolhida como exemplo (*WCAG2 SuccessCriterion 1.2.7*): ela possui uma descrição, uma prioridade, um ID e um nome, além de relacionamentos com outros elementos da ontologia que aqui foram omitidos. Uma vez selecionada o item apropriado da ontologia, um clique no botão OK faz com que a associação seja persistida ao modelo.

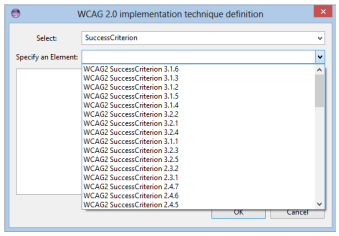


Figura 4.16: Elementos disponíveis para a ontologia *SuccessCriterion*

É possível rastrear os requisitos, modelos UML e técnicas de implementação através da geração de uma matriz de rastreabilidade. Neste trabalho foi utilizado o *Apache ODF Toolkit* (Apache, 2013a) para geração do documento de rastreamento com formato Open-Document Spreadsheet file format (ODS), através da API de alto nível *Simple* (Apache, 2013b). No documento, são geradas três planilhas: Requisitos *versus* Modelos, Requisi-

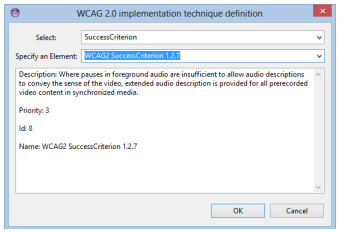


Figura 4.17: Seleção do elemento *WCAG 2.0 SuccessCriterion 1.2.7* da ontologia

tos *versus* Técnicas e Modelos *versus* Técnicas. A planilha Requisitos *versus* Modelos lista uma matriz de todos os Requisitos associados aos respectivos modelos. Requisitos que não estão associados a um modelo também são listados. Essa situação está prevista justamente para identificar problemas na associação que devam ser corrigidos. Para as planilhas Requisitos *versus* Técnicas e Modelos *versus* Técnicas, apenas os objetos que estão efetivamente referenciados na ferramenta são listados.

A Figura 4.18 mostra uma parte da matriz de rastreabilidade gerada automaticamente pela ferramenta *AccTrace*. A matriz deve ser gerada utilizando o *wizard* específico para esse fim, acessando na barra de tarefas as opções *File*, *New* e em seguida *Other...*, selecionando a opção *Traceability Matrix file wizard*. Para a geração da matriz ocorrer sem erros, é necessário informar o arquivo que armazena o modelo da ferramenta, com a extensão .acctrace.

A	B	C	D	E	F	G	H	I	J
	ActivityTravel Agency::Behaviours::New Activity	ActivityTravel Agency::Behaviou rs::Sales	ComponentTravel Agency::Components::Acc ounting	ComponentTrave l Agency::Compo nents::Web Agent::Agency App	ComponentTr avel Agency::Co mponents::A gency Offers	ClassTravel Agency::Co mponents::A gency Offers::Agen cy	ClassTravel Agency::Co mponents::C RM::Custom er	ClassTravel Agency::Co mponents:: Web Agent::Custo mers App::Custom ersApp	ClassTravel Agency::Co mponents::S upplier Purchasing:: Partner
Name: Tempo de Resposta de 2s ID: RNF1	X	X							
Name: Processar Venda ID: RF1			X	X	X	X	X	X	X
Name: Efetuar Pagamento ID: RF3			X	X	X	X	X	X	X
Name: Reservar ID: RF2			X	X	X	X	X	X	X
Name: Os subpassos devem ser salvos par	X	X							

Figura 4.18: Parte da matriz de rastreabilidade gerada pela ferramenta *AccTrace*

Para geração de código, foi utilizado o *plugin UML to Java Generator*, que por sua vez utiliza o ponto de extensão do *plugin Acceleo* (Eclipse, 2013a). O *plugin Acceleo* tem por finalidade transformar os modelos EMF em uma linguagem definida pelo usuário, através de arquivos com a extensão .mtl (*Model to Text Language*). Os arquivos .mtl possuem uma sintaxe própria, que pode ser vista no trecho de código mostrado a seguir.

```
[comment encoding = UTF-8 /]
[module classifierJavaFile('http://www.eclipse.org/uml2/4.0.0/UML')]

[import org::obeonetwork::pim::uml2::gen::java::common::documentation /]
[import org::obeonetwork::pim::uml2::gen::java::common::path /]

[import org::obeonetwork::pim::uml2::gen::java::services::importService /]

[template private classifierJavaFilePath(aClassifier : Classifier)]
```

```
    [if (not aClassifier.getNearestPackage().oclIsUndefined())]
        [aClassifier.genPackagePath()/][aClassifier.name/].java
    [else]
        [aClassifier.name.concat('.java')/]
    [/if]
[/template]
```

Enquanto o *Acceleo* tem por finalidade transformar modelos EMF em uma linguagem pré-definida pelo usuário, o plugin *UML to Java Generator* tem a finalidade específica de transformar modelos UML (persistidos como modelos EMF) em linguagem *Java*. Ele não gera apenas código *Java*, mas o projeto inteiro no *Eclipse*, criando as pastas, arquivos de configuração, informações de importação e exportação, ambiente de execução alvo, entre outros. Por este motivo também, o *plugin UML to Java Generator* é bastante simples. As partes principais do *plugin* são:

- a User Interface (UI), especificando opções de configuração, como modelo UML a ser utilizado, ambiente de execução alvo, entre outras opções;
- as classes de lançamento do *plugin Acceleo*, utilizando as informações da UI;
- os arquivos .mtl que descrevem como os modelos devem ser gerados, no caso, as informações de geração dos arquivos *Java* e do projeto *Eclipse*.

As alterações necessárias para que o *plugin UML to Java Generator* gerasse os comentários personalizados do modelo *AccTrace* foram:

1. alteração da UI, para que fosse possível indicar o arquivo .acctrace;
2. intervenção nas classes de lançamento, para que o arquivo .acctrace fosse corretamente carregado e recuperado quando necessário;
3. alteração dos arquivos .mtl necessários para que o código *Java* gerado incluísse o(s) comentário(s) do modelo *AccTrace*, se houver uma referência para o elemento UML avaliado no momento.

O comentário *AccTrace* é recuperado usando *query invoke* (Obeo Network, 2013), um mecanismo utilizado pelo *plugin Acceleo* que encapsula dentro dos arquivos .mtl chamadas nativas a funções *Java*. O *query invoke* usado foi incluído no arquivo *org.obeonetwork.pim.uml2.gen.java.services.typesServices.mtl* e é descrito a seguir.

```
[query public getComment(anOclAny : OclAny) : String
= invoke('org.obeonetwork.pim.uml2.gen.java.services.AccTraceServices',
'getComment(org.eclipse.emf.ecore.EObject)', Sequence{anOclAny}) /]
```

No momento desejado (por exemplo, logo após a construção da especificação de uma classe ou interface) o *query invoke* é chamado, passando como argumento o objeto UML avaliado no momento. Na classe *Java AccTraceServices* existe um método *getComment* que recebe como argumento um objeto *EObject*, ou seja, qualquer elemento EMF. Neste método, é verificado se o modelo *AccTrace* foi especificado (em caso negativo, nada é retornado). Logo em seguida, é verificado se no modelo *AccTrace* existe uma referência ao elemento UML avaliado (significando que existe uma associação entre um requisito, modelo e técnicas), e em caso positivo, o comentário é construído e retornado, sendo incluído no corpo do código *Java*. Caso o elemento UML não seja encontrado no modelo, uma *String* vazia é retornada.

O comentário *AccTrace* é construído utilizando a seguinte expressão regular *Java*:

```
String regex = "//!ACCTRACE!(/)?([^/\\\\0#]+(/)?)+#([^\\*\\*/])+";
```

A expressão regular foi construída de modo a permitir, através de uma *string* pré-definida, a recuperação do arquivo *Acctrace*, bem como do elemento de associação referenciado.

O comentário personalizado foi projetado de forma a não interfirir em outros comentários da linguagem. Iniciando pelas barras duplas (//), foi definido um identificador que tem a intenção de ser único (!ACCTRACE!), e o próprio Identifier (ID) do recurso RDF possui o caracter "#", que é usado para separador entre o recurso (arquivo a ser aberto) e o elemento a ser resgatado (no caso, a associação criada entre o requisito e o modelo UML). A opção de se utilizar um comentário de linha (ao invés de comentários de bloco, usados por exemplo pela documentação *JavaDoc*) decorreu do fato do comentário descrito ser incluído dentro do corpo da classe e métodos, e caso o programador decida comentar o trecho de código inteiro, esse comentário não atrapalharia essa operação.

Uma vez tendo os comentários personalizados nos código-fonte Java, o *plugin AccTrace* se encarrega de verificar os comentários para traduzí-los em informações úteis ao desenvolvedor. O *plugin* usa o ponto de extensão *Compilation Participant* da biblioteca Eclipse Java development tools (JDT), usado para acompanhar e recuperar informações úteis no processo de compilação do código. No caso deste trabalho, o ambiente entrega a Abstract Syntax Tree (AST) do modelo Java, e a partir dele os comentários são extraídos. Caso o comentário atenda à expressão regular de comentário do *plugin AccTrace*, uma marca (*marker*) é adicionada ao editor, indicando que naquele ponto há um comentário válido. A marca replica o comentário, de modo que ao passsar o mouse sobre a marca, um *popup* surge, conforme mostrado pela Figura 4.19.

Ao clicar na marca, uma visão é notificada para que o conteúdo do comentário seja traduzido. A Figura 4.20 mostra um comentário já traduzido, onde se pode observar informações como requisito, modelo UML e quais técnicas estão referenciadas por aquele comentário.

A Figura 4.21 mostra como recuperar as informações relevantes partindo de um comentário no formato AccTrace.

O comentário passa por um *Regex Match*, sendo decomposto em:

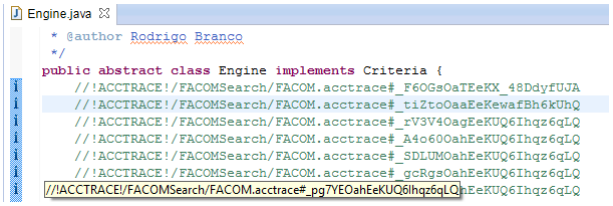


Figura 4.19: Comentário padrão *AccTrace* demonstrado utilizando o evento *mouse hover*

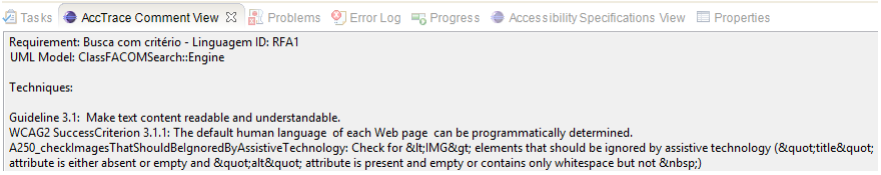


Figura 4.20: Explicitação do comentário selecionado

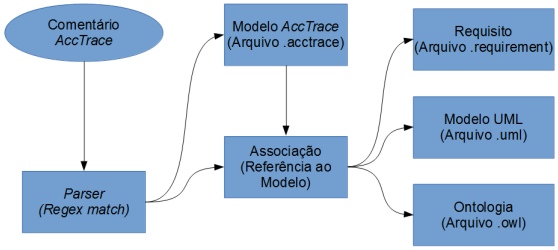


Figura 4.21: Passos para recuperação das informações relevantes através de um comentário padrão AccTrace

- Caminho relativo do arquivo do modelo AccTrace no *workspace* do *Eclipse*;
- Referência (ID) da associação dentro do modelo;

O modelo *AccTrace* é carregado por meio do caminho encontrado e logo em seguida a referência da associação é recuperada do modelo, usando seu ID como parâmetro de busca. A partir da referência é possível recuperar o Requisito, o Modelo UML e as técnicas de implementação de acessibilidade. Como no modelo AccTrace apenas as referência aos objetos é armazenada, qualquer mudança nos objetos envolvidas é vista em tempo real. A atualização da tela da visão pode não ser instantânea, dependendo da complexidade dos objetos envolvidos e da quantidade de recursos que devem ser recuperados.

A recuperação dos comentários personalizados encerra a proposta deste trabalho. Os passos aqui descritos descrevem a construção da ferramenta, seguindo a abordagem da Figura 4.5 e detalhada na Figura 4.6.

4.8 Considerações Finais

Foi possível observar, por meio da pesquisa bibliográfica apresentada no Capítulo 3, que a rastreabilidade dos requisitos, aliada ao processo de desenvolvimento de *software*, é um tema pouco explorado nos trabalhos anteriores. Por este motivo, foram utilizadas as abordagens aqui descritas para o desenvolvimento deste trabalho. A grande maioria dos trabalhos aqui relacionados dizem respeito ao produto pronto, focando em sua interface, navegação, arquitetura e avaliação. Dos trabalhos que tratam de rastreabilidade de requisitos e processos de desenvolvimento, não foram encontrados trabalhos que permitam a recuperação das informações de rastreabilidade do código da solução. E, principalmente, não foram encontrados trabalhos que vinculem aos requisitos/modelos UML suas respectivas técnicas de implementação de acessibilidade.

Este capítulo apresentou as tecnologias, ferramentas e abordagens utilizadas para a construção do *plugin Acctrace*, ferramenta CASE proposta por este trabalho. O próximo capítulo apresenta uma prova de conceito, que utiliza o MTA, os *plugins* e ferramentas aqui elencados para demonstrar a rastreabilidade dos requisitos de acessibilidade, e sua conexão com as técnicas de implementação de acessibilidade.

Capítulo 5

Prova de Conceito

Como prova de conceito, foi criado um projeto simples, de forma a utilizar o MTA, o *plugin AccTrace* e os outros recursos relacionados neste trabalho. Os atores elencados são fictícios, tendo os autores deste trabalho como especialista em acessibilidade. O objetivo não é aplicar a prova de conceito no MTA, por isso nem todos os artefatos presentes no MTA foram construídos. A ênfase considerada nesta prova de conceito está no modelo de rastreabilidade e na ferramenta proposta neste trabalho.

5.1 Definição do projeto

O cliente solicitou a construção de um *software* buscador. O objetivo deste *software* é fazer buscas na *internet* utilizando como chave uma ou mais palavras fornecidas pelo usuário. O buscador será um software *web*, sendo acessado através de navegadores.

5.1.1 Subprocesso 1 - Elicitação dos Requisitos do Sistema

No Subprocesso 1 (Elicitação dos Requisitos do Sistema), observando a Tarefa 1.1 (Identificar os Requisitos de Acessibilidade do Sistema) do MTA (Maia, 2010), o especialista em acessibilidade identificou que:

- não existem características específicas do usuário: todos usuários poderão acessar o sistema, independente de habilidade e deficiências dos mesmos;
- como requisitos de domínio, foi identificado que o *software* deve estar disponível para a linguagem nativa do usuário, e caso ele prefira, a linguagem pode ser alterada. A busca deve ser influenciada pela escolha da linguagem;
- não existem requisitos tecnológicos específicos: o sistema deve ser acessado independente de dispositivo ou tecnologia;

- requisito de performance: o sistema deve retornar o resultado em menos de 2 segundos, para não prejudicar a percepção do usuário;
- requisito de conformidade: o sistema deve seguir a recomendação de acessibilidade WCAG 2.0, sem nível definido.

Nenhum requisito de acessibilidade foi identificado nesta etapa; até agora, o sistema deve seguir as boas práticas genéricas de acessibilidade para englobar todos os conjuntos de usuários/deficiências.

5.1.2 Subprocesso 2 - Análise de Requisitos do Sistema

Partindo para o Subprocesso 2 (Análise de Requisitos do Sistema), na Tarefa 2.1 (Especificar os Requisitos de Acessibilidade do Sistema) (Maia, 2010), o especialista em acessibilidade identificou que:

- características do usuário: não existe um público alvo definido. Não é possível inferir qual o nível de experiência do público alvo;
- requisitos de domínio: como ordem de execução de tarefas, o usuário deve informar a chave de pesquisa, recebendo o resultado no passo seguinte;
- requisitos de tecnologia: dependendo da deficiência/impedimento momentâneo do usuário, as tecnologias podem variar desde um leitor de tela até o navegador textual. Isso deve ser considerado na utilização do *software*;
- alternativas pré-existentes: serviços como Google, Bing ou Yahoo fornecem buscadores com os mesmo propósitos;
- qualidade das alternativas: o buscador do Google promove a busca acessível levando em consideração alguns aspectos das páginas analisadas (Google, 2013). Não foi identificado se o buscador Bing ou Yahoo possui algum mecanismo que avalie a acessibilidade das páginas avaliadas, apesar das duas empresas possuírem áreas que tratam de questões de acessibilidade[1].

O especialista em acessibilidade constatou que 2 requisitos não funcionais de acessibilidade estão presentes: o mecanismo de busca será alterado toda vez que a linguagem padrão for alterada, e o mecanismo de busca será alterado quando o usuário informar suas necessidades específicas, por exemplo, sua deficiência. As duas abordagens podem ser desativadas caso o usuário prefira.

Nada mais pode ser dito sobre acessibilidade neste momento. Portanto, o especialista em acessibilidade acrescentou como requisito não funcional: os elementos do sistema devem ser: perceptíveis, operáveis, compreensíveis e robustos, seguindo os princípios do

[1] http://yaccessibilityblog.com/ e http://www.microsoft.com/enable/

WCAG 2.0. A avaliação dos requisitos de acessibilidade é feita na Tarefa 2.2 (Avaliar os Requisitos de Acessibilidade do Sistema) (Maia, 2010) e registrada no documento de requisitos de acessibilidade do sistema revisado.

5.2 Modelagem do Sistema

Os artefatos para que os desenvolvedores implementem o sistema começam a surgir a partir desta etapa.

5.2.1 Subprocesso 3 - Projeto Arquitetural do Sistema

No Subprocesso 3, Projeto Arquitetural do Sistema, as Tarefas 3.1 (Alocar Requisitos de Acessibilidade aos Elementos do Sistema) e 3.2 (Avaliar o Projeto Arquitetural do Sistema com Relação aos Requisitos de Acessibilidade) (Maia, 2010) tratam da decomposição do sistema em componentes menores, assim como a especificação da responsabilidade de cada componente. Para os requisitos de acessibilidade descritos anteriormente, é definido que, para cada um dos requisitos não funcionais (alteração do mecanismo de busca), um componente individual será gerado, podendo ter um elemento de renderização genérico, ou um ou mais componentes de renderização para cada uma das alterações do mecanismo de busca, ou até da combinação delas.

O requisito não funcional “os elementos do sistema devem ser: perceptíveis, operáveis, compreensíveis e robustos” deve permear todos os componentes do projeto, inclusive os já descritos anteriormente.

5.2.2 Subprocesso 4 - Análise de Requisitos do Software

No Subprocesso 4, Análise de Requisitos do Software, os requisitos de acessibilidade do *software* (interface e código) são estabelecidos.

Nas Taferas (Estabelecer os Requisitos de Acessibilidade do Software) e (Avaliar os Requisitos de Acessibilidade do Software) os requisitos de sistema são especificados, avaliados e documentados em requisitos de *software*. Todos os requisitos, tanto os funcionais (RF) como os não-funcionais (RNF), inclusive os de acessibilidade (RNFA), podem ser vistos abaixo:

- RF1: o sistema deve permitir que, dado uma chave de busca (uma ou mais palavras), o resultado da busca seja retornado;
- RNF1: o sistema deve retornar o resultado da busca em menos de 2 segundos, para não prejudicar a percepção do usuário;
- RNF2: o sistema deve permitir que o usuário altere a linguagem padrão da ferramenta;

- RNF3: o sistema deve permitir que o usuário altere o constraste da página;
- RNF4: o sistema deve permitir que o usuário altere o tamanho da fonte da página;
- RNF5: o sistema deve permitir que o usuário informe qual é o tipo de limitação/deficiência possui;
- RNF6: o sistema deve fornecer atalhos via teclado, para alterar o idioma e a informação sobre a limitação/deficiência, sempre voltando o foco para a caixa de texto principal;
- RNFA1: o sistema deve seguir a recomendação de acessibilidade WCAG 2.0, nível AAA;
- RNFA2: o sistema deve fornecer componentes perceptíveis, operáveis, compreensíveis e robustos;
- RNFA3: o sistema deve permitir que o tempo para ler e utilizar o conteúdo seja suficiente;
- RNFA4: o sistema deve entregar um conteúdo que não cause convulsão;
- RNFA5: o sistema deve permitir que o usuário navegue facilmente e encontre o conteúdo desejado.
- RNFA6: o sistema deve utilizar a linguagem especificada como critério de seleção no mecanismo de busca;
- RNFA7: o sistema deve utilizar o tipo de limitação/deficiência como critério de seleção no mecanismo de busca;
- RNFA8: o sistema deve permitir que todas as funcionalidades sejam acessíveis via teclado;
- RNFA9: o sistema deve permitir que, ao ser especificado o tipo de limitação/deficiência, a utilização de métrica de acessibilidade sejam utilizadas para classificar as páginas.

No *plugin Requirement Designer*, os requisitos são agrupados em 2 categorias principais: Requisitos Funcionais e Requisitos Não Funcionais. A categoria de Requisitos Não Funcionais possui um Subcategoria (Requisitos Não Funcionais de Acessibilidade). A Figura 5.1 mostra o cadastro dos requisitos na ferramenta.

O diagrama de casos de uso pode ser visto na Figura 5.2, assim como o diagrama de classes de análise pode ser visto na Figura 5.3. Como a implementação final deste projeto não é o foco deste trabalho, as especificações dos casos de uso, assim como outros diagramas UML que não são transformados no código final não serão construídos. O diagrama de casos de uso aqui apresentado também não é usado pelo gerador de código, mas pela importância do artefato entendeu-se que era necessário a construção do mesmo.

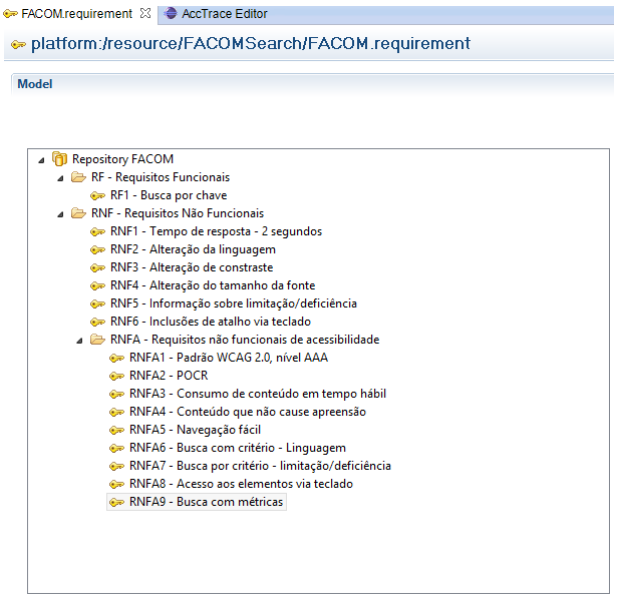


Figura 5.1: Cadastro dos requisitos no *plugin Requirement Designer*

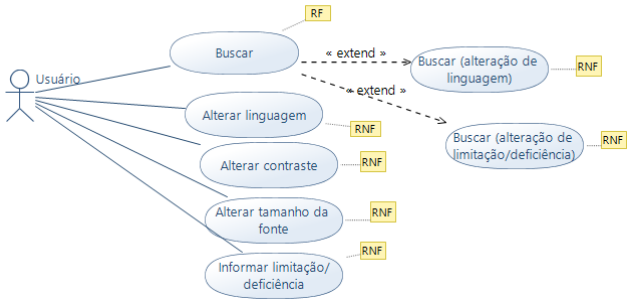


Figura 5.2: Diagrama de casos de uso do buscador

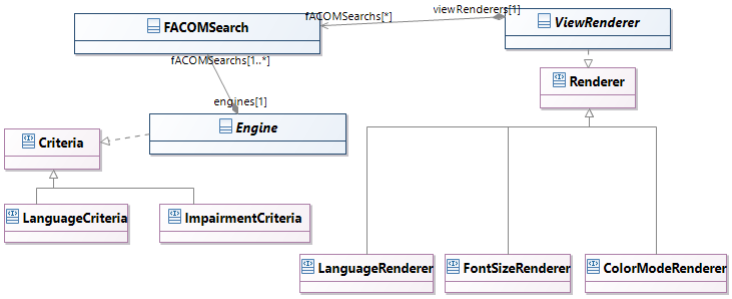


Figura 5.3: Diagrama de classes de análise do buscador

Vale notar que o MTA não especifica exatamente quando os diagramas apresentados devem ser construídos. Por este motivo, o momento oportuno para a construção dos mesmos foi entre os Subprocessos 4 e 5, pela gama de informações já levantada.

De posse dos diagramas UML, já é possível associá-los aos requisitos, utilizando o

plugin Requirement Designer. Com os requisitos e modelos UML já associados, o arquivo .acctrace pode ser criado, recebendo como entrada o arquivo .requirement. A Figura 5.4 mostra os elementos UML, e para o modelo selecionado, os requisitos que a ele estão associados.

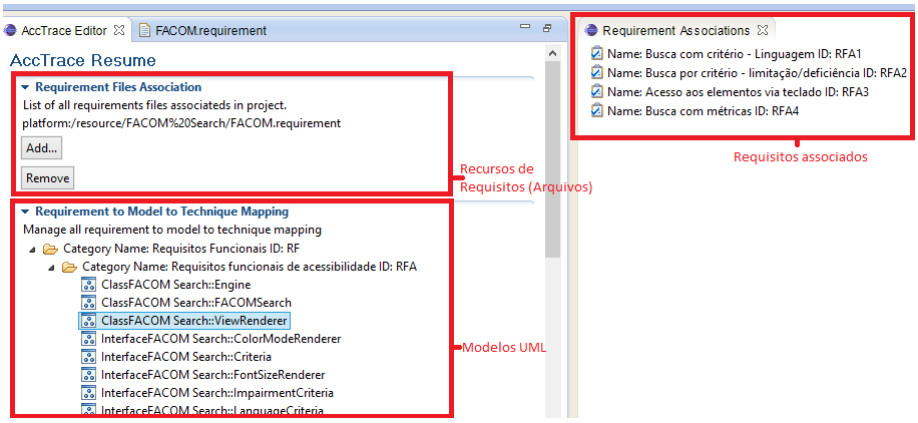


Figura 5.4: Modelos UML e requisitos associados

5.2.3 Subprocesso 5 - Projeto de Software

O Subprocesso 5 (Projeto de *Software*) visa fornecer um projeto onde os requisitos do software possam ser implementados e verificados. Os elementos de interface são definidos nas tarefas seguintes (Tarefa Projetar Interfaces Externas Acessíveis e Tarefa Realizar Projeto Navegacional Acessível), e Maia (2010) estabelece relacionamentos com o WCAG 2.0 nestas tarefas. Por este motivo, mostrou-se adequado especificar as técnicas de acessibilidade neste momento. A Figura 5.5 mostra o diagrama de classes de projeto do busca (expansão do diagrama de classes de análise).

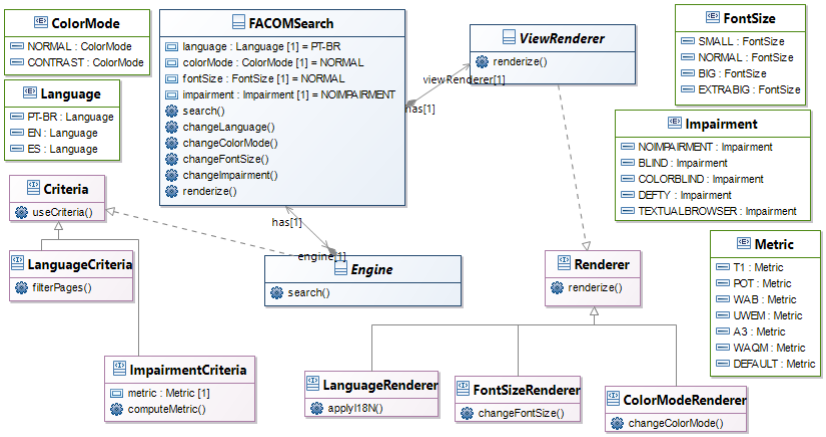


Figura 5.5: Diagrama de classes de projeto do buscador

Na Subtarefa Estabelecer a disposição dos elementos na interface, por exemplo, são citados três critérios de sucesso:

- **critério de Sucesso 1.3.1 - Informações e Relacionamento:** A informação, estrutura e relacionamento transmitido por meio de apresentação devem ser determinados programaticamente;
- **critério de Sucesso 1.3.2 - Sequência Significativa:** A sequência correta de leitura dos elementos da interface deve ser determinada programaticamente;
- **critério de Sucesso 1.3.3 - Características Sensoriais:** As instruções fornecidas para o entendimento e operação do conteúdo não devem confiar somente em características sensoriais dos componentes, tais como forma, tamanho, localização visual, orientação ou som.

A Subtarefa "Definir as cores dos elementos" sugere um critério de sucesso:

- **critério de Sucesso 1.4.1 - Uso de Cores:** As cores não devem ser usadas como único meio para transmitir informação, indicar uma ação, mostrar uma resposta ou distinguir um elemento visual.

A Subtarefa "Definir a ordem de navegação dos objetos" sugere um critério de sucesso:

- **critério de Sucesso 2.4.3 - Ordem do Foco:** Se a página Web pode ser navegada sequencialmente e a sequência de navegação afeta o entendimento ou operação, então os componentes focalizáveis devem receber o foco considerando a ordem dos elementos, preservando o entendimento e a operabilidade dos mesmos.

Maia (2010) não especificou nenhum critério de sucesso para a Subtarefa "Definir atalhos de navegação", mas é possível sugerir genericamente um que sirva como ponto de partida:

- **critério de Sucesso 2.1.1 - Teclado:** Toda a funcionalidade do conteúdo é operável através de uma interface de teclado sem requerer temporizações específicas para digitação individual, exceto quando a função subsequentes requer entrada que depende do caminho do movimento do usuário e não apenas dos pontos finais.

Os critérios de sucesso aqui apresentados são sugestões e podem ou não ser utilizados dependendo da situação. Neste ponto, podemos especificar as técnicas de acessibilidade no *plugin AccTrace*. É importante notar que não são tratados apenas critérios de sucesso que estão presentes no *plugin*; todas as abordagens, diretrizes, técnicas, entre outros (definidos nos arquivos de ontologia) estão disponíveis para a associação, dependendo apenas das características específicas do projeto e da maturidade do especialista em acessibilidade.

A Figura 5.6 mostra, como exemplo de uma das associações, a classe *Engine*, presente no diagrama de classes, associada ao requisito não funcional RNFA1 (Padrão WCAG 2.0, nível AAA), tendo as diretrizes 1.1, 2.1, 3.1 e 4.1 associadas como técnicas de acessibilidade.

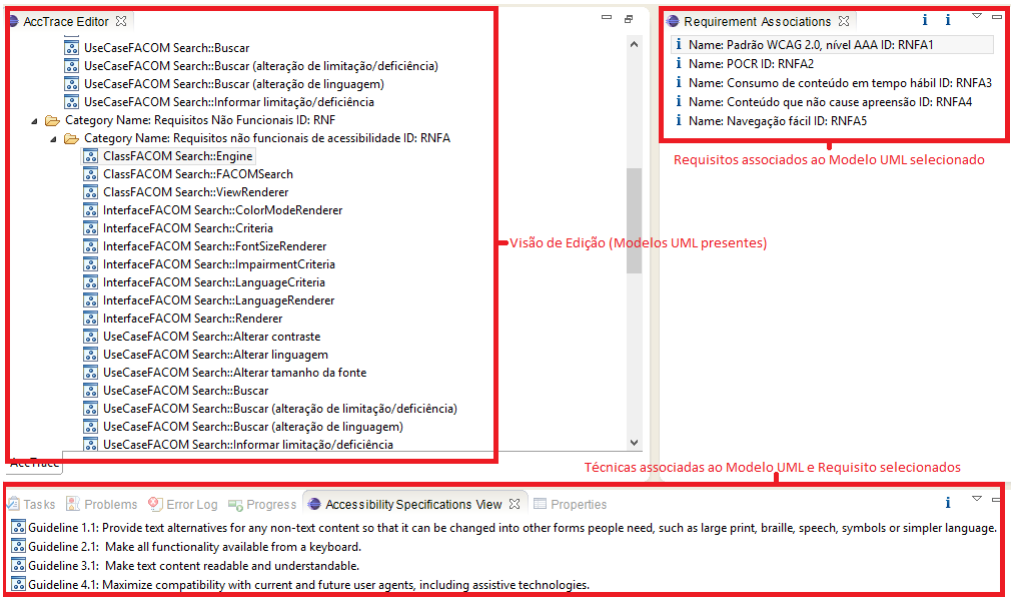


Figura 5.6: Associação das técnicas de acessibilidade para um modelo e requisito selecionado.

Neste ponto é possível gerar a matriz de rastreabilidade. As Figuras 5.7 e 5.8 mostram uma parte dos dados gerados, para a associação de requisitos e técnicas, bem como modelos e técnicas.

	A	B	C	D	E	F	G	H	I	J	K	L	
		WCAG2 Suc	Guideline 2.2	Guideline 2.4	WCAG2 Suc	WCAG2 Suc	WCAG2 Suc	Guideline 1.4	WCAG2 Suc	Guideline 3.1	WCAG2 Suc	Guideline 2.1	WCA
Name: Busca por critério - limitação/deficiência ID: RFA2		X											
Name: Consumo de conteúdo em tempo hábil ID: RNFA3			X										
Name: Informação sobre limitação/deficiência ID: RF6				X	X	X	X						
Name: Alteração de constraste ID: RF3								X	X				
Name: Busca com critério - Linguagem ID: RFA1										X	X		
Name: Inclusões de atalho via teclado ID: RF8												X	X
Name: Acesso aos elementos via teclado ID: RFA3												X	X
Name: Conteúdo que não cause apreensão ID: RNFA4													
Name: Navegação fácil ID: RNFA5				X	X	X	X						
Name: POCR ID: RNFA2										X		X	
Name: Alteração do tamanho da fonte ID: RF4													
Name: Alteração da linguagem ID: RF2													
Name: Padrão WCAG 2.0, nível AAA ID: RNFA1										X		X	

Figura 5.7: Parte da matriz de rastreabilidade de requisitos e técnicas

5.2.4 Subprocesso 6 - Construção do Software

A proposta da construção do software é produzir unidades de *software* executáveis que apropriadamente refletem o projeto do software. A Tarefa 6.1 (Especificar Técnicas para Implementação da Acessibilidade da Interface e do Código), segundo Maia (2010), sugere que técnicas específicas para atender aos requisitos de acessibilidade do sistema deverão ser pesquisadas tais como o documento de referência do WCAG que apresenta técnicas suficientes para a promoção de acessibilidade. Para a prova de conceito proposta, este passo foi feito no final do Subprocesso 5, utilizando o *plugin AccTrace* para tal.

A tarefa 6.2 (Codificar cada Unidade de Software de Acordo com Técnicas de Acessibilidade) refere-se à construção propriamente dita da solução proposta (Maia, 2010). O

	A	B	C	D	E	F	G	H	I	J	K	
1		Guideline 2.1	WCAG2 Suc	WCAG2 Suc	WCAG2 Suc	WCAG2 Suc	WCAG2 Suc	Guideline 1.3	Guideline 1.4	WCAG2 Suc	Fonts	Guid
2	InterfaceFACOM Search::Renderer	X	X	X	X	X	X	X	X	X	X	X
3	InterfaceFACOM Search::LanguageRenderer	X	X		X	X	X	X				X
4	UseCaseFACOM Search::Alterar linguagem	X	X	X	X	X	X	X				X
5	UseCaseFACOM Search::Alterar tamanho da fonte	X	X	X							X	X
6	InterfaceFACOM Search::LanguageCriteria	X										X
7	InterfaceFACOM Search::FontSizeRenderer	X	X	X							X	X
8	InterfaceFACOM Search::Criteria	X										X
9	UseCaseFACOM Search::Buscar (alteração de limitação/deficiência)	X										X
10	ClassFACOM Search::Engine	X										X
11	InterfaceFACOM Search::ColorModeRenderer	X	X	X					X	X		X
12	ClassFACOM Search::FACOMSearch	X	X		X	X	X	X	X	X		X
13	UseCaseFACOM Search::Alterar contraste	X	X	X					X	X		X
14	UseCaseFACOM Search::Buscar (alteração de linguagem)	X										X
15	ClassFACOM Search::ViewRenderer	X	X	X	X	X	X	X	X	X	X	X
16	UseCaseFACOM Search::Informar limitação/deficiência	X	X	X								X
17	InterfaceFACOM Search::ImpairmentCriteria	X										X
18	UseCaseFACOM Search::Buscar	X										X

Figura 5.8: Parte da matriz de rastreabilidade de modelos e técnicas

plugin modificado *UML to Java Generator* se encarregará de construir os códigos *stub* com os comentários personalizados *AccTrace*, recebendo como entrada os modelos UML e também o arquivo de referências gerados pelo *plugin AccTrace*.

A Figura 5.9 ilustra uma parte do código gerado, mostrando também alguma das classes geradas na visão *Project Explorer*, dando ênfase na classe *ViewRenderer*, observando os comentários *AccTrace* personalizados bem como a descrição de um deles na visão *AccTrace Comment View*.

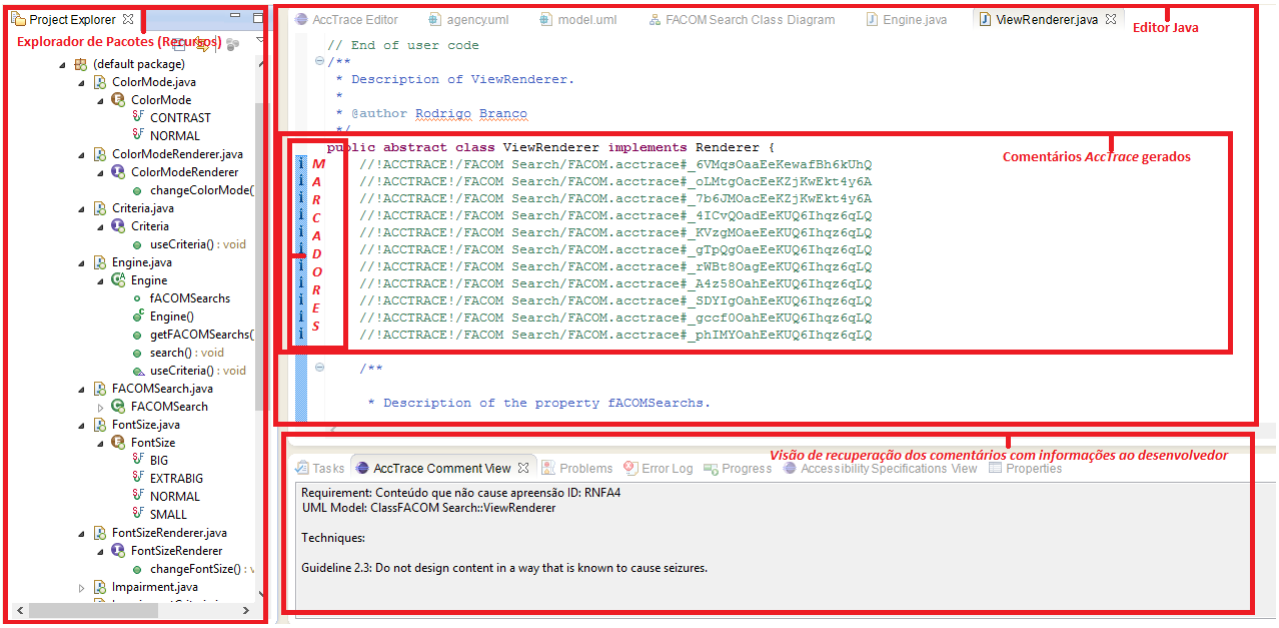


Figura 5.9: Visualização do código gerado, com os comentários *AccTrace* personalizados

Os desenvolvedores do *software* podem partir destes códigos *stub* e continuar a desenvolvimento da ferramenta, sendo beneficiados pelos comentários *AccTrace* espalhados pelas classes. A prova de conceito termina neste ponto, pois prosseguindo com o processo de desenvolvimento MTA, as próximas tarefas são a Tarefa 6.3 (Planejar Teste de Acessibilidade para Cada Unidade de Software) e a Tarefa 6.4 (Executar Teste de Acessibilidade de Cada Unidade de Software) (Maia, 2010), que estão fora do escopo desta dissertação. É importante notar que a ontologia utilizada inclui elementos de testes que podem ser

associados como técnicas de acessibilidade W3C (2013e) que podem ser utilizadas em trabalhos futuros.

5.3 Limitações da Prova de Conceito

A prova de conceito aqui apresentada visa demostrar como seria a utilização das técnicas e ferramentas em um ambiente real. Contudo, não foi possível verificar se os passos aqui apresentados seriam realmente utilizados em projeto real. Apesar de estender a ISO/IEC 12207, a utilização efetiva do MTA não foi encontrada na literatura, para servir como base para este trabalho.

A operação dos *plugins* construídos/alterados ao longo deste trabalho se mostrou difícil em casos pontuais. Algumas dificuldades só foram encontradas na execução da prova de conceito, sugerindo que algumas partes específicas da ferramenta devem ser adequadas, para que a usabilidade das ferramentas seja melhorada.

O volume de dados dos artefatos gerados foi relativamente pequeno. A falta de profissionais que efetivamente trabalhem com Engenharia de Requisitos, Modelagem e principalmente, a falta de um Especialista em Acessibilidade, profissionais estes que poderiam gerar dados mais próximos de um projeto real, impacta significativamente na análise do volume de dados *versus* facilidade de utilização. As ferramentas geralmente se tornam mais complexas e precisam de filtros complexos para entregar os dados corretos no momento certo, escondendo os demais, quando o volume de dados é muito grande.

A falta de um Especialista em Acessibilidade impactou negativamente em perceber se as informações apresentadas pela ferramenta foram suficientes ou desnecessárias. Não foi possível dizer se os mapeamentos encontrados na ontologia utilizada foram suficientes para fornecer informações úteis aos projetistas/desenvolvedores. Este não é um problema tão sério, já que a ontologia pode ser modificada/aprimorada utilizando a ferramenta *Protegé*, devendo o *plugin AccTrace* ser alterado para que comporte a nova realidade do domínio.

Devido a complexidade dos relacionamentos da ontologia utilizada, apenas as diretrizes do modelo de referência WCAG 2.0 foi utilizada, por já estarem previamente mapeadas. O *plugin AccTrace* foi inteiramente construído utilizando estes relacionamentos como orientação. Por este motivo, a inclusão de novos modelos (como por exemplo, o eMag 3.0) não é uma tarefa simples. Primeiro, a ontologia deve ser construída e depois o *plugin AccTrace* deve ser adaptado para refletir a nova realidade do modelo de referência. A consequência deste fato é que a atualização dos modelos de referência torna-se uma tarefa complexa, inclusive para a atualização do modelo WCAG 2.0, se houver diferenças na estrutura já utilizada na ontologia usada neste trabalho. Caso a estrutura entre os elementos da ontologia não seja alterada, não é necessário fazer alterações no *plugin AccTrace*.

Uma limitação, que deve ser tratada em trabalhos futuros, é que o sistema atualmente não trata o caso de remoção de requisitos/modelos UML dos artefatos iniciais. Mudanças nos requisitos e modelos UML são automaticamente atualizados (visto que apenas as referências a esses objetos são armazenadas). Contudo, as remoções dos elementos nos

artefatos causam um comportamento inesperado no sistema, obrigando o modelo *AccTrace* a ser atualizado manualmente (com comandos explícitos de remoção das associações no *plugin*).

5.4 Conclusões da Prova de Conceito

Apesar das limitações encontradas na execução da prova de conceito, os objetivos propostos foram atingidos. A seguir, apresenta-se como os objetivos propostos para abordagem desenvolvida neste trabalho foram atingidos:

- **desenvolvimento de uma ferramenta que seja orientada ao desenvolvedor (a apresentação dos resultados nas ferramentas tradicionais são adequadas para avaliação e auditoria de *sites*, e não para desenvolvedores):** as ferramentas aqui desenvolvidas foram construídas pensando em engenheiros de requisitos, analistas, arquitetos e desenvolvedores;
- **desenvolvimento de uma ferramenta que seja integrada ao ambiente de desenvolvimento do desenvolvedor:** não apenas do desenvolvedor, mas todos os profissionais anteriormente elencados utilizam o ambiente de desenvolvimento *Eclipse*, permitindo a total integração das ferramentas. A única ressalva é quanto a visualização da matriz de rastreabilidade, que deve ser visualizada com ferramentas que aceitem arquivos ODS, notadamente o *LibreOffice* The Document Foundation (2013);
- **desenvolvimento de uma ferramenta que apresente informações objetivas, e no momento em que o desenvolvedor desejar visualizar:** o especialista em acessibilidade define o que será apresentado ao desenvolvedor, e o desenvolvedor escolhe quando verá as informações, ativando a *view AccTrace Comment View*, clicando nos marcadores para que a informação possa aparecer;
- **desenvolvimento de uma ferramenta que tenha relação direta entre os requisitos e casos de uso com a etapa de codificação:** os *plugins* utilizados são interoperáveis, permitindo a definição e relacionamento dos requisitos e modelos UML, recuperados na etapa de codificação;
- **desenvolvimento de uma ferramenta que permita que seja feita o rastreamento dos requisitos de acessibilidade, desde a sua concepção até as fases de codificação:** o rastreamento pode ser feito no *plugin AccTrace*, bem como utilizando a matriz de rastreabilidade;
- **desenvolvimento de uma ferramenta que permita que o desenvolvedor consiga verificar, em nível de código, a associação dos requisitos e modelos:** o desenvolvedor consegue, a partir de um comentário *AccTrace* personalizado, recuperar informações sobre o requisito e modelo UML associado.

Portanto, foi possível rastrear os requisitos, desde a sua concepção e definição, passando pelos modelos e artefatos UML e finalmente, sendo recuperados já na fase de codificação. A entrega de informações que possam ser úteis ao desenvolvedor é efetuada. O especialista em acessibilidade informa qual técnica de acessibilidade determinado modelo/requisito deve usar como guia (seja abordagens, diretrizes, critérios de sucesso, técnicas, etc), estando estas informações disponíveis para consulta pelo desenvolvedor. O desenvolvedor terá, então, um ponto de partida para codificar o produto da forma correta.

5.5 Considerações Finais

Este capítulo apresentou a execução da prova de conceito, demonstrando a utilização das ferramentas construídas e alteradas ao longo deste trabalho, aplicando a um projeto fictício, que utilizou o modelo de processo de desenvolvimento MTA, gerando os artefatos necessários para que a rastreabilidade dos requisitos seja verificada e os códigos *stub* sejam gerados.

O próximo capítulo apresenta as contribuições e trabalhos futuros.

Capítulo 6

Conclusões

Este capítulo apresenta as conclusões gerais desta dissertação. Ele apresenta o resgate de desenvolvimento deste trabalho, sua principais contribuições, limitações, dificuldades e diretrizes para a execução de trabalhos futuros.

6.1 Considerações Iniciais

Este trabalho teve por objetivo entender como acontece o processo de propagação dos requisitos de acessibilidade, desde a fase de Engenharia de Requisitos até a fase de codificação, e entender como os desenvolvedores utilizam esta informação para construir um produto genuinamente acessível.

Ao logo do desenvolvimento deste trabalho, a pesquisa revelou que, primeiramente, poucos processos de desenvolvimento de *software* adotam medidas para incluir tópicos de acessibilidade em sua execução. Foi possível notar também que a rastreabilidade dos requisitos não acontece em nível de código.

Por este motivo, o MTA (Maia, 2010) foi utilizado como diretriz principal para o estudo do processo de rastreabilidade dos requisitos de acessibilidade. O estudo revelou que várias abordagens poderiam ser executadas; a adoção de uma delas ocorreu principalmente devido a fatores como complexidade e ferramentas tecnológicas de apoio disponíveis e utilizadas.

A rastreabilidade de requisitos foi obtida através de matrizes de rastreabilidade (para requisitos e modelos UML) e comentários personalizados (para o código fonte).

Como ponto adicional, foi inserido às associações entre requisitos e modelos UML técnicas de implementação de acessibilidade, para apoiar o desenvolvedor, já que nem todos possuem as habilidades necessárias para construir um produto acessível (Kavcic, 2005). Desta forma, os desenvolvedores podem, no momento da codificação, recuperar informações sobre os requisitos, modelos UML e técnicas de implementação para a efetiva implementação de acessibilidade em seu produto.

A pesquisa de Trewin *et al.* (2010) mostrou que os desenvolvedores estão insatisfeitos com as informações entregues pelas ferramentas de apoio à acessibilidade (geralmente ferramentas de avaliação). Por isso, foi utilizada a ontologia do Projeto Aegis (2013), que mapeia o documento de referência WCAG 2.0, e este fornece valiosas contribuições para a construção de produtos acessíveis, especificando abordagens, testes, critérios de sucesso, técnicas de implementação, entre outros, entregando assim informações úteis e que realmente ajudarão os desenvolvedores na tarefa de construção de seu produto.

Por fim, a pesquisa de Trewin *et al.* (2010) mostrou também que os desenvolvedores estão insatisfeitos com a utilização de ferramentas externas a seu ambiente de desenvolvimento para avaliação dos produtos. Por isso, a ferramenta construída neste trabalho foi um *plugin* para o *Eclipse*, permitindo manter em um mesmo ecossistema de tecnologias as ferramentas de Engenharia de Requisitos, Modelagem UML e codificação, permitindo a integração destas com o *plugin* desenvolvido.

6.2 Contribuições do Trabalho

O tema *Acessibilidade no Processo de desenvolvimento de software* ainda é pouco explorado, com poucos trabalhos publicados a respeito (Maia, 2010; Moulin e Sbodio, 2010). Em contrapartida, podem ser encontrados diversos estudos que dizem respeito ao rastreamento de requisitos (Ali *et al.*, 2011; Gotel e Finkelstein, 1994; Soonsongtanee e Limpiyakorn, 2010; Mader e Egyed, 2012), bem como encontrar estudos que utilizam ontologias para o mapeamento do domínio no processo de desenvolvimento ou na rastreabilidade dos requisitos (Assawamekin *et al.*, 2009; Martins e Machado, 2012; Noll e Ribeiro, 2007; guo *et al.*, 2009b). Contudo, não foi encontrado na literatura estudos específicos sobre a rastreabilidade de requisitos de acessibilidade durante o processo de desenvolvimento de *software*.

Além disso, este estudo mostrou ser possível especificar, antes das fases de codificação e vinculadas aos modelos e requisitos de acessibilidade, as técnicas de implementação que deverão ser visualizadas pelos programadores. A utilização de uma ontologia pré-definida do projeto Aegis (2013) ajudou a alcançar tal objetivo, estendendo as técnicas de implementação anteriormente ditas para abordagens, diretrizes, critérios de sucesso, entre outros.

Este estudo não teve por objetivo entregar uma ferramenta CASE que atenda às exigências aqui descritas; o objetivo principal foi discutir as alternativas para promover a rastreabilidade (utilizando técnicas já descritas na literatura e utilizadas em outros trabalhos), e a partir das várias opções disponíveis, escolher a mais adequada, levando em consideração as ferramentas disponíveis, contexto abordado e limitações inerentes.

Assim como no trabalho de Maia (2010), este trabalho pressupõe que um especialista em acessibilidade participe do processo de desenvolvimento, principalmente nas Subtarefas de acessibilidade propostas pelo MTA. A proposta deste trabalho pode se estender para outras áreas (por exemplo, usabilidade, segurança, etc), desde que haja um especialista e que o domínio esteja mapeado na forma de uma ontologia, ou outro formato que possa

ser interpretado por uma ferramenta CASE. Obviamente, a ferramenta aqui apresentada, da maneira como está, não poderia ser usada para tal propósito, mas pode ser modificada para que se adeque à essa situação, pois é possível perceber que a grande complexidade não é a ferramenta em si, mas sim possuir o especialista e o domínio mapeado.

As alternativas disponíveis promovem a rastreabilidade entre requisitos *versus* modelos UML. Contudo, não foi encontrado como é possível utilizar as alternativas para recuperar informações de rastreabilidade dos requisitos partindo do código fonte do programa. Também não foi possível identificar se essas alternativas permitem a associação de técnicas de implementação aos requisitos e modelos UML.

É possível apontar, portanto, cinco grandes contribuições fornecidas por este trabalho:

1. estender e dar suporte ao MTA, proposto por Maia (2010);
2. abordar o rastreamento dos requisitos de acessibilidade em um processo de desenvolvimento que inclui tarefas de acessibilidade (MTA);
3. permitir a especificação de técnicas de implementação de acessibilidade, mapeadas em uma ontologia pré-definida;
4. recuperar as informações de rastreabilidade por meio comentários personalizado no código fonte;
5. fornecer uma ferramenta como prova de conceito.

6.3 Limitações

Baseado nos estudos realizados e nos resultados da execução da prova de conceito, foi possível observar a necessidade de efetuar estudos empíricos, com o intuito de observar a validação da abordagem fornecida por este trabalho.

A abordagem utilizada apresentou a limitação de não permitir efetuar Engenharia Reversa em projetos que não tenham sido preparados para tal fim. Isto significa que não é possível recuperar, a partir de códigos fontes arbitrários, as informações sobre a rastreabilidade dos requisitos, modelos UML e técnicas de implementação, já que em nível de código, a informação necessária para a recuperação dos dados é justamente o comentário *AccTrace* personalizado, que não estará presente em tais projetos.

O MTA é um processo de desenvolvimento de *software* acadêmico, e por não ter uma grande aderência na indústria, implica em uma limitação direta para este trabalho. Contudo, a abordagem apresentada, apesar de usar o MTA como ponto de partida, não está fortemente vinculada a tal processo de desenvolvimento, dando indícios que outros processos podem ser utilizados. Ainda assim, esta abordagem ainda necessita do acompanhamento de um Especialista em Acessibilidade para que o lançamento das informações seja feito no momento correto.

6.4 Dificuldades encontradas

Este trabalho apresentou vários desafios. O primeiro deles foi descobrir como os requisitos de acessibilidade poderiam ser propagados desde seu levantamento até a fase de codificação, consequentemente sua vinculação com os modelos UML. Escolher a maneira como as informações são persistidas (neste caso, XMI/XML) também influenciou no planejamento e execução da abordagem aqui descrita.

A vinculação das técnicas de implementação de acessibilidade aos requisitos/modelos UML é uma inovação deste trabalho. Portanto, optou-se por utilizar XMI/XML, usando modelos EMF/*ECore* para gerenciamento destas informações.

A escolha das ferramentas tecnológicas foi de fundamental importância, e por isso foi um grande desafio descobrir quais eram as ferramentas mais apropriadas. Como em cada fase do processo de desenvolvimento ferramentas diferentes são usadas, era importante que as ferramentas fosses intercompatíveis.

Por fim, foi preciso encontrar um "repositório" com informações de acessibilidade úteis ao desenvolvedor, caso contrário este deveria ser construído. O projeto *Aegis* (Aegis, 2013) foi de extrema importância, fornecendo uma ontologia de acessibilidade com as informações sobre o domínio acessibilidade e o documento de referência WCAG 2.0.

6.5 Participação em evento

Este trabalho teve sua importância reconhecida por meio da seleção para participação, como trabalho de pós-graduação em andamento, e apresentação de poster no *PASQI Workshop*, realizado em Costa Rica e patrocinado por *Michigan Technological University, Pontificia Universidad Católica de Chile, Miami University, Universidad de Buenos Aires, Universidad de Chile e Universidad de Costa Rica* (Branco, 2013).

6.6 Trabalhos Futuros

É possível identificar várias atividades para trabalhos futuros:

- efetuar um estudo de caso com um projeto real, que utilize o MTA e use os *plugins* aqui elencados, inclusive os *plugins* construídos e customizados para promover a rastreabilidade e geração de código;
- efetuar um estudo de caso com usuários reais, utilizando o *software* construído no item anterior e avaliar a acessibilidade do mesmo;
- estudar a utilização de métodos dinâmicos (a serem definidos) de rastreabilidades dos requisitos;

- efetuar o mapeamento da ontologia para outros documentos de referência em acessibilidade, por exemplo, eMag 3.0;
- aumentar a usabilidade dos *plugins* construídos/modificados, melhorando as mensagens apresentadas, aproveitando o relacionamento da ontologia do projeto Aegis (2013);
- estender o escopo deste trabalho, incluíndo tarefas de testes e integração do *software* e do sistema (Subtarefas 7, 8, 9 e 10 do MTA (Maia, 2010));
- tratar o caso de remoção de requisitos/modelos UML já relacionados no modelo *AccTrace*;
- estender a matriz de rastreabilidade dos requisitos aqui construída, para incluir os casos de testes descritos no item anterior;
- utilizar outro domínio de interesse como base para os estudos futuros, como por exemplo usabilidade de *software*;
- integrar a ferramenta com o *framework Homero* (Oliveira, 2013), que fornece apoio à criação de *interfaces web* acessíveis.

O código fonte do trabalho pode ser encontrado acessando o seguinte endereço: `https://github.com/rodrigogbranco/acctrace.git`. No mesmo repositório é possível encontrar, dentro da pasta *umlToJavaPlugins*, o código fonte do *plugin UML to Java Generator* com as alterações efetuadas neste trabalho.

6.7 Considerações Finais

Existem várias perspectivas de melhoria. Contudo, este trabalho contribuiu com um passo importante na tentativa de promover a rastreabilidade dos requisitos de acessibilidade e também na tentativa de informar ao desenvolvedor, de forma objetiva, quais técnicas devem ser levadas em consideração no desenvolvimento de suas aplicações *web*.

Bibliografia

Accessible Consortium (2013a). Accessible - Applications Design and Development. http://www.accessible-eu.org/index.php/consortium.html. Acessado em Setembro de 2013.

Accessible Consortium (2013b). Accessible - Applications Design and Development - Ontology. http://www.accessible-eu.org/index.php/ontology.html. Acessado em Setembro de 2013.

Aegis (2013). AEGIS Ontology. http://www.aegis-project.eu/index.php?option=com_content&view=article&id=107&Itemid=65. Acessado em Setembro de 2013.

Akhter, F., Buzzi, M. C., Buzzi, M., e Leporini, B. (2009). Conceptual Framework: How to Engineer Online Trust for Disabled Users. Em *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 03*, WI-IAT '09, páginas 614–617, Washington, DC, USA. IEEE Computer Society.

Ali, N., Gueheneuc, Y., e Antoniol, G. (2011). Trust-Based Requirements Traceability. Em *Program Comprehension (ICPC), 2011 IEEE 19th International Conference on*, páginas 111–120.

Alves, D. D. (2011). Acessibilidade no Desenvolvimento de Software Livre. Dissertação de Mestrado, Universidade Federal de Mato Grosso do Sul. 135 páginas.

Apache (2013a). APACHE ODF TOOLKIT (INCUBATING). http://incubator.apache.org/odftoolkit/. Acessado em Setembro de 2013.

Apache (2013b). APACHE ODF TOOLKIT(INCUBATING) - SIMPLE API. http://incubator.apache.org/odftoolkit/simple/index.html. Acessado em Setembro de 2013.

Assawamekin, N., Sunetnanta, T., e Pluempitiwiriyawej, C. (2009). MUPRET: An Ontology-Driven Traceability Tool for Multiperspective Requirements Artifacts. Em *Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on*, páginas 943–948.

Astah (2013). Astah Professional. http://astah.net/editions/professional. Acessado em Setembro de 2013.

ATAG (2013a). Authoring Tool Accessibility Guidelines 1.0. http://www.w3.org/TR/ATAG10/. Acessado em Setembro de 2013.

ATAG (2013b). Authoring Tool Accessibility Guidelines (ATAG) 2.0. http://www.w3.org/TR/ATAG20/. Acessado em Setembro de 2013.

Baguma, R., Stone, R., Lubega, J., e van der Weide, T. (2009). Integrating Accessibility and Functional Requirements. Em Stephanidis, C., editor, *Universal Access in Human-Computer Interaction. Applications and Services*, volume 5616 de *Lecture Notes in Computer Science*, páginas 635–644. Springer Berlin / Heidelberg.

Bailey, J. e Burd, E. (2007). Towards More Mature Web Maintenance Practices for Accessibility. Em *Web Site Evolution, 2007. WSE 2007. 9th IEEE International Workshop on*, páginas 81 – 87.

Beck, K. (2000). *Extreme programming explained: embrace change*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Belani, H., Car, Z., e Caric, A. (2009). RUP-based process model for security requirements engineering in value-added service development. Em *Software Engineering for Secure Systems, 2009. SESS '09. ICSE Workshop on*, páginas 54 –60.

Bigham, J. P., Brudvik, J. T., e Zhang, B. (2010). Accessibility by demonstration: enabling end users to guide developers to web accessibility solutions. Em *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility*, ASSETS '10, páginas 35–42, New York, NY, USA. ACM.

Bittar, T. J., Lobato, L. L., e de Mattos Fortes, R. P. (2013). Accessibility approach to Web applications development: An Experimental Study. Em *12º Simpósio Brasileiro de Qualidade de Software (SBQS)*, Salvador - BA, Brasil.

Bonacin, R., Melo, A. M., Simoni, C. A. C., e Baranauskas, M. C. C. (2010). Accessibility and interoperability in e-government systems: outlining an inclusive development process. *Univers. Access Inf. Soc.*, 9(1):17–33.

Bonacin, R., Simoni, C. A. C., Melo, A. M., e Baranauskas, M. C. C. (2006). Organisational Semiotics: Guiding a Service-Oriented Architecture for e-Government. Em *International Conference on Organisational Semiotics (ICOS)*, páginas 47–58.

Brajnik, G. (2006). Web Accessibility Testing: When the Method Is the Culprit. Em Miesenberger, K., Klaus, J., Zagler, W. L., e Karshmer, A. I., editors, *ICCHP*, volume 4061 de *Lecture Notes in Computer Science*, páginas 156–163. Springer.

Brajnik, G. (2009). Validity and reliability of web accessibility guidelines. Em *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*, Assets '09, páginas 131–138, New York, NY, USA. ACM.

Branco, R. G. d. (2009). Ferramenta para avaliação de acessibilidade de *Web sites* baseada em métricas: uma abordagem baseada no Modelo de Acessibilidade para Governo Eletrônico do Brasil (*e-Mag*). Monografia (Graduação em Ciência da Computação), Curso de Ciência da Computação da UFMS.

Branco, R. G. d. (2013). Accessibilityin phases of Engineering Requirements and software programming: A support tool. Poster apresentado no Workshop Instituto Pan Americano de Qualidade de Software (PASQI), 15 a 26 de Julho, Michigan Technological University e Universidad de Costa Rica, San Jose e Puntarenas, Costa Rica.

Buhler, C., Heck, H., Perlick, O., Nietzio, A., e Ulltveit-Moe, N. (2006). Interpreting Results from Large Scale Automatic Evaluation of Web Accessibility. Em *ICCHP*, páginas 184–191.

Buzzi, M. C., Buzzi, M., Leporini, B., e Senette, C. (2008). Making Wikipedia editing easier for the blind. Em *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*, NordiCHI '08, páginas 423–426, New York, NY, USA. ACM.

Camargo, G. (2011). IRPF - Mudanças nas regras para 2011 - Imposto de Renda. http://www.impostoderenda.org/2011/02/16/irpf-mudancas-nas-regras-para-2011-imposto-de-renda/. Acessado em Setembro de 2013.

Carromeu, C., Paiva, D. M. B., Cagnin, M. I., Rubinsztejn, H. K. S., Turine, M. A. S., e Breitman, K. (2010). Component-Based Architecture for e-Gov Web Systems Development. Em *Proceedings of the 2010 17th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, ECBS '10, páginas 379–385, Washington, DC, USA. IEEE Computer Society.

Clark, J. (2006). To Hell with WCAG 2. http://www.alistapart.com/articles/tohellwithwcag2. Acessado em Setembro de 2013.

Cleland-Huang, J., Settimi, R., Duan, C., e Zou, X. (2005). Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability. Em *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, RE '05, páginas 135–144, Washington, DC, USA. IEEE Computer Society.

Cook, A. e Hussey, S. (1995). *Assistive technologies: principles and practice*. Mosby, Incorporated.

Cysneiros, L. M. e do Prado Leite, J. C. S. (2001). Using UML to reflect non-functional requirements. Em *Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research*, CASCON '01, páginas 2–. IBM Press.

DeGrace, P. e Stahl, L. H. (1990). *Wicked problems, righteous solutions*. Yourdon Press, Upper Saddle River, NJ, USA.

Deitel, H. e Deitel, P. (2001). *Java: como programar*. Bookman, 3ª edição.

desRivieres, J. e Wiegand, J. (2004). Eclipse: A platform for integrating development tools. *IBM Systems Journal*, 43(2):371 –383.

Dias, A. L., de Mattos Fortes, R. P., Masiero, P. C., e Goularte, R. (2010). Uma Revisão Sistemática sobre a inserção de Acessibilidade nas fases de desenvolvimento da Engenharia de Software em sistemas Web. Em *Proceedings of the IX Symposium on Human Factors in Computing Systems*, IHC '10, páginas 39–48, Porto Alegre, Brazil, Brazil. Brazilian Computer Society.

Drupal (2013). Drupal Brasil. http://drupal-br.org/. Acessado em Setembro de 2013.

EARL (2013a). Evaluation and Report Language (EARL) 1.0 Schema. http://www.w3.org/TR/EARL10-Schema/. Acessado em Setembro de 2013.

EARL (2013b). Evaluation and Report Language (EARL) Overview. http://www.w3.org/WAI/intro/earl. Acessado em Setembro de 2013.

Eclipse (2013a). Acceleo - transforming models into code. http://www.eclipse.org/acceleo/. Acessado em Setembro de 2013.

Eclipse (2013b). Eclipse Modeling Framework Project (EMF). http://www.eclipse.org/modeling/emf/. Acessado em Setembro 2013.

Eclipse (2013c). Eclipse Public License - v 1.0. http://www.eclipse.org/legal/epl-v10.html. Acessado em Setembro de 2013.

Eclipse (2013d). Ecore Tools. http://wiki.eclipse.org/index.php/Ecore_Tools. Acessado em Setembro de 2013.

Eclipse (2013e). OBEO. http://www.eclipse.org/membership/showMember.php?member_id=863. Acessado em Setembro de 2013.

Eclipse (2013f). ProR - Requirement Engineering Platform. http://www.eclipse.org/rmf/pror/. Acessado em Setembro de 2013.

Eclipse Marketplace (2013a). Requirement Designer (Indigo version) 1.0. http://marketplace.eclipse.org/node/407399/#.UeNytI2HvJK. Acessado em Setembro de 2013.

Eclipse Marketplace (2013b). UML Designer (Indigo version) 1.5. http://marketplace.eclipse.org/content/uml-designer-indigo-version/#.UeNzD42HvJJ. Acessado em Setembro de 2013.

Eclipse Marketplace (2013c). Uml to Java Generator 2.0.2. http://marketplace.eclipse.org/content/uml-java-generator/#.UeNzWI2HvJJ. Acessado em Setembro de 2013.

Encelle, B. e Baptiste-Jessel, N. (2007a). Generating Adaptable User Interfaces for Browsing XML Documents User Interfaces Adaptation Using User Profiles of Applications Policies. Em *Autonomic and Autonomous Systems, 2007. ICAS07. Third International Conference on*, página 66.

Encelle, B. e Baptiste-Jessel, N. (2007b). Personalization of user interfaces for browsing XML content using transformations built on end-user requirements. Em *Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, W4A '07, páginas 58–64, New York, NY, USA. ACM.

Faulkner, S. (2010). HTML5 and the myth of WAI-ARIA redundance. http://www.paciellogroup.com/blog/2010/04/html5-and-the-myth-of-wai-aria-redundance/. Acessado em Setembro de 2013.

Ferres, L., Verkhogliad, P., Lindgaard, G., Boucher, L., Chretien, A., e Lachance, M. (2007). Improving accessibility to statistical graphs: the iGraph-Lite system. Em *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, Assets '07, páginas 67–74, New York, NY, USA. ACM.

Ferretti, S., Mirri, S., Muratori, L. A., Roccetti, M., e Salomoni, P. (2008). E-learning 2.0: you are We-LCoME! Em *Proceedings of the 2008 international cross-disciplinary conference on Web accessibility (W4A)*, W4A '08, páginas 116–125, New York, NY, USA. ACM.

Freedom Scientific (2013a). JAWS. http://www.freedomscientific.com/jaws-hq.asp. Acessado em Setembro de 2013.

Freedom Scientific (2013b). MAGic Screen Magnification Software. http://www.freedomscientific.com/products/lv/magic-bl-product-page.asp. Acessado em Setembro de 2013.

Freire, A. P. (2008). Acessibilidade no Desenvolvimento de Aplicações Web: um estudo sobre o cenário brasileiro. Dissertação de Mestrado, Universidade de São Paulo.

Freire, A. P. (2012). *Disabled people and the Web: User-based measurement of accessibility*. Phd em ciência da computação, University of York.

Fuertes, J. L., Gutiérrez, E., e Martínez, L. (2011). Developing Hera-FFX for WCAG 2.0. Em *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, W4A '11, páginas 3:1–3:9, New York, NY, USA. ACM.

Garrett, J. J. (2005). Ajax: A New Approach to Web Applications. http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications. Acessado em Setembro de 2013.

Giakoumis, D., Votis, K., Tzovaras, D., Likothanassis, S., e Hassapis, G. (2010). Introducing accessibility in the Web services domain. Em *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 2, páginas 18 –22.

Goodman, D. (2001). *JavaScript: a bíblia*. Campus.

Google (2011). Google Accessibility. http://www.google.com/accessibility/labs/search/. Acessado em Setembro de 2013.

Google (2013). Accessibility in Google Search. https://support.google.com/websearch/answer/181196?hl=en. Acessado em Setembro de 2013.

Gotel, O. C. Z. e Finkelstein, A. C. W. (1994). An analysis of the requirements traceability problem. Em *Requirements Engineering, 1994., Proceedings of the First International Conference on*, páginas 94–101.

Governo Eletrônico (2007). e-MAG - Modelo de Acessibilidade de Governo Eletrônico. http://www.governoeletronico.gov.br/acoes-e-projetos/e-MAG. Acessado em Setembro de 2013.

Groves, K. (2011). How expensive is web accessibility? http://www.karlgroves.com/2011/11/30/how-expensive-is-accessibility/. Acessado em Setembro de 2013.

Groves, K. (2012). Web Accessibility Testing: Do Automatic Testing First. http://www.karlgroves.com/2012/02/02/web-accessibility-testing-do-automatic-testing-first/. Acessado em Setembro de 2013.

Gruber, T. R. *et al.* (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220.

guo, Y., Yang, M., Wang, J., Yang, P., e Li, F. (2009a). An Ontology Based Improved Software Requirement Traceability Matrix. Em *Proceedings of the 2009 Second International Symposium on Knowledge Acquisition and Modeling - Volume 01*, KAM '09, páginas 160–163, Washington, DC, USA. IEEE Computer Society.

guo, Y., Yang, M., Wang, J., Yang, P., e Li, F. (2009b). An Ontology Based Improved Software Requirement Traceability Matrix. Em *Knowledge Acquisition and Modeling, 2009. KAM '09. Second International Symposium on*, volume 1, páginas 160–163.

Halbach, T. (2010). Towards Cognitively Accessible Web Pages. Em *Proceedings of the 2010 Third International Conference on Advances in Computer-Human Interactions*, ACHI '10, páginas 19–24, Washington, DC, USA. IEEE Computer Society.

Hovater, S. (2008). UML-requirements traceability using IBM Rational RequisitePro, IBM Rational Software Architect, and BIRT, Part 1: Reporting requirements. Acessado em Setembro de 2013.

IBM (2011). Rational Policy Tester Accessibility Edition. http://www-01.ibm.com/software/awdtools/tester/policy/accessibility/. Acessado em Setembro de 2013.

IBM (2013a). ACCMD Briefing. http://wiki.cetis.ac.uk/ACCMD_Briefing. Acessado em Setembro de 2013.

IBM (2013b). IBM Rational Unified Process (RUP). http://www-01.ibm.com/software/awdtools/rup/. Acessado em Setembro de 2013.

IBM (2013c). Rational RequisitePro. http://www-03.ibm.com/software/products/br/pt/reqpro/. Acessado em Setembro de 2013.

IBM (2013d). Rational Software Architect. http://www-03.ibm.com/software/products/br/pt/ratisoftarch/. Acessado em Setembro de 2013.

IBM (2013e). WebSphere software. http://www-01.ibm.com/software/websphere/. Acessado em Setembro de 2013.

IEEE/EIA (1998). IEEE/EIA 12207 - Industry Implementation of International Standard ISO/IEC 12207 : 1995.

IETF (2013). The Internet Engineering Task Force (IETF). http://www.ietf.org/. Acessado em Setembro de 2013.

IMS (2013). IMS Access For All v2.0 Final Specification. http://www.imsglobal.org/accessibility/. Acessado em Setembro de 2013.

Irish, P. (2011). Semantics in practice and mapping semantic value to its consumers. http://paulirish.com/2011/semantics/. Acessado em Setembro de 2013.

IRPF (2011). Imposto de Renda 2011 só poderá ser declarado pela internet. http://irpf.estacaobr.net/imposto-de-renda-2011-so-podera-ser-declarado-pela-internet/. Acessado em Setembro de 2013.

IRPF (2013). Declaracao Imposto de Renda 2013 - Como Declarar. http://impostoderenda2013.com/declaracao-imposto-de-renda-2013-como-declarar.html. Acessado em Setembro de 2013.

ISO/IEC (1998). *ISO/IEC 12207 - Standard for Informational Technology - Software Lifecycle Processes*. ISO/IEC, 1, ch. de la Voie-Creuse - CP 56 - CH-1211 Geneva 20 - Switzerland.

ISO/IEC (2012). *Information technology - W3C Web Content Accessibility Guidelines (WCAG) 2.0*. ISO/IEC, Case postale 56 - CH-1211 Geneva 20, Switzerland.

Jacobson, I., Booch, G., e Rumbaugh, J. (1999). *The unified software development process*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Jain, H. (2013). Traceability Matrix. https://sites.google.com/site/himanijain84/testing/traceability. Acessado em Setembro de 2013.

JavaCC (2011). Java Compiler Compiler [tm] (JavaCC [tm]) - The Java Parser Generator. http://javacc.java.net/. Acessado em Setembro de 2013.

Joomla (2013). Joomla! http://www.joomla.org/. Acessado em Setembro de 2013.

Kavcic, A. (2005). Software Accessibility: Recommendations and Guidelines. Em *Computer as a Tool, 2005. EUROCON 2005.The International Conference on*, volume 2, páginas 1024 –1027.

Lazar, J., Dudley-Sponaugle, A., e Greenidge, K.-D. (2004). Improving web accessibility: a study of webmaster perceptions. *Computers in Human Behavior*, 20(2):269–288.

Lee, J., Cho, B., Youn, H., e Lee, E. (2009). Reliability Analysis Method for Supporting Traceability Using UML. Em *Advances in Software Engineering*, volume 59 de *Communications in Computer and Information Science*, páginas 94–101. Springer Berlin Heidelberg.

LYNX (1998). Lynx. http://lynx.browser.org/. Acessado em Setembro de 2013.

Mader, P. e Egyed, A. (2012). Assessing the effect of requirements traceability for software maintenance. Em *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, páginas 171–180.

Maia, L. S. (2010). Um processo para o desenvolvimento de aplicações Web Acessíveis. Dissertação de Mestrado, Universidade Federal de Mato Grosso do Sul. 94 páginas.

Manohar, P. e Parthasarathy, A. (2009). An Innovative Braille System Keyboard for the Visually Impaired. Em *Computer Modelling and Simulation, 2009. UKSIM '09. 11th International Conference on*, páginas 559 –562.

Martín García, Y. S., Miguel González, B. S., e Yelmo García, J. C. (2009). Prosumers and accessibility: how to ensure a productive interaction. Em *Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibililty (W4A)*, W4A '09, páginas 50–53, New York, NY, USA. ACM.

Martins, J. e Machado, R. (2012). Ontologies for Product and Process Traceability at Manufacturing Organizations: A Software Requirements Approach. Em *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*, páginas 353–358.

Martínez, A. B., Juan, A. A., Álvarez, D., e Suárez, M. C. (2009). WAB*: A Quantitative Metric Based on WAB. Em *ICWE '9: Proceedings of the 9th International Conference on Web Engineering*, páginas 485–488, Berlin, Heidelberg. Springer-Verlag.

Masuwa-Morgan, K. (2008). Introducing AccessOnto: Ontology for Accessibility Requirements Specification. Em *Ontologies in Interactive Systems, 2008. ONTORACT '08. First International Workshop on*, páginas 33 –38.

McPherson, S. S. (2009). *Tim Berners-Lee: Inventor of the World Wide Web*. USA Today Lifeline Biographies. Twenty First Century Books.

Melo, A. M. e Baranauskas, M. C. C. (2006). Design para a inclusão: desafios e proposta. Em *Proceedings of VII Brazilian symposium on Human factors in computing systems*, IHC '06, páginas 11–20, New York, NY, USA. ACM.

Michail, S. e Christos, K. (2007). Adaptive Browsing Shortcuts: Personalising the User Interface of a Specialised Voice Web Browser for Blind People. Em *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*, páginas 818–825, Washington, DC, USA. IEEE Computer Society.

Microsoft (2009). *Engineering Software for Accessibility*. O'Reilly Media, Inc.

Mikic, F., Anido, L., Valero, E., e Picos, J. (2007). Accessibility and Mobile Learning Standardization. Em *Systems, 2007. ICONS '07. Second International Conference on*, página 32.

Moodle (2013). Moodle. http://moodle.org/. Acessado em Setembro de 2013.

Moreno, L., Martínez, P., e Ruiz-Mezcua, B. (2009). Integrating HCI in a Web Accessibility Engineering Approach. Em Stephanidis, C., editor, *Universal Access in Human-Computer Interaction. Applications and Services*, volume 5616 de *Lecture Notes in Computer Science*, páginas 745–754. Springer Berlin / Heidelberg. 10.1007/978-3-642-02713-0_79.

Moulin, C. e Sbodio, M. (2010). Improving the accessibility and efficiency of e-Government processes. Em *Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on*, páginas 603–610.

Mozilla (2012). Text to Voice. https://addons.mozilla.org/pt-br/firefox/addon/text-to-voice/. Acessado em Setembro de 2013.

NCE-UFRJ (2002). Projeto DOSVOX. http://intervox.nce.ufrj.br/dosvox/. Acessado em Setembro de 2013.

NCSU (1997). The principles of universal design. http://www.ncsu.edu/ncsu/design/cud/pubs_p/pud.htm. Acessado em Setembro de 2013.

NDA (1999). The National Disability Authority. http://www.nda.ie/. Acessado em Setembro de 2013.

NE10 (2011). Bancos podem oferecer conta sem tarifa a partir desta terça-feira. http://jc.uol.com.br/canal/cotidiano/economia/noticia/2011/02/28/bancos-podem-oferecer-conta-sem-tarifa-a-partir-desta-tercafeira-259223.php. Acessado em Setembro de 2013.

Newton, A. (2008). *MooTools Essentials: The Official MooTools Reference for JavaScript and Ajax Development*. Apresspod Series. Apress.

NIDirect (1995). The Disability Discrimination Act (DDA). http://www.nidirect.gov.uk/the-disability-discrimination-act-dda. Acessado em Setembro de 2013.

Noll, R. e Ribeiro, M. (2007). Ontological Traceability over the Unified Process. Em *Engineering of Computer-Based Systems, 2007. ECBS '07. 14th Annual IEEE International Conference and Workshops on the*, páginas 249–255.

Noy, N. F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R. W., e Musen, M. A. (2001). Creating Semantic Web Contents with Protege-2000. Em *Protegé-2000. IEEE Intelligent Systems (2001*, páginas 60–71.

Obeo Network (2013). Queries. http://www.obeonetwork.com/page/the-acceleo-queries. Acessado em Setembro de 2013.

Oliveira, B. (2011). Conheça As Vantagens De Vender Pela Internet. http://www.artigonal.com/negocios-online-artigos/conheca-as-vantagens-de-vender-pela-internet-693945.html. Acessado em Setembro de 2013.

Oliveira, R. C. d. (2013). Homero: Um framework de apoio ao desenvolvimento de interfaces de aplicações Web acessíveis. Dissertação de Mestrado, Universidade Federal de Mato Grosso do Sul.

OMG (2013). Requirements Interchange Format (ReqIF). http://www.omg.org/spec/ReqIF/. Acessado em Setembro de 2013.

Open Source Software (2009). LEA - Lightweight Eyetracking Algorithm. http://lea-eyetracking.sourceforge.net/. Acessado em Setembro de 2013.

Opera Software (2013). Opera version history. http://www.opera.com/docs/history/#o10. Acessado em Setembro de 2013.

Orchard, L., Pehlivanian, A., Koon, S., e Jones, H. (2010). *Professional JavaScript Frameworks: Prototype, YUI, ExtJS, Dojo and MooTools*. Wiley.

Parmanto, B. e Zeng, X. (2005). Metric for Web accessibility evaluation. *JASIST*, 56(13):1394–1404.

Pauwels, S. L., Hübscher, C., Leuthold, S., Bargas-Avila, J. A., e Opwis, K. (2009). Error prevention in online forms: Use color instead of asterisks to mark required-fields. *Interact. Comput.*, 21:257–262.

Planalto (2000a). LEI No 10.048, DE 8 DE NOVEMBRO DE 2000. http://www.planalto.gov.br/ccivil_03/leis/l10048.htm. Acessado em Setembro de 2013.

Planalto (2000b). LEI No 10.098, DE 19 DE DEZEMBRO DE 2000. http://www.planalto.gov.br/ccivil_03/Leis/L10098.htm. Acessado em Setembro de 2013.

Planalto (2004). DECRETO Nº 5.296 DE 2 DE DEZEMBRO DE 2004. http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2004/decreto/d5296.htm. Acessado em Setembro de 2013.

Plone (2013). Plone. http://plone.org/. Acessado em Setembro de 2013.

Resig, J. (2006). *Pro JavaScript Techniques*. Expert's voice in Web development. Apress.

Rey, C. (2002). *Macromedia Flash MX: Training from the Source*. Training from the source. Macromedia Press.

RSS (2011). RSS Specifications. http://www.rss-specifications.com/rss-specifications.htm. Acessado em Setembro de 2013.

Sandim, H. d. C. (2009). Pantaneiro: Um framework para desenvolvimento de webapps em uma plataforma e-gov. Dissertação de Mestrado, Universidade Federal de Mato Grosso do Sul.

Sarcar, S., Ghosh, S., Saha, P., e Samanta, D. (2010). Virtual keyboard design: State of the arts and research issues. Em *Students' Technology Symposium (TechSym), 2010 IEEE*, páginas 289 –299.

Schutta, N. e Asleson, R. (2006). *Pro Ajax and Java Frameworks*. Expert's voice in Web development. Apress.

SeEBrowser (2011). SeEBrowser (Semantically Enhanced Browser). http://seebrowser.it.teithe.gr/?q=en/node/32. Acessado em Setembro de 2013.

Select (2013). Select Architect (BMM, BPMN, UML). http://www.selectbs.com/analysis-and-design/select-architect. Acessado em Setembro de 2013.

Sherman, P. (2001). Cost-Justifying Accessibility. Austin: Austin Usability. http://www.gslis.utexas.edu/~I385t21/AU_WP_Cost_Justifying_Accessibility.pdf. Acessado em Setembro de 2013.

Sidar (2003). Revendo a Acessibilidade com Estilo. http://www.sidar.org/hera/. Acessado em Setembro de 2013.

Sirithumgul, P., Suchato, A., e Punyabukkana, P. (2009). Quantitative evaluation for web accessibility with respect to disabled groups. Em *W4A '09: Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibililty (W4A)*, páginas 136–141, New York, NY, USA. ACM.

SMIL (2013). Synchronized Multimedia Integration Language (SMIL 3.0). http://www.w3.org/TR/smil/. Acessado em Setembro de 2013.

SOAP (2013). SOAP Version 1.2 Part 0: Primer. http://www.w3.org/TR/2003/REC-soap12-part0-20030624/. Acessado em Setembro de 2013.

Soonsongtanee, S. e Limpiyakorn, Y. (2010). Enhancement of requirements traceability with state diagrams. Em *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 2, páginas V2–248–V2–252.

Sparx Systems (2013). Sparx Systems. http://www.sparxsystems.com/enterprise_architect_user_guide/10/. Acessado em Setembro de 2013.

Sun, Z. e Zhang, J. (2009). On Accessibility of Concept, Principle and Model of Educational Web Sites Design. Em *New Trends in Information and Service Science, 2009. NISS '09. International Conference on*, páginas 730 –733.

Tarr, A. (2011). WAI-ARIA Roles in Accessible Admin Template. http://community.joomla.org/blogs/community/963-wai-aria-roles-in-accessible-admin-template.html. Acessado em Setembro de 2013.

Teague, J. (2001). *DHTML and CSS for the World Wide Web*. Visual quickstart guide. Peachpit Press.

Tennison, J. (2001). *XSLT and XPath On The Edge*. M and T Books on the Edge Series. Wiley.

Thatcher, J., Burks, M. R., e Heilmann, C. (2006). *Web Accessibility: Web Standards and Regulatory Compliance*. Friends of Ed.

The Document Foundation (2013). Libre Office. http://pt-br.libreoffice.org/. Acessado em Setembro de 2013.

Thinyane, H. e Thinyane, M. (2009). ICANSEE: A SIM based application for digital inclusion of the visually impaired community. Em *Innovations for Digital Inclusions, 2009. K-IDI 2009. ITU-T Kaleidoscope:*, páginas 1 –6.

TinyMCE (2013). TinyMCE - Home. http://www.tinymce.com. Acessado em Setembro de 2013.

TJCE (2011a). Tribunal de Justiça do Estado do Ceará - Emissão de certidão criminal negativa. http://www4.tjce.jus.br/siscertidao/. Acessado em Setembro de 2013.

TJCE (2011b). Tribunal de Justiça do Estado do Ceará - Portaria N.617/2008. http://www4.tjce.jus.br/siscertidao/portaria6172008.pdf. Acessado em Julho de 2012.

Treasury Board of Canada Secretariat (2013). Web Standards for the Government of Canada. http://www.tbs-sct.gc.ca/ws-nw/index-eng.asp. Acessado em Setembro de 2013.

Trewin, S., Cragun, B., Swart, C., Brezin, J., e Richards, J. (2010). Accessibility challenges and tool features: an IBM Web developer perspective. Em *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, W4A '10, páginas 32:1–32:10, New York, NY, USA. ACM.

UAAG (2013a). User Agent Accessibility Guidelines 1.0. http://www.w3.org/TR/UAAG10/guidelines.html#Guidelines. Acessado em Setembro de 2013.

UAAG (2013b). User Agent Accessibility Guidelines (UAAG) 2.0. http://www.w3.org/TR/UAAG20/. Acessado em Setembro de 2013.

UAAG (2013c). User Agent Accessibility Guidelines (UAAG) Overview. http://www.w3.org/WAI/intro/uaag.php. Acessado em Setembro de 2013.

United States Government (2009). Section 508: 508 Law. http://www.section508.gov/index.cfm?FuseAction=Content&ID=3. Acessado em Setembro de 2013.

U.S. Department of Justice (2011). Section 508. http://www.justice.gov/crt/508/. Acessado em Setembro de 2013.

van Schaik, P. e Ling, J. (2008). Modelling user experience with web sites: Usability, hedonic value, beauty and goodness. *Interact. Comput.*, 20:419–432.

Vigo, M., Arrue, M., Brajnik, G., Lomuscio, R., e Abascal, J. (2007). Quantitative metrics for measuring web accessibility. Em *W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, páginas 99–107, New York, NY, USA. ACM.

Vigo, M. e Brajnik, G. (2011). Automatic web accessibility metrics: Where we are and where we can go. *Interact. Comput.*, 23:137–155.

Visual Paradigm (2013). Visual Paradigm. http://www.visual-paradigm.com/. Acessado em Setembro de 2013.

Votis, K., Oikonomou, T., Korn, P., Tzovaras, D., e Likothanassis, S. (2009). A visual impaired simulator to achieve embedded accessibility designs. Em *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 3, páginas 368 –372.

W3C (2013a). Authoring Tool Accessibility Guidelines (ATAG) Overview. http://www.w3.org/WAI/intro/atag.php. Acessado em Setembro de 2013.

W3C (2013b). OWL 2 Web Ontology Language Document Overview (Second Edition). http://www.w3.org/TR/owl2-overview/. Acessado em Setembro de 2013.

W3C (2013c). OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features/. Acessado em Setembro de 2013.

W3C (2013d). Resource Description Framework (RDF). http://www.w3.org/RDF/. Acessado em Setembro de 2013.

W3C (2013e). Techniques for WCAG 2.0. http://www.w3.org/TR/WCAG20-TECHS/. Acessado em Setembro de 2013.

W3C (2013f). Web Content Accessibility Guidelines 1.0. http://www.w3.org/TR/WCAG10/. Acessado em Setembro de 2013.

W3C (2013g). World Wide Web Consortium. http://www.w3.org/. Acessado em Setembro de 2013.

WAB Cluster (2009). Unified Web Evaluation Methodology version 0.5. http://www.wabcluster.org/uwem05/. Acessado em Setembro de 2013.

WAI (2013). Web Accessibility Initiative (WAI). http://www.w3.org/WAI/. Acessado em Setembro de 2013.

WAI-ARIA (2013). WAI-ARIA Overview. http://www.w3.org/WAI/intro/aria.php. Acessado em Setembro de 2013.

WCAG (2013a). How WCAG 2.0 Differs from WCAG 1.0. http://www.w3.org/WAI/WCAG20/from10/diff.php. Acessado em Setembro de 2013.

WCAG (2013b). WCAG Overview. http://www.w3.org/WAI/intro/wcag. Acessado em Setembro de 2013.

WCAG (2013c). Web Content Accessibility Guidelines (WCAG) 2.0. http://www.w3.org/TR/WCAG20/. Acessado em Setembro de 2013.

WCAG Samurai (2006). WCAG Samurai. http://wcagsamurai.org/. Acessado em Setembro de 2013.

WebAnywhere (2013). WebAnywhere - A Screen reader on the go. http://webanywhere.cs.washington.edu/. Acessado em Setembro de 2013.

Webcredible (2011). Disability Discrimination Act (DDA) & web accessibility. http://www.webcredible.co.uk/user-friendly-resources/web-accessibility/uk-website-legal-requirements.shtml. Acessado em Setembro de 2013.

Wikipedia (2013). List of wiki software. http://en.wikipedia.org/wiki/List_of_wiki_software. Acessado em Setembro de 2013.

Wordpress (2013). Wordpress — Brasil. http://br.wordpress.org/. Acessado em Setembro de 2013.