UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

MAMA: A Model-driven Approach Minding Accessibility

Lianna Mara Castro Duarte

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP SÃO CARLOS

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Lianna Mara Castro Duarte

MAMA: A Model-driven Approach Minding Accessibility

Thesis submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Doctor in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Renata Pontin de Mattos Fortes

USP – São Carlos
April 2022

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Dm

DUARTE, LIANNA MARA
MAMA - A Model-driven Approach Minding Accessibility / LIANNA MARA DUARTE; orientador Renata Pontin de Mattos Fortes. -- São Carlos, 2022.
140 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2022.

1. Acessibilidade. 2. Desenvolvimento Orientado a Modelos. 3. Desenvolvimento Móvel. I. Pontin de Mattos Fortes, Renata , orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Lianna Mara Castro Duarte

MAMA: uma abordagem orientada a modelos visando acessibilidade

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutora em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Renata Pontin de Mattos Fortes

USP – São Carlos
Abril de 2022

Este trabalho é dedicado a minha família, especialmente, aos meus pais Gilmar e Elda, ao meu marido Márcio e minhas filhas Sarah e Marília que proporcionaram todo o apoio necessário para conclusão deste desafio. Dedico também aos meus queridos amigos ajudaram ao longo desta jornada e que, de alguma forma, colaboraram para esta conquista.

ACKNOWLEDGEMENTS

Agradeço a Deus em primeiro lugar que nunca me abandona nas horas mais difíceis.

A minha Orientadora Dra Renata Pontin de Mattos Fortes, pela oportunidade, pela orientação, paciência, compreensão, disponibilidade e dedicação.

Agradeço à minha família pelo apoio, incentivo , em especial a meu pai Gilmar, minha mãe Elda e minhas irmãs Lanna, Liannara e Lannara pelas palavras de incentivo em todos os momentos da minha vida.

Ao meu marido Márcio pelo amor, paciência e incentivo (Te amo!) e as minhas filhas Sarah e Marília.

A meus colegas de trabalho e de programa que me incentivaram a continuar nesta caminhada, em especial ao Prof. Dr. Thiago Carvalho de Sousa grande incentivador e a minha colega de laboratório Flavia pelo incentivo constante.

Agradeço a UESPI e a USP pela oportunidade deste programa de doutorado interinstitucional.

A todos os que contribuíram, incentivaram, torceram e lembraram de mim em suas orações, para que eu conseguisse chegar nesta etapa da minha vida.

“Mas eu sei que um dia a gente aprende
Se você quiser alguém em quem confiar
Confie em si mesmo
Quem acredita sempre alcança! ”
(Renato Russo)

RESUMO

DUARTE, L. M. C. **MAMA: uma abordagem orientada a modelos visando acessibilidade** . 2022. 146 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2022.

A necessidade de promover oportunidades iguais para todos os usuários de sistemas interativos tem sido uma preocupação constante. Com a ampla popularização das tecnologias da Internet pelo mundo, acessibilidade tornou-se cada vez mais essencial para manter as pessoas capazes de se comunicar e se beneficiar de todas as vantagens. Embora a acessibilidade seja geralmente exigida por lei, o desenvolvimento de aplicativos acessíveis requer um conhecimento bastante especializado. Na literatura, muitos pesquisadores propõem abordagens diferentes para considerar a acessibilidade durante a geração de aplicativos e, que com a adoção do desenvolvimento orientado a modelos torna-se menos custoso. Com base nos estudos desses trabalhos, foi possível observar que alguns deles apenas apresentam definições teoricamente, outros não consideram meios para apoiar recursos de acessibilidade e outros que implementam acessibilidade por meio de transformações ao invés de modelá-los desde um nível abstrato. Por outro lado, esses estudos forneceram insights para esta investigação. Assim, propomos uma abordagem que conduz a produção de aplicativos considerando acessibilidade para aliviar a sobrecarga de conhecimentos especializados dos desenvolvedores. `MAMA`, significa, no acrônimo em inglês, **M**odel-driven **A**pproach **M**inding **A**ccessibility, abordagem orientada a modelos tendo em mente a acessibilidade. `MAMA` foi formulada com base em IFML e estendendo sua modelagem nos níveis adequados de abstração. Para evidenciar a viabilidade da adoção de `MAMA`, descrevemos as razões para cada especialização dos artefatos e apresentamos um aplicativo móvel acessível gerado passo a passo.

Palavras-chave: Desenvolvimento Orientado a Modelos, Acessibilidade, Desenvolvimento Móvel.

ABSTRACT

DUARTE, L. M. C. **MAMA: A Model-driven Approach Minding Accessibility**. 2022. 146 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2022.

The need to promote equal opportunity for all users of interactive systems has been a constant concern. Since Internet technologies have spread worldwide, accessibility is essential to allow people able to communicate and benefit from all the advantages. Even though accessibility is often required by law, developing accessible apps requires much-specialized knowledge. Many researchers have proposed different approaches to consider accessibility during the generation of applications, and simultaneously, having a model-driven development becomes less costly. Studies of scientific literature regarding these approaches resulted in a few works with only theoretical definitions, others with no consideration to support accessibility features, and another implemented through transformations rather than modeling them at an abstract level. On the other hand, these studies provided insights to this investigation. We have proposed an approach that conduces application production considering accessibility to alleviate this knowledge overload on developers. MAMA stands for a **M**odel-driven **A**pproach **M**inding **A**ccessibility. The MAMA was formulated based on IFML (Interaction Flow Modeling Language) and extending its modeling in the adequate levels of abstraction. To evidence the feasibility of the MAMA adoption, we describe the rationale for each specialization of the artifacts and present an accessible mobile app step by step generated.

Keywords: Model-driven Development, Accessibility, Mobile Development.

LIST OF FIGURES

LIST OF CHARTS

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

MAMA	A **M**odel-driven **A**pproach **M**inding **A**ccessibility
AccDesignUI	Accessible Design User Interface
AccUI	Accessible User Interface
ADA	Americans with Disabilities Act
API	Application Programming Interface
AT	Assistive Technology
ATAG	Authoring Tool Accessibility Guidelines
ATL	Atlas Transformation Language
AUI	Abstract User Interface
AWA	*Accessibility for Web Applications*
BBC	British Broadcasting Corporation
BPMN	Business Process Model And Notation
CIM	Computational Independent Model
CRUD	Create Read Update and Delete
CRUD	Create, Read, Update, Delete
CUI	Concrete User Interface
DAO	Data Access Object
DSL	Domain Specific Languages
EMF	Eclipse Modeling Framework
EMF	Eclipse Modeling Framework
EP	Execution Platform
fUML	Foundational UML
GUI	Graphical User Interfaces
IBGE	Brazilian Institute of Geography and Statistics
IFML	Interaction Flow Modeling Language
IP	Interaction Platform
IPs	Integration Points
JSF	Java Server Faces
M2M	Model to Model
M2T	Model to Text
MAML	Münster App Modeling Language

maRs	mobile accessibility Recommendations
MDA	Model Driven Architecture
MDD	Model-Driven Development
MDE	Model-Driven Engineering
MDSD	Model-Driven Software Development
MOF	Meta Object Facility
MTA	Methodology for Developing Accessible Web Applications
MTL	Model to Text Language
MVC	Model View Controller
MVVM	Model View ViewModel
OCL	Object Constraint Language
ODM	Ontology Definition Metamodel
OMG	Object Management Group
PDWAU	*Process for Developing Web system with Accessibility and Usability*
PIM	Platform Independent Model
PoC	Proof of Concept
PSM	Platform Specific Model
RIA	Rich Internet Application
SCI	Spinal Cord
SysML	OMG System Modeling Language
TBI	Head Injuries
TDD	Telecommunication Device for the Deaf
UAAG	User Agent Accessibility Guidelines
UI	User Interface
UID	User Interaction Diagrams
UIDL	UI Description Language
UIDL	User Interface Description Language
UIDs	User Interaction Diagrams
UML	Unified Modeling Language
US	United States
UsiXML	USer Interface eXtensible Markup Language
UX	User EXperience
VRS	Video Relay Service
W3C	World Wide Web Consortium
WAI	Web Accessibility Initiative
WAI-ARIA	Accessible Rich Internet Applications
WCAG	Web Content Accessibility Guidelines

WebAIM	Web Accessibility in Mind
WHO	World Health Organization

CONTENTS

CHAPTER

1

INTRODUCTION

1.1. Context and Motivation

Technology and the Internet have become an integral part of people's lives. In many countries, the scenario shows the increasing reliance on information technology products and services in people's lives (KAUFHOLD *et al.*, 2021). The Internet is increasingly used for administrative information and services, education and training, commerce, news, workplace interaction, civic participation, healthcare, recreation, entertainment (CALVO; SEYEDARABI; SAVVA, 2016), and so on. The importance of being connected became even more evident with the outbreak of COVID-19. Internet connectivity helped people in routine activities such as maintaining business operations, communicating with friends and relatives, ensuring online access to essential goods and services (DE'; PANDEY; PAL, 2020), children's education, and home office, which accelerated the forms of human interaction and showed its global significance.

Statistical data show that the number of Internet users worldwide in 2019 was 4.13 billion, which means that more than half of the world's population is currently connected to the Internet (STATISTA, 2021). At the beginning of 2021, the global population stood at 7.83 billion. By January 2021, there were 4.66 billion active Internet users worldwide (KEMP, 2021). Of these, 92.6% (4.32 billion) accessed the Internet via mobile devices.

Today, 5.22 billion people use mobile phones, representing 66.6% of the world's population. The number of unique mobile users has increased by 1.8% (93 million) since January 2020, while the total number of mobile connections has increased by 72 million (0.9%) to a total of 8.02 billion at the beginning of 2021 (KEMP, 2021). The number of smartphone users now exceeds three billion worldwide and is expected to increase by several hundred million over the next few years.

The need to promote equal opportunity for all users of interactive systems is not a new concern, and work on web accessibility has been in the literature for more than twenty

years (BERNERS-LEE, 1997; PACIELLO, 2000). Accessibility is the extent to which products, systems, services, environments, and facilities can be used by people from a population with the broadest range of user needs, characteristics, and capabilities to achieve identified goals in identified contexts of use (ISO 9241-11, 2018). In several countries, it is now a legal requirement that establishes responsibility for accessing content on the Web (SKJERVE; GIANNOUMIS; NASEEM, 2016; XIONG; FARENC; WINCKLER, 2007; MORENO; MARTINEZ, 2019).

Over the past 15 years, both the United States and international governments have developed legislation to ensure equal rights for people with disabilities, including equal access to electronic technology and information technology (Level Access, 2021). However, many users are currently left behind and denied full membership in the information society due to barriers that prevent them from fully accessing information systems (MORENO; MARTINEZ, 2019). In this context, several people with disabilities face barriers to interacting with various software applications.

World Health Organization (WHO) statistics show that approximately 15% of the world's population has some form of disability (World Health Organization *et al.*, 2011). In the United States, the number of people with disabilities comprised 13.1% of the population in 2018 (HOUTENVILLE; BOEGE, 2020). In Europe, one-fifth of the population is expected to have some form of disability by 2020 (European Commission, 2021). In Brazil, the latest census from the Brazilian Institute of Geography and Statistics (IBGE) indicates that about 23% of Brazilians declare themselves with some disability (IBGE, 2010). In addition, the elderly population in the world is increasing exponentially, leading to a natural loss of abilities over the years, with many disabilities showing up later in life. For this reason, the elderly population has also been included as a contingent with disabilities (World Health Organization *et al.*, 2015).

All this data exposes how exclusive the digital world can be, especially for people with disabilities. Accessibility is a requirement for any user who wants to interact with software and is essential for developers and organizations that want to create high-quality software (W3C, 2019). The benefits of accessibility in computer systems extend beyond people with disabilities. If all types of interactive applications are designed with accessibility in mind, they can provide a better experience for all users.

Using *Alexa*[1] as a data source, a study conducted in January 2018 examined the navigation and purchase process on the most visited e-commerce sites in Brazil. It found that 99% of websites have navigation barriers for people with disabilities (WPT, 2019). Recent research in this area concluded that awareness of accessibility and knowledge of compliance requirements or guidelines among professionals is still low (DUARTE *et al.*, 2016; LAZAR, 2019; RIEGER *et al.*, 2020; PATEL *et al.*, 2020; LEITE *et al.*, 2021).

Leite *et al.* (2021) surveyed 872 people involved in mobile application development in the

[1] <https://www.alexa.com/>

Brazilian industry. Participants indicated that the reasons why accessibility is not considered in their projects include: missing requirements, insufficient time, lacking training, and no focus on users with disabilities. However, there are several factors that can influence the implementation of digital accessibility, such as the knowledge of designers and developers, who usually justify how they deal with accessibility by reporting factors like the size of the company and the different resources such as time and project costs (PAIVA; FREIRE; FORTES, 2021) and inadequate resources (appropriate development methods, supporting tools) and experts (BI *et al.*, 2021).

As software systems become more complex over time, it is challenging to integrate accessibility into the development process. Accessibility requirements fit into different phases of the software. The usual approach is to modify the design at the end of the development process (W3C, 2014; MARTÍN; CECHICH; ROSSI, 2011; GRECO, 2019), which can make implementation difficult. Gaggi, Quadrio and Bujari (2019) concluded that there is still a long way to go to achieve a more accessible web. Creating tools to help web designers and developers with this task could significantly improve the current situation.

Due to the importance of accessibility and in order to understand the negative factors and propose approaches to integrate accessibility at early stages and throughout the development process, several research studies have been developed (MOLINA; TOVAL, 2009; MARTÍN; CECHICH; ROSSI, 2011; PANACH; AQUINO; PASTOR, 2014; BRANCO; CAGNIN; PAIVA, 2014; DIAS, 2014; OLIVEIRA *et al.*, 2016; PATEL *et al.*, 2020; BI *et al.*, 2021; LEITE *et al.*, 2021). All of this suggests that developers need to consider accessibility in their project decisions to provide accessible User Interface (UI) interactions, that this is not a trivial task, is still a big challenge.

Another problem is that software modeling language standards have neglected user interaction with computer systems by not supporting user interface and interaction modeling instead of focusing on architectural issues and the back-end. Thus, developing the front-end of applications has become a costly and inefficient process where manual coding is still the predominant development approach. As a result, reuse is low and cross-platform portability remains a challenge, and to constant rework and refinements to the implementation (OMG, 2015a; BRAMBILLA; FRATERNALI, 2014a).

Support with a standardized language emerged in 2015 with Interaction Flow Modeling Language (IFML) (OMG, 2015a), when the Object Management Group (OMG) architecture board formally adopted the language specification. The main goal of IFML is to provide resources for system architects, software engineers, and software developers, to define user interaction flow models that describe the main dimensions of the application front-end (OMG, 2015a). The language is fully integrated with other OMG specifications like Unified Modeling Language (UML), OMG System Modeling Language (SysML), and Business Process Model And Notation (BPMN), that implements the Model Driven Architecture (MDA) framework.

MDA is a realization of the Model-Driven Engineering (MDE) paradigm spearheaded by

the OMG. The aim is to have an approach to the design, development, and implementation of software. MDA provides guidelines for structuring software specifications that are expressed as models (GROUP, 2021). Model-Driven Engineering, Model-Driven Development (MDD), and Model-Driven Software Development (MDSD) are used synonymously in the literature, referred in this work as MDD. The MDD paradigm has been a significant part of software engineering. Feedback from users and different industries indicate the need for this interconnection in the technical industry among User EXperience (UX), UI (User Interface) and MDD UI helps MDD models to be deployed on different user platforms (AGGARWAL *et al.*, 2021).

In general terms, accessibility is deemed an extra time and resource cost (PATEL *et al.*, 2020), being overlooked and not considered at the start of a software development project. An attractive alternative that could deal with the current challenges of accessible applications is the MDD (Model-Driven Development) approach, which has drawn the accessible software development community's attention due to its ability to generate code from models (ORDOÑEZ; HILERA; CUEVA, 2020). Another advantage is that with models, we can represent a system from different points of view/levels independently of any development or implementation and use the artifacts throughout the development process in different abstraction levels, which can help the awareness of accessibility concepts.

Recent studies investigate the generation of interfaces with MDD approaches (ORDOÑEZ; HILERA; CUEVA, 2020; GAMITO; SILVA, 2020; AGGARWAL *et al.*, 2021; RIEGER *et al.*, 2020), the integration of quality requirements in the development process, market support and acceptance (PATEL *et al.*, 2020; ANJOS *et al.*, 2020; ROSA; VALENTIM, 2020; BI *et al.*, 2021; LEITE *et al.*, 2021) to discover the most appropriate way to improve products and services and integration for the public with disabilities, which shows that there is still a big fault for science and the market.

Due to the relevance of studies related to accessibility, as well as the lack of support favoring the adoption of approaches at a level of abstraction closer to user interfaces and interactions (which are subject to constant technological updates, such as frameworks and languages), this work proposes a model-driven approach that allows the design and generation of accessible applications from the beginning.

1.2. Objectives and Research Questions

Given the context, motivation, and research gaps discussed in the previous section, the research aim of this thesis is defined as:

> ***To develop a model-driven approach for generating accessible mobile applications that consider accessibility requirements early in the development process***.

To help us achieve the research aim, we set the following specific objectives:

- To review the main accessibility documents guidelines and recommendations;
- To integrate the main abstracted accessibility concepts on MDD process;
- To propose an extension of the IFML language to accommodate the modeling of accessibility requirements in the UI components based on the guidelines and recommendations found on the literature;
- To develop tools (e.g., model editor) and automatic code generators for testing the approach;
- To evaluate the proposed approach by developing a Proof of concept.

To guide the research process, we defined the Research Questions (RQs) that served as the basis for achieving the proposed objectives. The questions that guided the development of the research in the bibliographic databases were:

RQ1. *– How can we support the modeling of user interfaces and interactions in applications to create more accessible mobile systems using a Model-driven approach?*

With this question, we seek to find out what approaches, practices, and strategies are used in current work to model accessibility requirements (non-functional requirements) in the early stages of the development process (and not only in the testing phase). Furthermore, we investigate how user interfaces and interactions (with or without accessibility) are modeled in the context of a model-driven development process.

RQ2. *– How to abstract early on the model of user interface interaction flow to produce accessible applications?*

The purpose of this question was to identify research that addresses modeling in the context of model-driven development, requirements or standards, and standards for user interface development and user interactions. In addition, we investigated how researchers and developers deal with quality requirements (non-functional requirements) within model-driven development processes.

We defined these RQs to study accessibility problems to allow the abstraction of concepts related to general accessibility problems to appropriately classify and integrate the recommendations and solutions at each level of abstraction of the MDD process development strategy.

1.3. Research Design

In this section, we show the methodological process used to develop the studies. To understand the different types of research Wazlawick (2015) defines current research methods in computer science into three basic types: formal research, which requires the elaboration of a theory and formal proof of the correctness of that theory; empirical research, in which a new approach presented is compared with others through community-accepted tests; exploratory research, in which it is neither possible to prove a theory nor to present statistically accepted results, but case studies are here, qualitative analysis, and exploratory research in emerging areas where argumentation and persuasion are the main tools are appropriate. According to (GIL, 2009), research can be classified according to its objectives as follows:

Exploratory research aims at getting to know the problem better, making it clearer or hypothesizing. It involves bibliographical research, interviews with people who have had practical experience with the problem under study, and analysis of examples to aid understanding.

Descriptive research aims to describe the characteristics of a particular population and/or phenomenon and to identify the relationships between variables and determine the nature of those relationships.

Explanatory research focuses on identifying the factors that determine or contribute to the occurrence of phenomena and deepens knowledge about reality by explaining the reason, the cause of things.

Regarding the objectives, we can describe the research as exploratory research. We conducted the research through five activities, which are described below:

1. **Documentary research and literature review** - we seek to track the development of work done by researchers around the world, it is essential to sift through information in books, theses, articles, dissertations, journals, conference proceedings, and websites throughout the period of this study. In addition to this framework, the study of software documentation is related to the development of accessible applications. This activity is summarized in Chapter 3.

2. **Acquisition of knowledge** in the areas involved in the research - with the aim of deepening the concepts related to MDE, IFML, and accessible applications (including W3C guidelines and other recommendations) integrating the concepts involved. Web applications were initially selected as they represent typical examples of features used in the most common applications, and after mobile applications were included in the study. This activity is partially presented in Chapter 2.

3. **Investigate and test tools** that could be used in the research - considering the practices related to the development of parts of interactive applications (as mentioned in the previous activity) and the Model-driven development process, tools were investigated to support/-facilitate the implementation of the proposal and provided tools to support the modeling process. This activity is partially presented in Chapter 4.

4. **To define an architecture/approach** - this activity is central to meet the objectives of this thesis. It defines the model-driven approach for the development of accessible applications. It is the consolidation of knowledge previously acquired in the practices of the previous activities, where the concepts have been abstracted into the main components of the approach, and provide support for the concepts relevant for implement accessibility; transformers, code generators, derivations to the IFML standard to consider the accessibility and usability requirements. This activity is also presented in Chapter 4.

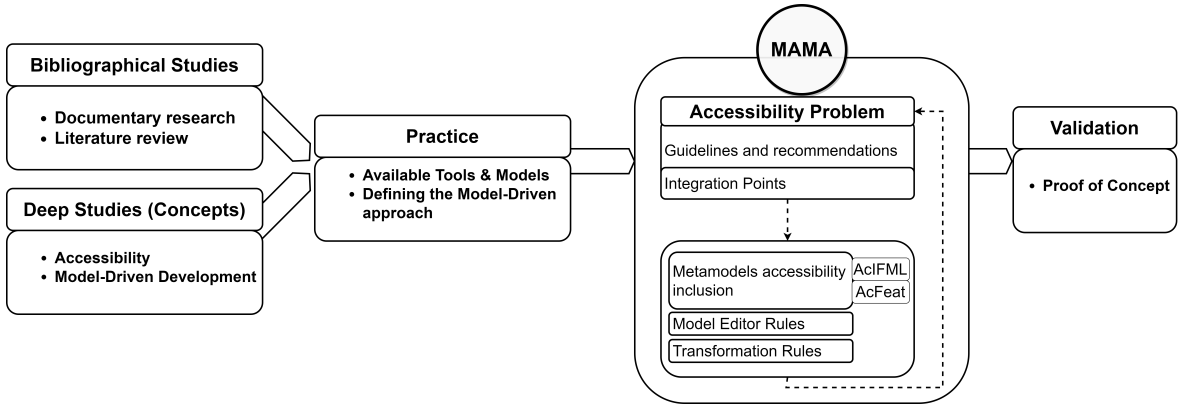
5. **Design, development, and testing of tools to support the developers on the approach** - to validate the approach, we proposed a set of components for the MDD process. Thus, the transformation tools were implemented, and the MDD process and the artifacts were considered on the proof of concept from the beginning to the end of the process. In this activity, we seek to validate the approach in order to obtain data to analyze its benefits, advantages, and disadvantages. This activity is presented in Chapter 5.

Figure 1 shows a diagram of the activities that summarize the proposal of this thesis and that were necessary to achieve the main objective. We conducted bibliographic studies through documentary research and literature review to understand the state of the art while deepening the areas of Accessibility and Model-Driven Development. For the practical implementation, we investigated the available tools and models and defined the architecture of the approach.

We sought to identify the most common accessibility problems on mobile devices that blind and visually impaired users face and then identify what guidelines, recommendations, and techniques we should use to solve the problem. We determined where to integrate each guideline into the architecture of MAMA. Once we identified the integration points, we abstracted what could be integrated or accommodated into the metamodels from the guidelines and recommendations.

We accommodated the abstractions through components and semantics in IFML to define the `AcIFML` extension. We also created a metamodel for the presentation abstraction metamodel, which we called `AcFeat`. We enriched the model editor to guide the modeler during the modeling process in terms of accessibility. And finally, we mapped the abstracted components to the specific platform components via transformation rules for generating a final accessible application. And we validated the approach using a proof of concept by modeling and generating an accessible application.

Figure 1 – The Design of this Research (accomplished by the 5 activities)



Source: Elaborated by the author.

1.4. Thesis Organization

The present work has been divided into six chapters. In this chapter, we provided the characterization of this thesis to give the reader a general overview of the context in which this research is inserted, the motivation to undertake in terms of research gaps, as well as the general and specific objectives and the research questions formulated and a description of the research method used. The remaining of this document is organized as follows:

- Chapter 2 presents the theoretical framework with the necessary directly related concepts to support the approach developed in this research
- Chapter 3 chapter provides an overview of the state-of-art and fundamental works for the construction of the approach proposed in this research
- Chapter 4 presents the Model-driven Approach Minding Accessibility approach
- Chapter 5 presents proof of Concept conducted to evaluate the approach proposed as well as to detach the main contributions of adopting it regarding not overloading developers with implementation of accessibility issues
- Finally, in Chapter 6, we revisit the work conducted in this thesis and provide the concluding remarks, the contributions and limitations, and future works regarding this thesis proposal. We also provide a list of publications that result from the research on this work.

CHAPTER

2

THEORETICAL FRAMEWORK

This chapter discusses the theoretical background necessary to understand this thesis. The awareness of these concepts was essential for the research development, ensuring alignment with the current scientific scenario. Section 2.1 overviews the basic theory behind software accessibility and some existing guidelines. Within Section 2.2, we will explore the Model-driven development approach. Subsection 2.3.1 presents a briefing of the basic concepts of language UML and Section 2.3 IFML language's central notions. The final section, Section 2.4, outlines the final remarks about the chapter.

2.1. Software Accessibility

According to ISO 9241-111: 2018 (ISO 9241-11, 2018), accessibility extent at which products, systems, services, environments and facilities can be used by people from a population with the widest range of user needs, features and capabilities to achieve identified objectives in identified contexts of use. The Brazilian Inclusion Law (No. 13.146) considers accessibility as the possibility and condition of reach for the safe and autonomous use of spaces, furniture, urban equipment, buildings, transport, information and communication, including their systems and technologies, as well as other services and facilities open to the public, for public use or private for collective use, both in urban and rural areas, by people with disabilities or reduced mobility (CIVIL, 2015).

The Americans with Disabilities Act (ADA) defines disability as a physical or mental impairment that substantially limits one or more major life activities, a person who has a history or record of such an impairment, or a person who is perceived by others as having such an impairment (COOK, 1991). In general, disabilities can be divided into several broad subcategories, which include the 8 (eight) main types of disabilities: *(1.)* mobility/physical (upper limb(s) disability, lower limb(s) disability, manual dexterity disability, coordination disability with different organs of the body); *(2.)* Spinal Cord (SCI) (paraplegia, quadriplegia); *(3.)* Head

Injuries (TBI) (acquired or traumatic); *(4.)* vision (blindness, low vision, color blindness); *(5.)* hearing (deafness and hearing loss); *(6.)* cognitive/learning (dyslexia, Attention Deficit Hyperactivity Disorder (ADHD), brain injury, Down syndrome, autism, and dementia); *(7.)* psychological (Personality Disorders, schizophrenia); *(8.)* invisible (chronic pain, chronic fatigue) (WORLD, 2021), which may be present from birth, permanent, temporary, situational or acquired over time, as some that come with age.

Microsoft (2016) assumes that disability happens at the points of interaction between people and society. There are three types of disabilities: permanent, temporary, and situational that someone may have related to our four senses of touch, sight, hear, and speak (World Health Organization *et al.*, 2001; MICROSOFT, 2016; O'NEILL, 2021). Based on the literature review, we grouped the senses with another disability category: cognition.

Over the years, Assistive Technology (AT) has emerged among the various components that can help users with disabilities interact with the world, which are resources that help people with disabilities live more independently and self-reliant lives and improve their quality of life. The United States (US) Section 508 defines AT as any item, piece of equipment, or system (whether acquired commercially, modified, or customized) that is commonly used to increase, maintain, or improve functional capabilities of individuals with disabilities (GSA, 2020a). There are several types of AT, such as screen readers, refreshable Braille displays, video magnifiers, cochlear implants, Video Relay Service (VRS), Telecommunication Device for the Deaf (TDD), specialized mice and keyboards, pointing devices (head pointers, mouth sticks), sip and push switches, voice recognition software, eye tracking, trackballs, and others.

For each type of disability, there are many functional challenges in everyday activities, and using digital systems, which present many barriers for people with disabilities, is no different. Tools and technologies must be designed and developed in a way that people with disabilities should be able to use, perceive, understand, navigate, interact, and contribute (W3C, 2019). However, it can be an arduous and challenging goal to achieve (MIÑÓN *et al.*, 2014; KAWAS; VONESSEN; KO, 2019). Still, because of the contingent of people with disabilities, content creators and developers should be aware of factors that can make their content or products more accessible. However, for these technologies to achieve their goals, systems must be prepared for accessibility.

Accessibility is an attribute of product quality (content and services), ensuring that people with disabilities perceive, understand, navigate, and interact with the Web and contribute to it. Other target groups should also benefit, including older people whose abilities change with age (W3C, 2019). It is essential for developers and organizations that want to create high-quality software (W3C, 2019). Different organizations have developed various standards, guidelines, recommendations, strategies, and resources to support incorporating accessibility features and requirements into the development process to assist developers and organizations.

Among the organizations, the World Wide Web Consortium (W3C) is committed to lead-

ing the Web to its full potential and promoting a high level of usability for people with disabilities. To this end, the Web Accessibility Initiative (WAI) launched (W3C, 2021a) to organize strategies, standards, and supporting materials to help the software development community understand and implement accessibility, specifications that serve as a foundation for all platforms, such as:

- *Web Content Accessibility Guidelines (WCAG)* - explain how to make content accessible and available to all users, especially those with disabilities;
- *User Agent Accessibility Guidelines (UAAG)* - document aimed at developing user agents that can reduce barriers to accessibility on the web for people with disabilities;
- *Authoring Tool Accessibility Guidelines (ATAG)* - which aims to assist developers in the implementation of accessible web content authoring tools and creation of accessible authoring tools;
- *Accessible Rich Internet Applications (WAI-ARIA)* - aims to make applications that include dynamic content and advanced user interface controls accessible, generally developed with Ajax, Javascript, and related technologies.

2.1.1. Accessibility Guidelines

The most relevant, widely accepted, and well-known accessibility documents are the WCAG guidelines (W3C, 2021c). Version 1.0 of the WCAG was first published in 1999. This version was updated to 2.0 and published in December 2008. WCAG 2.0 is internationally recognized as the benchmark for web accessibility and is referenced in the regulations of several countries. In June 2018, WCAG 2.1 was published, updating WCAG 2.0 and extending the W3C accessibility guidelines while maintaining the W3C standard for implementable, technology-neutral, objectively testable, and universally applicable accessibility guidelines (W3C, 2018a) and bringing 1 (one) new guideline and 17 (seventeen) new success criteria.

WCAG 2.1 has been widely supported by industry, including accessibility-focused companies, the disability community, research, education, and government (W3C, 2018a). WCAG 2.0 contains 12 guidelines and WCAG 2.1 contains 13 guidelines, both organized under four principles (W3C, 2021c), acronym POUR that stands for:

- **Perceivable:** users must be able to process the content by seeing, hearing, or touching it
- **Operable:** content must be operated with both a keyboard and a mouse, but also through other alternative input devices
- **Understandable:** software must present content in a consistent, predictable, readable form
- **Robust:** content adopts standards that allow it to function or be understood across a wide range of technologies and will continue to operate into the future as technologies evolve.

When creating or evaluating a product with accessibility in mind, product owners, developers, designers, or content providers need to understand the principles of POUR to ensure an accessible user experience. WCAG 2.1 documents 13 (thirteen) guidelines whose relationship to the principles, and descriptions are summarized at Table 1. This table shows a total number of Success Criteria that should be verified for each Principle. In fact, for each guideline, there are testable Success Criteria, which are in three levels of compliance:

(a.) Level **A** with 30 (thirty) success criteria (the most basic web accessibility features);
(b.) Level **AA** with 20 (twenty) success criteria (the most significant and common barriers for disabled users); and
(c.) Level **AAA** with 28 (twenty-eight) success criteria (the highest level of web accessibility).

Further, there are many techniques associated with each success criteria (W3C, 2021c), so achieving at least an acceptable level of accessibility requires meeting all success criteria at levels A and AA.

Table 1 – The 13 Guidelines in WCAG 2.1

Principle	**Guidelines Description**	**Number of Success Criteria**
Perceivable	**1.1.** Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.	29
	1.2. Provide alternatives for time-based media.	
	1.3. Create content that can be presented in different ways (for example simpler layout) without losing information or structure.	
	1.4. Make it easier for users to see and hear content including separating foreground from background.	
Operable	**2.1.** Make all functionality available from a keyboard.	29
	2.2. Provide users enough time to read and use content.	
	2.3. Do not design content in a way that is known to cause seizures or physical reactions.	
	2.4. Provide ways to help users navigate, find content, and determine where they are.	
	2.5. Make it easier for users to operate functionality through various inputs beyond keyboard.	
Understandable	**3.1.** Make text content readable and understandable.	17
	3.2. Make Web pages appear and operate in predictable ways.	
	3.3. Help users avoid and correct mistakes.	
Robust	**4.1.** Maximize compatibility with current and future user agents, including assistive technologies.	3

Source: Adapted from W3C (2018b)

Due to Web universality, software accessibility studies generally refer to web page accessibility based on WCAG recommendations (ORDOÑEZ; HILERA; CUEVA, 2020). In the W3C WAI, for example, mobile accessibility is covered in the existing accessibility standards/guidelines, so there are no separate mobile accessibility guidelines (W3C, 2021b). However, WCAG 2.1 provides additional success criteria that specifically address mobile platforms.

In addition, accessibility has become a legal requirement in several countries, establishing responsibility for access to content on the Web (SKJERVE; GIANNOUMIS; NASEEM, 2016; XIONG; FARENC; WINCKLER, 2007; MORENO; MARTINEZ, 2019). The most famous regulation is Section 508 of the United States of America, which specifies that "agencies must give disabled employees and members of the public access to information comparable to the access available to others" (GSA, 2020b). Section 508 is aligned with WCAG and incorporated WCAG 2.0 in 2017.

There are some regulations and recommendations in other countries as well, such as Brazil, Japan, France, Canada, Italy, Spain, Australia, Chile, the Netherlands, and South Korea. There are also other regulatory documents with accessibility considerations as for example ISO 9241 (ISO, 2008; ISO 9241-11, 2018), ISO/IEC Guide 71 (ISO, 2014b), ISO/IEC TR 29138 (ISO, 2018), ISO/IEC 24751 (ISO, 2008), ISO 14289 (ISO, 2014a).

Additional documentation is available to help developers and content providers with accessibility support, such as operational systems accessibility Application Programming Interface (API)s like Android (Google LLC, 2021a), IOS (Apple Inc., 2021), Microsoft (MICROSOFT, 2021b; MICROSOFT, 2021a; MICROSOFT, 2016; MICROSOFT, 2018). The community also has well-structured documents to raise developer awareness of accessibility implementation, such as Web Accessibility in Mind (WebAIM) (WEBAIM, 2021), British Broadcasting Corporation (BBC) Accessibility Guideline Barrier Walkthrough Guide (BBC, 2020), IBM accessibility(IBM, 2021), IMS Access for All (INC., 2021), GuAMA - Guide to the Development of Accessible Mobile Applications (SIDI, 2019; ANJOS *et al.*, 2020), and A11Y accessibility guide (A11Y, 2021). All of these initiatives reinforce the importance of having accessible products available to all, and were also used to support our approach.

Large technology companies often have internal regulations mandating accessibility for their products and services (SNIDER; II; TREWIN, 2020). However, professionals still have little awareness of accessibility or knowledge of compliance requirements or guidelines (FREIRE; RUSSO; FORTES, 2008; DUARTE *et al.*, 2016; RIEGER *et al.*, 2020; LAZAR, 2019; PATEL *et al.*, 2020). Rather than considering accessibility from the beginning of a software development project, a common approach is to consider it after the artifact is nearly complete (ISO, 2008; SILVA; GONçALVES; PEREIRA, 2017; RIEGER *et al.*, 2020), and modify it to incorporate accessibility that was initially overlooked. This approach negatively impacts the final product by increasing costs, requiring more time, making it challenging to incorporate accessibility, and causing issues such as product design inconsistency.

An attractive alternative that could address the current challenges of accessible applications is the MDD approach, which, in addition to the benefits of the approach, has attracted the attention of the accessible software development community because of its ability to generate code from models (ORDOÑEZ; HILERA; CUEVA, 2020).

2.2. Model-Driven Development

In software engineering, a model is an abstraction that is not limited to a particular object or phenomenon. It is associated with a class (LUDEWIG, 2003), but with concepts that are much less bound to the underlying implementation technology and thus abstracting from unnecessary details (AMELLER *et al.*, 2021). Models can be an abstraction of a system or thing used to describe a representation, generally in miniature, to show the structure or appearance of something for specific purposes. Therefore, it allows focusing on different aspects that are of interest to stakeholders (SCHMIDT, 2006a). Traditionally, it can refer to an artifact formulated in a modeling language, such as UML, that describes a system using various diagram types (KÜHNE, 2005).

When developing software applications, you can choose from a variety of methods. Most current development processes are still code-centric, meaning programmers write the code manually. One of the disadvantages of code-centric development is the heavy workload for programmers, which leads to a time-consuming development process and difficulties in maintaining and evolving software products. And some problems for the end-user, such as significantly different design solutions for the exact specifications of different platforms. As software systems become more complex and often need to be deployed in different environments, devices, and platforms, an alternative is to use model-driven approaches.

In this context emerged MDE, which is a software development paradigm that relies on models and model transformations to develop, maintain, and evolve software, using high-level abstractions that capture key features of a domain (MELLOR *et al.*, 2004). MDE includes model-based tasks of a complete software engineering process, concentrating on creating models or abstractions closer to domain concepts.

MDD is a subset of MDE and considers models and model transformations as the center of software analysis, design, and implementation of software engineering processes (SCHMIDT, 2006a; BÉZIVIN, 2006). A relevant aspect is an emphasis on bridges between technological spaces and the integration of knowledge bases developed by different research communities (FAVRE, 2004). Among the main advantages of using MDD approaches, the literature cites the following factors (MELLOR *et al.*, 2004; LUCRÉDIO, 2009):

(a.) **Increasing of productivity:** repetitive tasks can be implemented in the transformations; thus, development time is optimized;

(b.) **Avoidance of repetition:** with the automation of transformations, the same model can be transformed into code for different platforms and devices (portability), avoiding repetitive tasks (reuse);

(c.) **Maintainability and documentation:** since models are part of the software and changes are made directly to them, consistency between code and documentation is maintained, making maintainability easier;

(d.) **Easy of communication:** the approach facilitates communication between the team and *stakeholders* by providing a common vocabulary that is used throughout the project;
(e.) **Improvement of quality:** thanks to models and transformations, the occurrence of semantic and accidental errors is lower, which favors more efficient implementations.

On the other hand, MDD approaches have disadvantages that should be considered, such as those cited by Lucrédio (2009):

(a.) **High initial investment:** MDD requires building a model-oriented reuse infrastructure, which takes more time and effort and requires a higher initial investment;
(b.) **Complexity:** the modeling tools, transformations, and code generators increase the complexity of the project because they are more difficult artifacts to build and maintain;
(c.) **Learning curve:** it is necessary to learn new processes and tools, which requires additional training of professionals involved in the project;
(d.) **Performance curve:** generally, code generators end up containing a larger amount of code, so that the result may have lower performance compared to handwritten code;
(e.) **Rigidity:** since most of the code is generated, the final software is less flexible, which is beyond the developer's control.

MDD has made incredible contributions to leveraging abstraction and automation in almost every software and systems development area, and analysis (BUCCHIARONE *et al.*, 2020). MDA, as well as Microsoft Software Factories (GREENFIELD; SHORT, 2003), and the Eclipse Modeling Framework (EMF) of the Eclipse Project (STEINBERG *et al.*, 2008), are examples of the MDD approach.

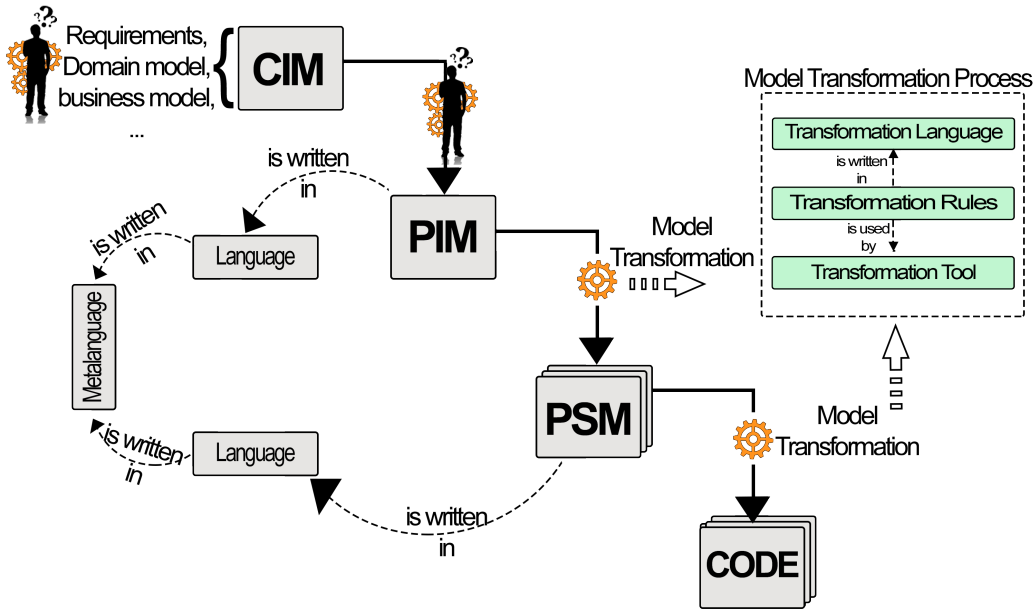
MDA is a branch of software engineering spearheaded by the OMG, which proposes an architecture that considers all project phases. It provides an approach for deriving value from models and architecture to support the entire life cycle of physical, organizational, and systems (SIEGEL, 2014).

Figure 2 illustrates the MDA framework. It ranges from Computational Independent Model (CIM) of requirements and then Platform Independent Model (PIM) analysis and design, to Platform Specific Model (PSM) and final code. Models are described by languages, written in a metalanguage, which is a model (metamodel) that defines a language for expressing a model (SIEGEL, 2014).

In the MDA framework, transformation rules describe how to transform one or more constructs in the source language to one or more constructs in the target language. Accordingly, a transformation tool performs the transformation from a source PIM to a target PSM. A second (or the same) transformation tool transforms the PSM into code.

In this work, we have considered the two OMG languages: IFML and UML (a prerequisite for the IFML model), as key elements for the design, transformation of the models, and generation of the final application, with emphasis on the interfaces.

Figure 2 – Overview of the MDA framework



Source: Elaborated by the author.

2.3. Interaction Flow Modeling Language

IFML is a modeling language that arose from the need to supplement the lack of specific diagrams for modeling interfaces to express the content, user interaction, and behavioral control of software front- ends (OMG, 2015b). The language proposal is to standardize the modeling of user interfaces in applications and express the content, navigation paths, user events, and interactions displayed on these interfaces, as well as to connect to the business logic and persistence layer of application front-ends from different domains (OMG, 2015a).

The OMG adopted it as a standard for modeling user interfaces in applications in 2013, but the 1.0 specification was not released until March 2015. The language was inspired by WebML (CERI; FRATERNALI; BONGIO, 2000), and WebRatio (ACERBIS *et al.*, 2008), and its specification consists of four artifacts, each specified according to OMG standards:

(I.) *The IFML metamodel* - specifies the structure and associations between the elements defined by the Meta Object Facility (MOF) metamodeling language

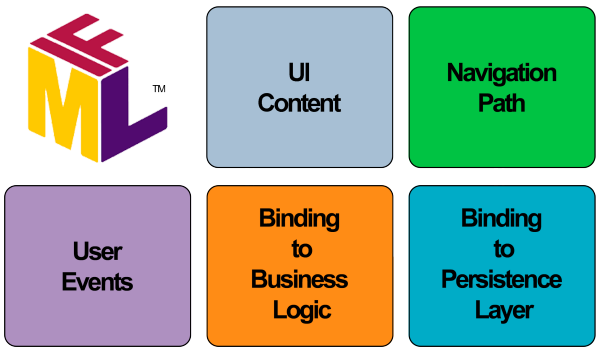
(II.) *The IFML UML profile* - it uses UML concepts to design and extend class diagrams, state machines, and elements - defined in conformance with the UML 2.4 profile policy

(III.) *The IFML visual syntax* - it has graphical notations for representing elements and models defined by OMG standards Diagram Definition (DD) and Diagram Interchange (DI)

(IV.) *The IFML XMI* - provides the IFML model exchange format, for tool portability.

The objectives include five dimensions, shown in Figure 3, using UI to express the content displayed on these interfaces; navigation paths to model the flow of the application; user events and interaction to model how users can interact with the system; and the connection with the business logic and persistence layer of application front-ends from different domains (OMG, 2015a). The focus is on the user interface and interactions, and it is designed to exchange and reference external models (BRAMBILLA; FRATERNALI, 2014a).

Figure 3 – Objectives of the IFML language



Source: Adapted from Brambilla (2015)

IFML language supports the specification of the front-end of applications independently of the technological details of their realization (BRAMBILLA; FRATERNALI, 2014a) and brings several benefits to the development process of application front-ends (OMG, 2015a):

- *Formal specification* of the different perspectives of the front-end: content, interface composition, interaction and navigation options, and connection with the business logic and the presentation
- *Isolation of UI* from front-end platform implementation-specific issues
- *Separation of concerns* in the user interaction design, thus granting maximum efficiency to all the different developer roles
- *Communication of interface and interaction design* to non-technical stakeholders, permitting validation of requirements from subject matter experts and clients sooner in the development process
- *Automatic generation* of code is also possible for the application front-end

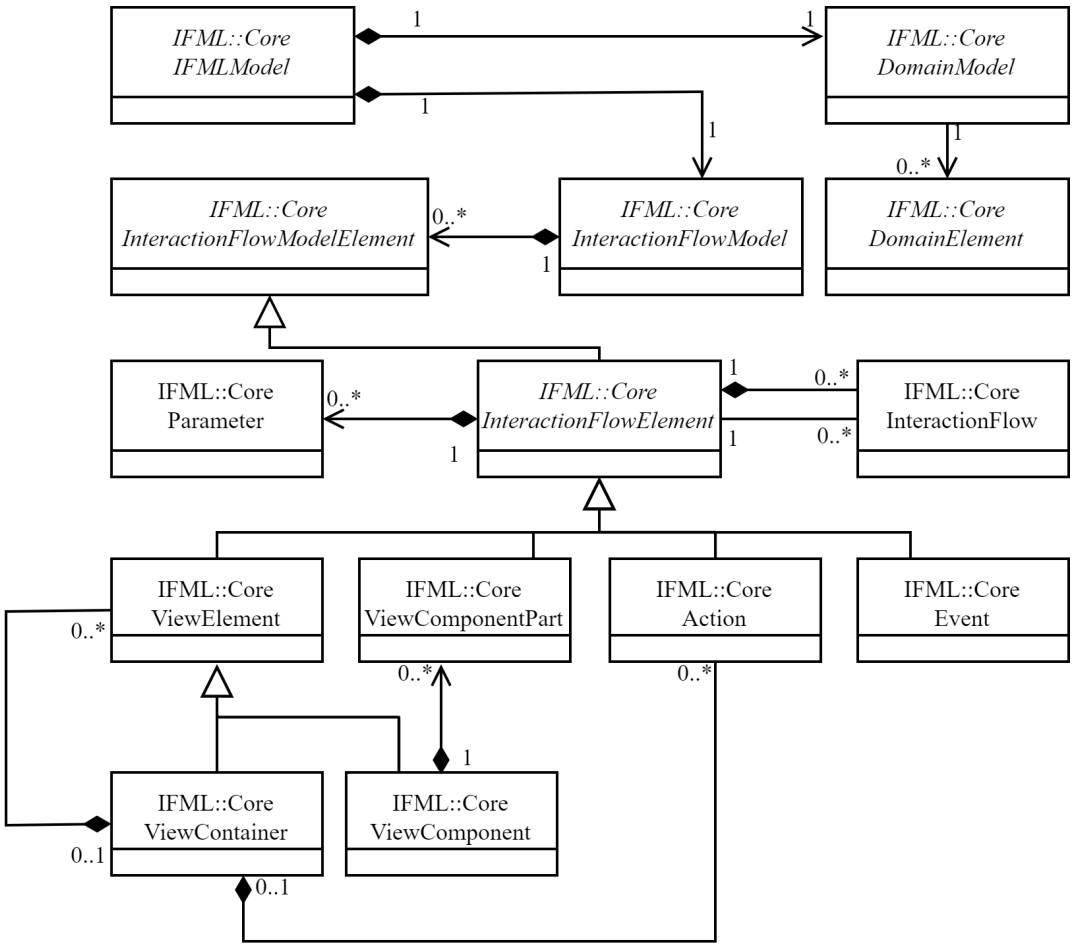
The technical foundations of the IFML language are based on the OMG MDA framework (BRAMBILLA; FRATERNALI, 2014a) and ensure seamless integration with the OMG specifications of the other layers of the software system, interoperating perfectly with UML, SysML,

SoaML, and BPMN (OMG, 2015b). The platform-independent IFML metamodel provides a description of user interfaces with a set of concepts to characterize the essential aspects of the user's interaction with a software application interface (OMG, 2015b).

The implementation of IFML 1.0 is organized into three packages: *IFML Core Package* contains all concepts for creating the interaction infrastructure, *IFML Extension package* contains concepts that extend the Core package, and *IFML Data type Package* contains data types determined by the UML metamodel and specializes some UML meta-classes as origins for IFML meta-classes.

We can see a fragment of the IFML metamodel in the Figure 4 with the core concepts of the language. The root element is the concept *IFMLModel*, which contains two main concepts: the *DomainModel*, to which the concepts associated with the domain language are mapped, and the *InteractionFlowModel*, which contains the concepts of user interfaces and interactions. We examined the derived meta-classes of "*InteractionFlowModel*" in detail, as they were the target of our proposal since they represent the interactive objects rendered in the screen interfaces.

Figure 4 – Fragment of IFML metamodel - Core Concepts



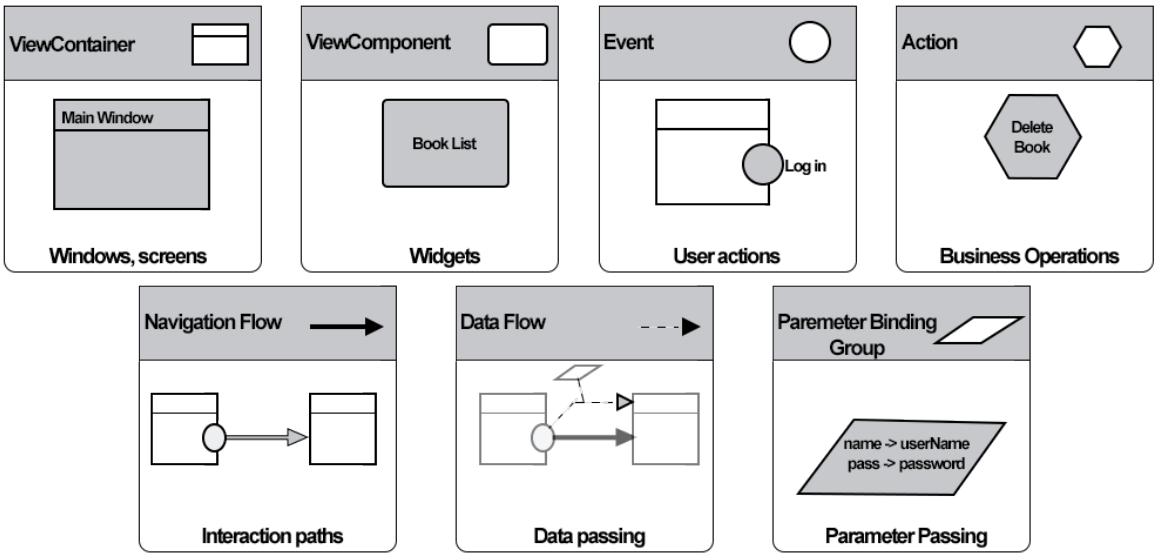
Source: Adapted from OMG (2015a)

IFML core concepts allow developers to specify the organization of the interface. Consequently, an IFML model supports the following design perspectives of front-end modeling (BRAMBILLA; FRATERNALI, 2014a):

- *Composition of the view* (`Core::ViewContaier`) – defining what the visualization units of the UI are, how they are organized, and which ones are displayed simultaneously
- *Content of the view* (`Core::ViewComponent` and `Core::ViewComponentPart`) – consisting of the definition of the content elements of the view, which are displayed by the application to the user, and which inputs are captured by the user and delivered to the application, consisting of the definition of the view components, i.e., the content and data input elements contained in view containers
- *Events* (`Core::Event`) – defining the interaction events supported by the application that can affect the user interface state.
- *Actions* (`Core::Action`) – covering the business operations triggered by the events
- *Effects of interaction* (`Core::InteractionFlow`) – modeling the effects of events and action execution on the state of the user interface through Interaction Flows
- *Parameter binding* (`Core::Parameter`) – defining the input-output dependencies between the view elements of the user interface or between view elements and the triggered actions.

To achieve these design perspectives, the language introduces the basic IFML building blocks with a visual syntax shown in Figure 5. For the UI, the ViewContainer building block is used to model the overall view structure that organizes the application's windows or screens, which is designed by one or more top-level containers and each with its nesting relationships and sub-containers. The base of the widgets is the ViewComponent block, which expresses the content of the view (content display and data input). The Event block defines the events that affect the state of the user interface (they can be generated by the interaction of a user, by the application itself or by an external system).

Figure 5 – IFML building blocks



Source: Elaborated by the author.

Another important IFML modeling concept for this thesis approach is **Context**, a runtime aspect of the system that determines how the user interface should be configured and what content it may display (OMG, 2015a). **Context** has a special meaning for accessible applications to meet specific user interface accessibility requirements. An IFML **Context** has several dimensions called ContextDimensions, which have UserRole, Device, and Position as predefined specializations in the IFML 1.0 specification.

If the user context satisfies all ContextDimensions, access is granted to the ViewElements of the ViewPoint and to the events that can be triggered on them. The ViewPoint is a reference to an interrelated set of InteractionFlowModelElements to facilitate understanding of a complex system, to allow or disallow access to the system by a specific UserRole, or to adapt the system to a specific context change (OMG, 2015a).

The requirements for a Context to be active are expressed by Object Constraint Language (OCL) [1] expressions called ActivationExpressions. OCL is a formal language that modelers can use to specify application-specific operations/actions programming language independently, that when executed, do alter the state of the system. The predefined Context and ContextDimensions elements on the IFML language can be extended to represent finer-grained or other context perspectives, e.g., network connectivity or temporal aspects (BRAMBILLA; FRATERNALI, 2014a), and with the concepts extended on our proposed extension, accessibility context dimensions.

The prerequisite for creating an IFML model is to have at least one diagram model that can represent the minimal structural aspects of a system to describe the domain. It is necessary

[1] https://www.omg.org/spec/OCL/

for the back-end to be loaded (although this is not our main focus) at least the structural features, such as the features of a UML class diagram. Many languages and guidelines have been proposed for domain modeling, which are now consolidated. For this reason, IFML does not propose another language for domain modeling, but instead uses the class diagrams of the UML (OMG, 2015a). The motivation for this choice is that IFML is an OMG standard based on UML, and thus the use of class diagrams results in a notation that is suitable for both domain and front-end modeling(OMG, 2015b).

2.3.1. UML and the IFML model

UML is a modeling language for elaborating the structure of software projects, commonly used for visualizing, specifying, constructing, and documenting artifacts of complex software systems (BOOCH; RUMBAUGH; JACOBSON, 2005). Its structure consists of a four-layer metamodel architecture whose objective is to present an overall infrastructure with the distribution of concepts at multiple levels of abstraction suitable for defining the semantics associated with complex models (BOOCH; RUMBAUGH; JACOBSON, 2005).

UML provides a set of diagrams and elements for modeling computer systems. It also defines extensions and specializations, exposing a flexible and adaptable character without loss of formalization (BOOCH; RUMBAUGH; JACOBSON, 2005). Many OMG specifications are defined in UML, making the OMG's standard modeling language a foundation of MDA (although it is not a requirement - MOF[2] is the mandatory modeling foundation for MDA (SIEGEL, 2014).

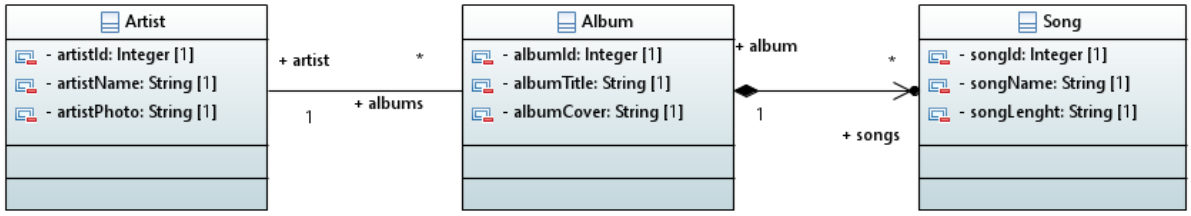
The OMG currently promotes the use of a subset of UML called Foundational UML (fUML)[3]. It is an executable subset of standard UML used to define, in an operational style, the structural and behavioral semantics of systems (OMG, 2020), which diagrams correspond to similar diagrams in the UML 2 superstructure specification. The fUML language provides concepts for defining classes with attributes and operations, abstract classes, multiple inheritances, enumerations, as well as an extensible type system (OMG, 2020). The foundational subset defines a basic virtual machine for UML and the specific abstractions supported thereon, enabling compliant models to be transformed into various executable forms for verification, integration, and deployment(OMG, 2020).

As mentioned earlier, at the beginning of building an IFML model, we need at least one class diagram to define the structural concepts of the application domain (other OMG language diagrams can be added). This class diagram will be used on the first transformation, mapping the UML domain model to IFML DomainModel. A class diagram is a static UML structure diagram that describes the structure of a system by representing the classes of the system, their attributes, operations (or methods), and the relationships among the objects (BOOCH;

[2] MOF - <https://www.omg.org/mof/index.htm>
[3] fUML - <https://www.omg.org/spec/FUML/About-FUML/>

RUMBAUGH; JACOBSON, 2005). Figure 6 shows an example of a class diagram with the relations and attributes.

Figure 6 – UML Simple Class Diagram Example



Source: Elaborated by the author.

In Figure 6, the main concepts of a system are represented in the three UML classes: Artist, Album and Song. There are many basic meanings of relationships among these classes that are also represented in the Diagram. Just to illustrate part of these semantics, it is possible to interpret it in such a way that there is an instance of Album described by the following attributes: albumId, albumTitle and albumCover, and is associated with at least one instance of Artist. On the other hand, an Artist is an object of the class Artist even there is no Album. In fact, all the symbols in the Diagram are well defined and express specific meaning in a summarized way, then Figure 6 presents other relevant concepts in terms of attributes and relations among these three UML classes.

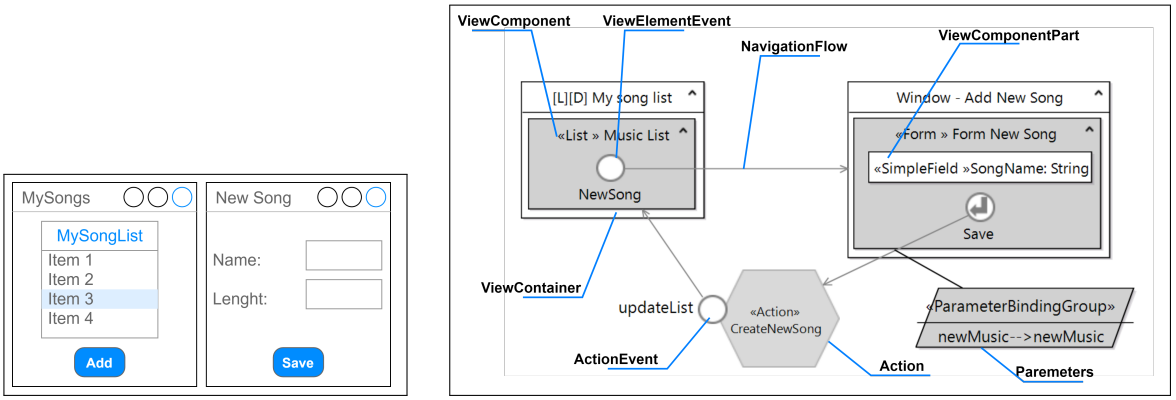
2.3.2. IFML model

Figure 7 shows a fragment of an example of one IFML model for a simple user song list application using the domain model Figure 6. Figure 7a shows a simple wireframe UI with a list of songs that the user can interact with through a button click to add a new song, open a form, and register new music. In Figure 7b we can see the modeled user interface and interaction with IFML using the language concepts such as:

- a ViewContainer (My song List)
- a ViewComponent (Music List)
- a ViewElementEvent (NewSong)
- a NavigationFlow, which represents an available interaction path
- ViewComponentPart (SimpleField SongName)
- the Action, and
- the ActionEvent (updateList).

The IFML metamodel specifies the structure and semantics of IFML constructs. Figure 8 shows a model tree Figure 8a for the UI defined in Figure 7 with the modeled structure and hierarchy among the UI components and the user interactions. Figure 8b illustrates the semantics of components using a SimpleField with its semantic attributes.

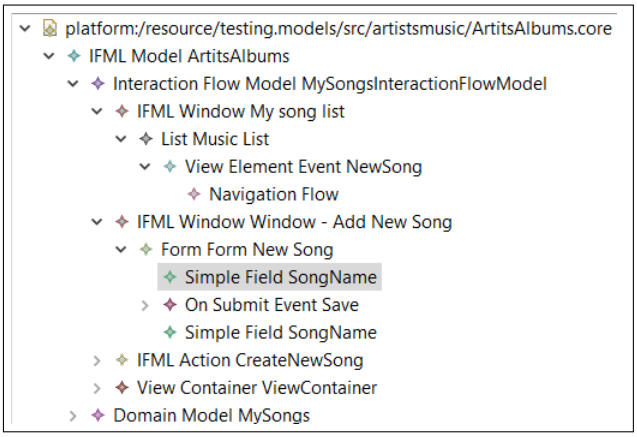
Figure 7 – IFML model example

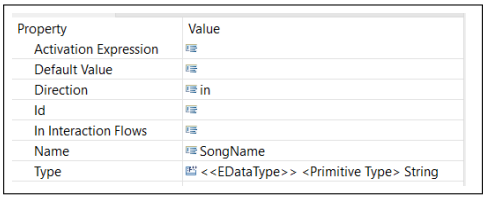


(a) Simple Wireframe Example (b) IFML model example

Source: Elaborated by the author.

Figure 8 – IFML Model Tree and SimpleField Semantics





(a) My Songs IFML model tree (b) SimpleField SongName Semantics

Source: Elaborated by the author.

2.4. Final Remarks

This chapter has covered the theoretical foundations necessary for the development of this thesis. We presented concepts related to software accessibility, its context, and reference guidelines. Then, we gave an overview of MDD and its main pillars. Subsequently, we have discussed the IFML language and its main concepts. Next chapter, we present how we conducted the bibliographic review of the state-of-art. The selected works are directly related to the solution presented in this thesis, with studies that have addressed accessibility or usability early in the development process and model-driven approaches to generate the UI and some strategies for the inclusion of accessibility or usability on MDD processes.

CHAPTER

3

LITERATURE REVIEW

A variety of research influences this work. This chapter shows how we conducted the bibliographic review of the state-of-art to select the works that have a direct bearing on the solution proposed in Chapter 4. The chapter begins with Section 3.1, which addresses the review protocol used at the beginning of the research process and how the dataset was updated during the research. Related work follows, which has been divided into two sections: Section 3.2, work that addresses accessibility and usability requirements in the development process; Section 3.3, studies that generate the UI based MDD approaches, works that used the IFML language, and works that address the accessibility/usability requirements using model-driven approach to generate the UI.

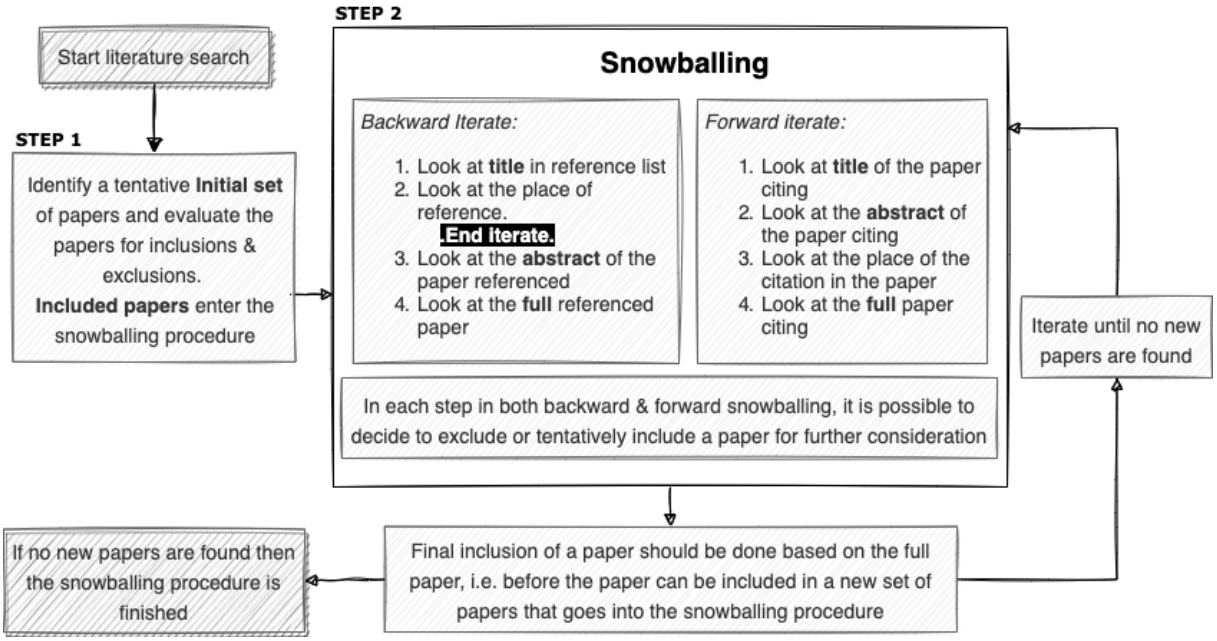
3.1. Review Methodology

A literature review protocol defines the methods to be used to conduct a particular literature review. According to Wohlin (2014), systematic literature studies, including reviews and maps, have emerged to synthesize evidences and allow researchers to come to a common understanding of the status of a research area in software engineering over the past decade. The Snowballing protocol (WOHLIN, 2014), presented in Figure 9, was used to guide and initiate the process of exploring the state-of-art of research on the topics of interest.

According to Wohlin (2014), the first step in the Snowballing procedure is to formulate search terms to define an initial set of papers to which the approach can be applied. The definition of this initial set of papers is considered the first challenge and must include certain specific characteristics, such as:

- Relevant articles from various scientific communities;
- Not so small number of articles;

Figure 9 – Snowballing overview



Source: Adapted from Wohlin (2014)

- If many articles are found, then identifying relevant articles with many citations may be an alternative;
- The initial set must have diversity, i.e., cover different editors, authors, years;
- The initial set must be formulated using keywords in the search query, considering synonyms.

The second step consists of the iterations: forward and backward. Forward iterations consist of analyzing works that cite the article under study forward, while backward iterations consist of analyzing article references. After performing the iterations, new papers must be identified and added to the stack for a new iteration. It should be noted that only articles found from the "included" articles should be used in the analysis, following the planned inclusion and exclusion criteria.

For the generation of the initial set, the planning phases were performed, following the procedures recommended in the systematic mapping (KITCHENHAM; CHARTERS, 2007). The Parsifal[1] tool was used to support some of the review procedures. Parsifal is an *online* tool designed to support researchers to carry out systematic literature reviews within the context of Software Engineering (LTD., 2014).

[1] Parsifal - <https://parsif.al/>

3.1.1. State-of-art Questions

After defining the theme and the need to conduct the research, we defined the questions that served as the basis for the state-of-the-art research process in topics related to the research object. The questions that guided the development of the research in the bibliographic databases were:

Q1. – *How have accessibility and usability requirements been inserted into development processes?*

With this question, we aim to identify what approaches, practices, and strategies are used in current work to model accessibility requirements (non-functional requirements) in the early stages of the development process, what recommendations are made at each stage of the process, and where these requirements are incorporated.

Q2. – *How do model-driven approaches deal with the generation of UI and user interaction?*

This question objective was to investigate how user interfaces and interactions (with or without accessibility) are modeled as part of a model-driven development process.

Q3. – *How is the IFML language used by researchers on the MDD context?*

With this question, we wanted to retrieve the works that are using the IFML language and how researchers included the language on the MDD processes.

Q4. – *How do model-driven approaches handle the generation of UI including non-functional requirements such as accessibility?*

The aim of this question was to investigate how researchers and developers deal with quality requirements (non-functional requirements) within model-driven development processes.

3.1.2. Start Set Definition

To check whether similar works already exist on the topics under investigation and to select the primary works to start the review, a search was performed starting from the journal portal CAPES[2]. This portal is a virtual library that provides Brazilian teaching and research institutions access to the best international scientific production.

We used the following search strings in the *IEEExplore Digital* Library[3], *ACM Digital* Library[4], *Springer* Link[5], and *ScienceDirect*[6] databases:

[2] <https://www-periodicos-capes-gov-br.ezl.periodicos.capes.gov.br/>
[3] <https://ieeexplore.ieee.org/>
[4] <https://dl.acm.org/>
[5] <https://link.springer.com/>
[6] <https://www.sciencedirect.com/>

- `((Accessibility OR Usability) AND ("Model Driven Development" OR MDD) AND ((User AND (Interaction OR Interfaces)) OR (IFML)) AND NOT (Tests OR Evaluation OR Adaptation OR Validation))`
- `("Accessibility Requirements" AND ("Model Driven Development" OR "Software Engineering" OR Architecture OR Process))`

For the search, the syntax of each bibliographic database, keywords, synonyms and terms according to Table 2 were considered.

Table 2 – Keywords, synonyms, and equivalent terms used on the search

Keyword	**Synonym/Equivalent Term**
`Accessibility`	*Usability*, *inclusive design*, *universal design*
`Model Driven Development`	*Model-driven*, MDD, MDE, MDA, DSL, *Domain Specific Language*
`User Interface`	MDDUI, *Front-end*, *Graphical User Interface*, GUI, *User Interface Patterns*
`Interaction Flow Modeling`	IFML, *User Interaction*
`Accessibility Requirements`	*Usability Requirements*, *Non-functional Requirements* *Quality Requirements* *Accessibility Patterns*, *Usability Patterns*

Source: Elaborated by the author.

The initial search was conducted in 2017, limiting search interval to the last 5 (five) years. After the initial search, we acquired 1.437 documents: 93 papers from the *ACM Digital* Library dataset, 21 from *IEEExplore Digital* Library, 893 from *Springer* Link, and 151 documents from *ScienceDirect*. We then applied the inclusion and exclusion criteria described as follows.

Inclusion and Exclusion Criteria

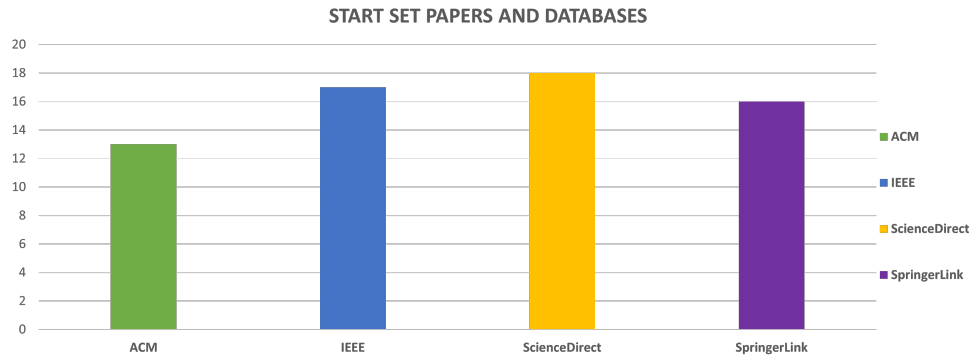
- Inclusion Criteria:
  1. Papers that present primarily or secondarily model-driven approaches aimed at modeling and generating user interfaces;
  2. Studies that primarily or secondarily present approaches that use the language IFML (Interaction Flow Modeling Language);
  3. Studies that consider the use of accessibility requirements in the early stages of the development process;
- Exclusion Criteria:
  1. Duplicated articles (equal versions in more than one search source);
  2. Outlined versions of the same study, whose only the had only their most recent or complete version was included unless there was different information;

3. Articles that are not in the domain of Computer Science;
4. Studies that do not focus on the scope of the research;
5. Articles that focus on usability/accessibility evaluation, testing, or interface adaptation and re-adaptation of interfaces;
6. Call for conferences;
7. Study abstracts, keynote speeches, courses, and tutorials;
8. Studies that are not presented in English or Portuguese;
9. Studies whose full version could not be accessed.

With the set from the first search, we applied the inclusion and exclusion criteria, and after, the set of articles was reduced to 453 articles. Despite the exclusion of the keywords test, adaptation, and evaluation, many titles and abstracts pointed to papers on these topics. After their exclusion, the result was a new set of 375 articles. These were selected to be read and examined in their entirety and further refined to create a reduced initial set with papers related to the idea of this research. We have focused on selecting papers that show practical MDD implementations and accessibility/usability processes, as these can help us find relevant theoretical aspects.

In total, 13 (thirteen) articles were selected from *ACM Digital Library*, 17 articles from *IEEExplore*, 18 articles from *ScienceDirect*, 16 articles from *SpringerLink* (see Figure 10). So in total **64 articles** (see Appendix A, Table 16) for the initial set, from different years (Figure 11), from different authors and communities, which respect the diversity required by the protocol.

Figure 10 – Initial Set Papers and Databases

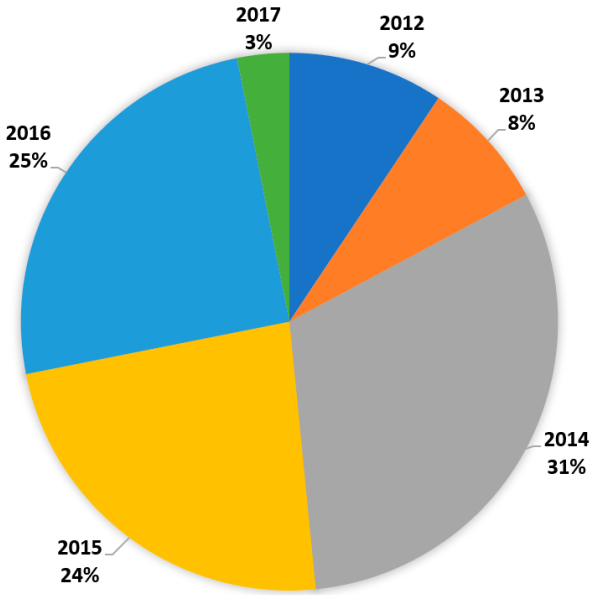


Source: Research data.

After defining the initial set for the proposed iterations in *Snowballing*, we used the search engine *Google Scholar* to analyze the other necessary factors such as the number of citations and the related articles. In this way, some articles were added as they were considered relevant to the research.

To keep the dataset updated, and include periodically relevant studies, we monitored the selected articles, strings, and citations in the Google Scholar engine. We also performed a

Figure 11 – Papers by year-of-publication, in the Initial Set Papers



Source: Research data.

search in the journals of the selected related works and backward/forward iterations with the new added papers. Then, a total of **187 articles** (see Appendix A, Table 17) were gathered from the backward/forward iterations.

As discussed in the previous chapters, producing accessible products is a challenging task. Some studies have directly inspired decisions about implementations of the approach proposed in this thesis and have been used to guide decisions. Based on the questions described in Subsection 3.1.1, the related works that most influenced this proposal were divided into two groups, as presented in the next sections: in Section 3.2, works that focus more on approaches, practices, or strategies that address accessibility/usability requirements from the early stages of the development process; and in Section 3.3, studies that generate the UI based MDD approaches, works that used the IFML language, and works that address the accessibility/usability requirements using model-driven approach to generate the UI.

3.2. Accessibility and Usability Early on the Development Process

Several studies in the literature propose to integrate accessibility into the process of Software Engineering and make recommendations to guide developers to create accessible features at particular stages of the development process. Even though our primary focus is on

accessibility, accessibility and usability are strongly related (ISO 9241-11, 2018). Some of the standards and papers refer to accessibility with usability features in the literature, primarily classifying accessibility as a subcategory of usability. ISO (2008) defines **accessibility** as the "**usability** of a product, service, environment, or facility by people with the broadest range of capabilities". For this reason, both accessibility and usability have been considered.

There is general agreement that accessible products are more usable for all and provide several benefits for all. Several studies have suggested that quality requirements such as usability and accessibility should be considered early in the development process. They all confirm that many usability and accessibility features are easier to implement if they are planned from the beginning of the development or redesign of a website or application and that some requirements elicited in the late software development phase cannot be realized (SHIROGANE, 2014). This section addresses some of them, those that had the most significant influence on the path taken in this work.

Dias, Fortes and Masiero (2012) extends the requirements engineering phase of Lowe and Pressman (2009) to include activities to incorporate usability and accessibility requirements from early web system development. The authors define tasks that should be performed during requirements elicitation to ensure usability and accessibility, such as defining alternative text for visual content and considering device independence. They emphasize the need to focus on the user necessities and reviewing the potential limitations posed by each disability category. This work was necessary for awareness of the concepts from the requirements elicitation stage. Complementary to this study, Dias (2014) describes a theoretical framework called *Process for Developing Web system with Accessibility and Usability* (PDWAU), which extended (LOWE; PRESSMAN, 2009) with usability and accessibility activities. Each activity (communication, planning, modeling, and construction) of the development process received an overall of the tasks used to boost the definition of roles and activities in the thesis proposal.

Steen-Hansen and Fagernes (2016) proposed a set of process-oriented guidelines to incorporate accessibility throughout the web application development and testing process so that accessibility becomes a natural part of the development process. The authors provide decisions about the development process, divided into five main sections: Have accessibility expertise on the team, communicate accessibility within the team, follow existing design principles, Test accessibility at key stages and log all accessible script modules, Introduce accessibility from the beginning.

The guidelines do not certify that the final product meets specific standards but state that the product development process occurred based on an approach that considered product accessibility decisions (STEEN-HANSEN; FAGERNES, 2016). It also emphasizes the importance of team communication with an accessibility specialist, continuously iterative, to guide decisions such as selecting languages and development support tools, as incorrect decisions could hinder accessibility implementation due to unsupportive resources. This work highlights essential

factors such as the relevance of awareness within the team or a component with accessibility expertise to achieve an accessible product.

Silva *et al.* (2019) proposed a conceptual model for incorporating accessibility throughout the software lifecycle, based on a methodology described as a hybrid that combines user-centered and agile development methods and consists of five phases: Analysis, Design, Coding, Deployment, and Maintenance. This theoretical approach objectively describes the phases with tasks for the traditions code-centric development process. This study enforced that, since coding is considered a manual task, the design detail defines the greater automation in code generation. Software development teams must have at least the basic knowledge of incorporating accessibility features into their coding activities. Tasks in this model were also mapped to our solution by correlating and transferring responsibilities and activities to specific stages of our MDD process.

Andrade *et al.* (2018) proposed an approach to integrating accessibility into the software development process using Acero, tool support that integrates with the process Methodology for Developing Accessible Web Applications (MTA)(MAIA, 2010) and integrates other approaches such as the Homero framework (OLIVEIRA *et al.*, 2014). Acctrace (BRANCO; CAGNIN; PAIVA, 2014) to support all phases of the development process. The process MTA follows the ISO/IEC 12207 (ISO 12207, 2008) development process. Even though we are not following ISO/IEC 12207, the set of sub-processes defined by Andrade *et al.* (2018) were investigated in the IFML language, especially some of the design and construction phases, and influenced theoretically of the approach on this thesis. For example, the responsibilities for Software design for the designer and the accessibility specialist involve designing the accessible external interfaces to provide multiple views to address the trade-off between diverse types of user groups. Also, establish the layout of the accessible interface elements, such as labels, images, text editing fields, and other elements considering the abilities of these users.

3.3. Model-driven User Interfaces Generation

There is an enormous amount of work on MDD that focuses on the generation of UI with different strategies and Domain Specific Languages (DSL). This section presents studies relevant to our work, including MDD-specific strategies for transformations between models and UI generation with references to accessibility and/or usability. Also, we included mobile UI generation (where we close the scope for proof-of-concept) and work that includes the use of the IFML language.

UI development using model-driven approaches can be described by a model or collection of models, which may consist of different layers and levels of abstraction that guide the development. Such models can also support the UI design and/or generation process of these UIs in a semi-automated or automated strategy, reducing the workload for UI development and making the process more productive (DELGADO *et al.*, 2016). According to Ahmed and

Ashraf (2007), the main idea of model-based user interface development is to identify valuable abstractions that can highlight the most important aspects and features of an interactive system and its design.

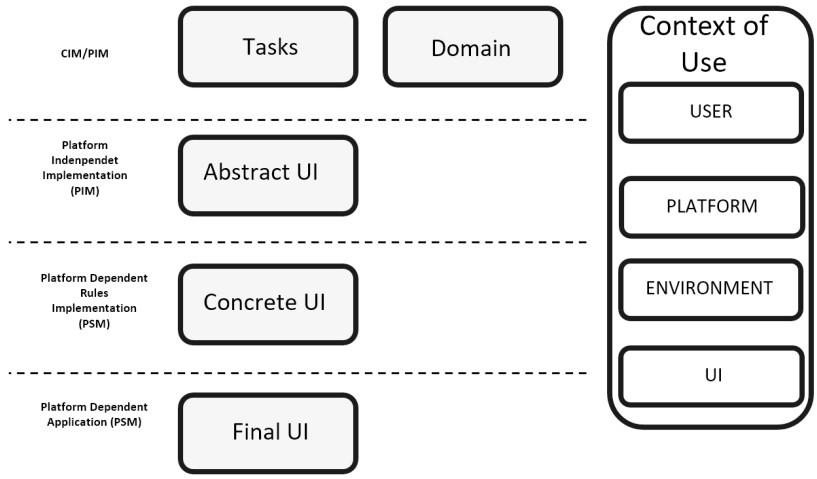
Martín, Cechich and Rossi (2011) present an approach to modeling web accessibility that moves from abstract to concrete architectural views and uses aspect orientation in the early stages of the development process through a UML-based tool, User Interaction Diagrams (UID), to represent user interaction in UML. The approach uses accessibility modeling as an independent aspect related to the architectural parts of the project. The authors argue that it is possible to address accessibility in the early stages of the development cycle independently of other quality requirements and integrate it with the architectural parts, which allows for the development of a more modular system and the reuse of accessibility aspects. According to the authors, the benefits of modeling accessibility in early design phases outweigh the need for a developer to implement it. Another highlighted benefit is that the designer can learn the basics of Web accessibility and should then be able to incorporate that knowledge into the software architecture.

The Martín, Cechich and Rossi (2011)'s work inspired the solution proposed in this thesis, although it does not consider all development process phases. It indicates the feasibility of defining abstractions and integration points related to UI components in the design phase and provides some accessibility recommendations required for MDD approaches. It is an evolution of the approaches in Martín *et al.* (2007), Martín, Mazalu and Cechich (2010), Bustos, Martín and Cechich (2010), and Martín *et al.* (2010). While focusing on the aspect-oriented approach in the design phase, it served to define some choices on the definition of the solution proposed in this work, such as finding integration points for accessibility requirements at different levels of component abstraction and presentation while maintaining the separation of concepts.

Cameleon Reference Framework (CALVARY *et al.*, 2003) defines a structure with four levels of abstraction: Tasks and Domain, Abstract User Interface, Concrete User Interface, and Final User Interface to classify user interfaces that support multiple targets or multiple contexts of use. Several works in the literature that traverse different abstraction levels to generate user interfaces refer to Cameleon's abstraction levels to generate applications from a higher level of abstraction to the final UI. The highest level of Cameleon (see Figure 12) is the tasks within the domain, from which an abstract UI is derived as a platform-independent model. Next is the concrete UI, which is platform-specific but still implementation-independent. At the end is the final UI, which represents the generated platform-specific code.

Ammar, Trabelsi and Mahfoudhi (2015) proposed an approach that provides a parameterized transformation for handling usability during the transformation process on MDD and extends the Cameleon framework with usability attributes for generating the concrete components. In this way, component representations are mapped according to defined and desired usability attributes, allowing for different presentation alternatives during automated transformations depending on the usability goal. An example is transformation options for a date input field component, which

Figure 12 – Cameleon Reference Framework



Source: Adapted from Calvary *et al.* (2003)

can be mapped to a drop-down list, or a radio button group for error prevention.

Ammar, Trabelsi and Mahfoudhi (2016) present further development of Ammar, Trabelsi and Mahfoudhi (2015) approach. The authors have included traceability mechanisms that define metrics using the conceptual primitives of the conceptual models, early usability evaluation, and recommendations for modeling process selection to allow the generation of usable applications. The work focuses on transformation and following the same principle as Ammar, Trabelsi and Mahfoudhi (2015), the studies made by Hentati *et al.* (2015) and Hentati *et al.* (2016) worked on the usability extension of the Cameleon framework as it moves from Abstract User Interface (AUI) to Concrete User Interface (CUI). All these studies try to conduct the generation of the components for the presentation of the final UI through usability attributes. Despite a usability specific strategy focused on transformation alternatives we brought features of the parameterized transformations as semantic properties to the AUI level design inspired on these approaches.

Zeferino and Vilain (2014) use the concept of abstract widgets as key to their approach. The authors define a CIM in which interactions with states, transitions, inputs, and outputs are described so that abstract widgets models (PIM) are generated as a combination of User Interaction Diagrams (UIDs) along with the automatic transformations. They use an ontology of abstract widgets consisting of 11 concepts to represent interface elements. Then, the abstract widgets model is transformed into platform-specific models using Java Server Faces (JSF) specific widgets (PSM) as an example and generation using Acceleo[7] specific templates. Although we did not use an ontology to define the abstract components, the concepts and ontology of the widgets helped us define the concepts for the extension and code generation. Also, we use the same transformation language for the code generator.

[7] <https://www.eclipse.org/acceleo/>

Generation of the UI level using MDD approaches can be achieved using a variety of techniques. MD^2 (HEITKÖTTER; MAJCHRZAK; KUCHEN, 2013a) is an MDD framework that focuses on generating cross-platform mobile business applications using the Model View Controller (MVC) architecture to generate native mobile code. It is based on a declarative textual DSL developed using `Xtext`[8] technology. This provides a textual editor with features such as syntax highlighting, content help, and validation that allows modelers to succinctly write the application model, views, controllers, and UI workflow.

Miñón *et al.* (2014) propose a methodological approach for integrating accessibility requirements into user interface development based on the User Interface Description Language (UIDL) language. More specifically, the authors use the framework USer Interface eXtensible Markup Language (UsiXML), which is an XML-compatible markup language that describes user interfaces for different contexts of use, such as graphical user interfaces, audio interfaces, user interfaces, and multimodal user interfaces (LIMBOURG *et al.*, 2004).

The strategy of Miñón *et al.* (2014) is to integrate accessibility requirements into metamodels and models at different points in the user interface development methodology, extracting the semantic expressiveness of UIDLs metamodels. They followed the *Accessibility for Web Applications* (AWA) process (MORENO; THESE, 2010) to abstract WCAG requirements, compatible with the Cameleon framework (CALVARY *et al.*, 2003). We applied similar strategy by abstracting the semantics of the recommendations to the abstract components to accommodate them in the DSL and also leverage various integration points of the MDD approach.

Miñón *et al.* (2014) argue that it is not a good strategy to make extensions to metamodels, citing as the main drawback the fact that it is not possible to use the tools offered by frameworks. We believe accessibility should not be overlooked by development processes, languages, and patterns. So, they should incorporate the ability to implement accessible software as a natural process, and tools also need to be extended accordingly. However, some concepts cannot be directly inserted into the models. Therefore, the strategy chosen in this work is very relevant because the requirements for accessibility in the development process are multidimensional and usually must be implemented and complemented in different stages.

Krainz and Miesenberger (2017a) have developed Accapto, a generic toolkit for accessible mobile application design and development that focuses on designing interactions with the accessible smartphone during the design phase using XML-based widgets. The first step of this approach is to describe the DSL model in XML and model the screen (Accapto uses XML tags for each element), UI elements and possible interactions in the application. The modeled application is enriched for accessibility at runtime using a library of accessibility patterns, according to the chosen platform and the profiles defined in the application definition at the beginning of the process (example: `profile: blind`).

[8] <https://www.eclipse.org/Xtext/> - `Xtext` is a framework for development of programming languages and domain-specific languages.

Runtime modules, such as a *TouchAreaExtender* module, allow the touch- target- area of UI elements to be extended at runtime for better accessibility adapting the final UI. The code generator provides an Android application prototype model with implementations for the platform-specific ATs. Despite the accessibility options, some features such as descriptions are still optional and no guidance is provided for modelers.

Krainz, Miesenberger and Feiner (2018a) conducted an accessibility evaluation of the Accapto-generated application based on WCAG 2.0 guidelines and the accessibility recommendations of "Material Design"[9]. The evaluation was performed with 42 participants. As a result, the authors state that MDD can significantly improve the accessibility of applications.

Vaupel *et al.* (2018) present mobile specific modeling language and an infrastructure for the MDD of native apps in Android and iOS. The authors argue that the approach allows a flexible app development on different abstraction levels with compact modeling of standard app elements such as standard data management and increasingly detailed modeling of individual elements to cover, for example, specific behavior. The paper presents a model called feature model that classify properties that are particularly relevant for the development of mobile native apps that support runtime configuration of mobile apps for different user and system contexts (VAUPEL *et al.*, 2018). We used some concepts of the feature model of this approach as reference, despite this do not cover accessibility or usability, to develop the IFML extension, to make our approach more flexible.

Bouraoui and Gharbi (2019) evaluated some UIDLs accessibility such as Context Toolkit (SALBER; DEY; ABOWD, 1999), Cortex (DURAN-LIMON *et al.*, 2003), UIML (HELMS *et al.*, 2009), UsiXML (LIMBOURG *et al.*, 2005), MariaXML (PATERNO'; SANTORO; SPANO, 2009), MD^2 (HEITKÖTTER; MAJCHRZAK; KUCHEN, 2013a), IFML (OMG, 2015a), Supple (GAJOS; WELD; WOBBROCK, 2010), Transport APP (KRAINZ; FEINER; FRUHMANN, 2016), and Android XML. It is Incorporated the reviews made by Mitrovic, Bobed and Mena (2016) and Chmielewski *et al.* (2016) with UIDLs for mobile computing, and concerning with them discuss the importance of separating UI concerns from the application concerns, also referring to Guerrero-Garcia *et al.* (2009) UIDL review. The languages accessibility evaluation can be seen, summarized in Table 3.

[9] <https://material.io/guidelines/usability/accessibility.html>

Table 3 – Summary of the UIDLs

Project	MDE/MBUI approach	Availability	Multi-platform/ Multi-device	Accessibility features	Code generation	Design/ Implementation/ Runtime
Context Toolkit	–	✔	Multi-platform Multi-device	–	No	Runtime
Cortex	–	–	Multi-platform Multi-device	–	No	Runtime
Compose	–	✔	Mobile	–	Partly	Implementation/ Runtime
UIML	✔	✔	Multi-platform	NDP [a]	Yes	Design
UsiXml	✔	–	Multi-platform	NDP [a]	Yes	Design
MD^2	✔	✔	Mobile (Android/iOS)	–	Yes	Design/Runtime
MariaXML	✔	✔	Multi-platform Multi-device	CBA [b]	Yes	Design
IFML	✔	✔	Multi-platform Multi-device	–	Yes	Design/ Implementation
Supple	–	–	Multi-platform Multi-device	Motor impairment	Yes	Design/Runtime
Transport App	✔	–	Mobile (Android)	Visual impairment	Partly	Implementation
Android XML	–	✔	Mobile (Android)	✔	UI	Implementation
IP MM/AccUI MM/ EP MM	✔	✔	Multi-platform Multi-device	✔	Yes (Markup) Partly (PL)[c]	Design

✔ - Does correspond
– - Does not correspond
[a] NDP: Not developed for this purpose
[b] CBA: Could be applied
[c] PL: Programming Languages

Adapted from: Bouraoui and Gharbi (2019)

Bouraoui and Gharbi (2019) present an MDD approach to assist developers throughout the development process of accessible UIs labeled as Accessible Design User Interface (AccDesignUI). The authors define different metamodels among them a new UIDL with abstract components with enriched semantics with aspects important for accessibility. This study uses models from different abstraction levels followed by transformations and provides a generic framework to integrate accessibility concerns starting from the design time (BOURAOUI; GHARBI, 2019). The modeling process goes from three basis metamodels Interaction Platform (IP) metamodel (modeling the target platform characteristics), Accessible User Interface (AccUI) metamodel (UI components), and Execution Platform (EP) metamodel (platform specific metamodels). The IP and AccUI models are transformed and merged into a model adapted model for the AccUI to the defined IP, then a second transformation is performed to generate a specific EP model, this model is refined to generate the final UI.

According to the authors, the approach allows the generation of accessible interfaces focusing on the graphical and adaptive aspects on different platforms, regardless of the developer expertise in UI accessibility. Despite maintaining separation of concerns on the AccUI metamodel complains about presentation details and the IP metamodel requires specific details of the

pretended target platform hardware specific details such as display technologies (LED, OLED, etc), mouse mechanisms description as well as software details that might be unknown to define basic accessibility.

Since its adoption by the OMG, the IFML language has been extended and used in various contexts. Brambilla, Mauri and Umuhoza (2014a) extended IFML to include mobile-specific components and described implementation experience that includes the development of automatic code generators for cross-platform mobile applications based on HTML5, CSS, and JavaScript optimized for the Apache Cordova framework. In the same context of mobile extensions but with a different approach, Fatima *et al.* (2019) use the IFML language as a PSM for Android applications by inserting on IFML specific native Android components and presentation details, and argue that this approach can help improve the usability of user interfaces generated with IFML.

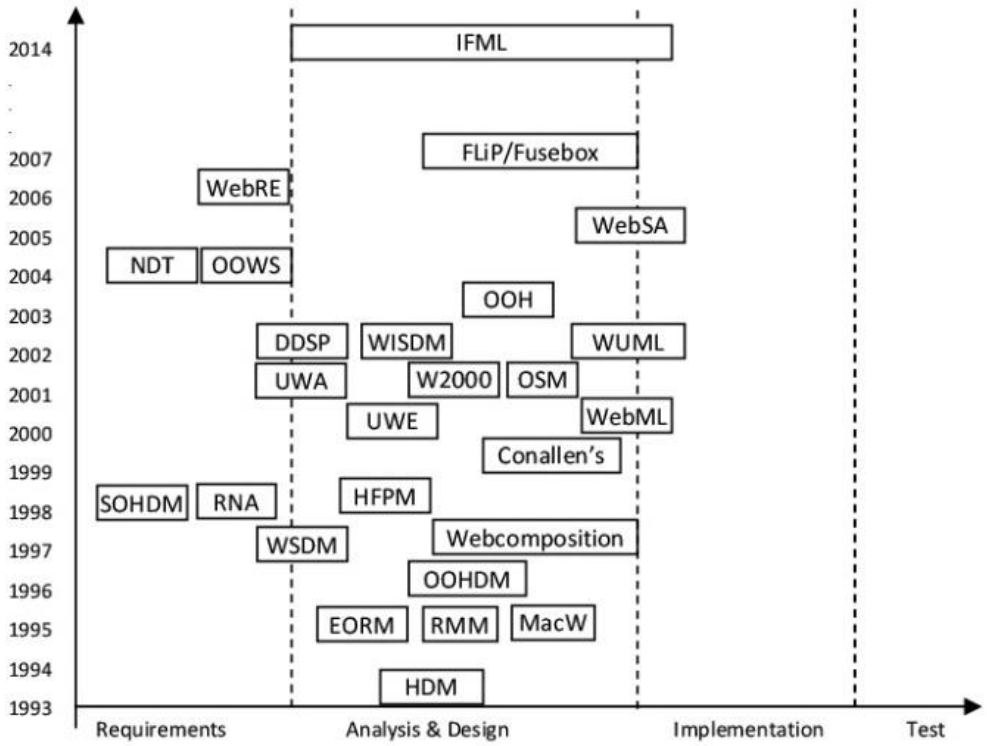
Wakil and Jawawi (2017a) analyzed IFML models in the Web development lifecycle to show the capability of the method in process development and discussed the role of IFML language in the development process of an interactive application. The authors concluded that IFML needs requirements but does not necessarily support them. They affirmed that IFML stands between the analysis/design and implementation phases and supports code generation, as shown in Figure 13, but starts with requirements gathering and usability testing, but does not fully support them and does not support formulation, planning. On the other hand, IFML can be integrated with other models to address these other phases.

Queiroz *et al.* (2018) conducted an empirical study to investigate how the usability of IFML models is perceived and propagated. The study used participants to model usability mechanisms with IFML; participants attempted to implement modeled mechanisms such as feedback, undo/cancel, wizard, and user profile (favorites list). They concluded that some mechanisms, such as undo, could not be perceived in the modeling phase. Some even perceived mechanisms, such as progress feedback, were not transferred to the final application based on the modeling. The study shows that it is possible to represent usability mechanisms easily perceived during modeling with IFML, but some improvements need to be perceived and propagated. We add this necessity also for accessibility and even for automatic transformations.

Roubi, Erramdani and Mbarki (2016a) extended the graphical part of IFML to generate graphical rich internet applications. For example, this extension added a component called Properties that composes the IFML SimpleField component to identify the suitable Rich Internet Application (RIA) corresponding component. The authors also proposed a metamodel for RIA applications.

In this context, Laaz and Mbarki (2016b) use ontologies in combination with IFML from abstract models to produce an HTML5 model. The authors used the IFML and Ontology Definition Metamodel (ODM) and integrated them with UI concepts. The logical model is defined with the ODM and to the interaction model with IFML; then they are combined to generate

Figure 13 – Languages and lifecycle coverage



Source: Wakil and Jawawi (2017a)

the final HTML5 UI. Examples of the combined transformations on PIM to PSM include features like the data properties (width, height, name, size, etc.) modeled on the ODM model and IFML InteractionFlowExpression and ActivationExpression to HTMLScriptElement defined for the same UI component. In addition, Laaz and Mbarki (2019) have developed OntoIFML as an approach that uses IFML-based tool plugins to facilitate web application development for annotated page generation.

Wakil, Jawawi and Rachmat (2018) comment on the limitations of the approaches of Roubi, Erramdani and Mbarki (2016a) and Laaz and Mbarki (2016b) to extend IFML for RIA applications. Wakil, Jawawi and Rachmat (2018) propose a different approach to extending IFML for RIA by adding RIAModel abstraction component to the core IFML package to model client and server applications, deriving all the main core IFML concepts as extended RIA concepts. The authors discuss the challenges of extending IFML, citing that each new RIA element presents a new challenge, such as reforming the model between interface and content and how to distribute the new element between client and server sides.

The works of Roubi, Erramdani and Mbarki (2016a), Laaz and Mbarki (2016b), and Wakil, Jawawi and Rachmat (2018) have limitations and do not address accessibility or usability. Nevertheless, the strategies for IFML extension and how they combine models to generate the UI with richer semantics inspired the way we conducted our work on extending IFML to

accommodate accessibility, even though we do not use ontologies.

Yigitbas, Mohrmann and Sauer (2015a) propose a model-driven interface development process that integrates HCI patterns. The authors show feasibility by implementing IFML-based Graphical User Interfaces (GUI) standards for building RIA applications. The AUI modeling is made with IFML models. According to the authors, the GUI patterns were formalized using instantiating parameters and application constraints corresponding to the extension of IFML metamodel components and model-to-model transformations performed using the Atlas Transformation Language (ATL). The integrated patterns are documented in a pattern catalog (general pattern meaning and instantiation parameters, and application constraints). An extension of IFML describes the formalization of the instantiation parameters. In contrast, the formalization of the application constraints is described by transformation rules that extend the model-to-model transformation written in ATL.

Yigitbas and Sauer (2016) present an MDD approach based on runtime context changes to create context-adaptive user interfaces that enable personalized, flexible, and task-based use of cross-channel applications. Supports modeling, transformation, and execution of context-adaptive user interfaces using Adapt-IFML and Context-IFML to adapt the application to the channels. The idea is extended and Yigitbas *et al.* (2020) introduce ContextML as a new modeling language for the context-of-use and AdaptML for UI adaptation rules. The first step is to model the AUI with IFML and UML class diagram, then adaptation modeling with the extended IFML context, which allows explicit modeling of UI adaptation rules, and Adapt-IFML, which specifies the adaptation rules. The models are transformed into different channels of the front-end and communicate at the back-end with an application server that performs adaptation and synchronization based on MAPE-K (COMPUTING *et al.*, 2006) with user/platform/environment as context determinants.

Although we initially excluded adaptive approaches from our initial selection of documents, these works emerged during our review because they use the IFML language. They do not use IFML ContextDimensions, and authors preferred to complement the modeling with their approach for context with a new language for that. Despite this characteristic, these approaches inspired this thesis proposed approach of our extension of the runtime Context feature of IFML, which we integrated on our IFML extension.

Rieger and Kuchen (2018), Rieger and Kuchen (2019b) developed a graphical DSL for specifying mobile business applications that also provides tool support for domain modeling, called Münster App Modeling Language (MAML). The authors argue that MAML addresses the tradeoff between technical complexity and graphical simplification that allows non-experts to model and generate Android and iOs apps and create standalone apps for Wear OS smartwatches without writing code manually. The language is based on five design goals: automatic cross-platform app creation, focus on domain experts for non-technical stakeholders, data-driven process, modularization, and declarative description.

The authors also place MAML between BPMN 2.0 and IFML in modeling complexity and affirm that IFML is the most similar approach. According to Rieger and Kuchen (2018), an advantage of MAML models is the fact that they are self-contained and do not rely on synchronization with external information, unlike IFML models, which are connected with other UIM standards. Rieger and Kuchen (2018) concluded that MAML has better readability compared to IFML. Despite this affirmation, MAML DSL is domain-focused, unlike IFML, whose focus is on user interfaces and interactions.

3.4. Final Remarks

This chapter describes the study conducted on the current state of the art in topics directly related to this research. First, the protocol used for the bibliographic review was presented, describing how the works were obtained. Then, selected works from the literature were presented that addressed accessibility and usability in the development process. This was followed by works dealing with the generation of interfaces based on models, works dealing with accessible model-based interfaces, and works dealing with the use of the IFML language.

To understand the activities and possible integration points, we started our studies with some approaches that helped us outline the different activities in the different phases of the development lifecycle to meet the accessibility requirements, guidelines, and recommendations. We reiterate that developers need to consider accessibility requirements at different stages of development and that these requirements must be introduced early in the development lifecycle.

We also explored MDD approaches to understand how these approaches handle user interface and interaction modeling and how the IFML language is used in this context. We also explored how non-functional requirements such as accessibility and usability are inserted at different levels of abstraction and how accessibility recommendations can be abstracted.

We reviewed the concepts and implementations of UI and the accessibility or usability features by examining the selected works approaches to outline the features of interest according to our state-of-art questions. Table 4 provides a summary of the reviewed works according to relevant features related to the present work. Comparison of the selected studies was made based on observed characteristics, regarding accessibility/usability integration, use of MDD, UI DSLs and IFML usage, type of editor, target platforms, and language. These reviewed works are grouped in 20 lines in the table, representing proposals their authorship.

There are 27 works from literature listed on Table 4. Twenty-three (23) of these are MDD approaches, of which twenty-two (22) deal with aspects of modeling the UI layer and only nineteen (19) generate code for that layer. Eight (8) papers deal directly with accessibility, four (4) of which are conceptual models (DIAS; FORTES; MASIERO, 2012; DIAS, 2014; STEEN-HANSEN; FAGERNES, 2016; SILVA *et al.*, 2019) that address accessibility/usability requirements in various processes and make theoretical definitions of the tasks involved in

Table 4 – Summary of Related Works

PROPOSAL	A/U[u]	MDD	Modeling Aspects	UI DSL	Language Editor	Target	Mobile Native Code
Martín, Cechich and Rossi (2011)	A	✔	UI, Behavior	UID	Graphical	HTML	–
Heitkötter, Majchrzak and Kuchen (2013a)	–	✔	Data, UI, Behavior	MD^2	Textual	Android iOS	✔
Dias, Fortes and Masiero (2012) Dias (2014)	A/U	–	CBA[a]	CBA[a]	CBA[a]	CBA[a]	CBA[a]
Miñón *et al.* (2014)	A	✔	UI, Behavior, Context	UsiXML	Textual/ Wizzards	XHTML	–
Brambilla, Mauri and Umuhoza (2014a)	–	✔	Data, UI, Behavior	IFML	Graphical	Apache Cordova	–
Zeferino and Vilain (2014)	–	✔	UI, Behavior	UID	Graphical	JSF	–
Ammar, Trabelsi and Mahfoudhi (2015) Ammar, Trabelsi and Mahfoudhi (2016)	U	✔	UI	AUI Metamodel	Tree editor	JAVA	–
Hentati *et al.* (2015) Hentati *et al.* (2016)	U	✔	UI	AUI Metamodel	Tree editor	JAVA	–
Steen-Hansen and Fagernes (2016)	A	–	CBA[a]	CBA[a]	CBA[a]	CBA[a]	CBA[a]
Roubi, Erramdani and Mbarki (2016a)	–	✔	UI	RIA extended IFML	Graphical	NS[b]	CBA[a]
Krainz and Miesenberger (2017a)	A	✔	UI	Accapto XML	Textual(XML)	Android	✔
Wakil, Jawawi and Rachmat (2018)	–	✔	Data, UI, Behavior	RIAModel extended IFML	Graphical	NS[b]	CBA[a]
Andrade *et al.* (2018)	A	✔	Requirements UI	Acero*	Textual(PHP) Wizzards Tree Editor	JAVA	–
Vaupel *et al.* (2018)	–	✔	Data, UI, Behavior	PIMAR	Tree editor	Android iOS	✔
Bouraoui and Gharbi (2019)	A	✔	Data, Context, Behavior, UI	AccUI/IP/ EP MM	Graphical / Tree editor	XHTML/ iOS Objective C	✔ iOS
Laaz and Mbarki (2016b) Laaz and Mbarki (2019)	–	✔	Data, UI, Behavior	OntoIFML	Graphical	HTML5	–
Fatima *et al.* (2019)	–	✔	Data, UI, Behavior	IFML Android (As PSM)	Graphical	NS[b]	CBA[a]
Rieger and Kuchen (2018) Rieger and Kuchen (2019b)	–	✔	Data, UI, Behavior	MAML	Graphical	Android iOS	✔
Silva *et al.* (2019)	A	–	CBA[a]	CBA[a]	CBA[a]	CBA[a]	CBA[a]
Yigitbas, Mohrmann and Sauer (2015a) Yigitbas and Sauer (2016) Yigitbas *et al.* (2020)	–	✔	Data, Context, Behavior, UI	UML/IFML/ AdaptML/ Context-ML	Graphical/ Textual	Angular	–

✔ - Does correspond
– - Does not correspond
* – Multiple tools
[a] CBA: Could be applied
[b] NS: Not Specified or do not generate code
[u] Accessibility (A) / Usability (U)

Source: Elaborated by the author.

implementing accessibility during the development lifecycle.

Most of the selected MDD approaches that generate the UI layer add the presentation features directly into the UI model, keeping it very close to the PSM, or add only the generated code, increasing the rigidity of the generated application. Nine (9) studies addressed mobile platforms, five generated native code, and one (1) generated code for Apache Cordova; the other three (3) studies did not address final UI code. Only six (6) of the selected papers consider context modeling, three (3) do not consider accessibility but include user and environment modeling (at least on the metamodel), one (BOURAOUI; GHARBI, 2019) a specific model for the AT including hardware and software specification details.

Five (5) studies address accessibility features at UI, all using different approaches. (KRAINZ; MIESENBERGER, 2017a) enriches UI for runtime accessibility, (MIÑÓN *et al.*, 2014) made improvements to AUI to meet accessibility requirements, including these requirements through Model to Model (M2M) transformation rules from the task model to the AUI model adding specifications according to the accessibility requirements of each modeled entity. Only two (2) of them (KRAINZ; MIESENBERGER, 2017a; BOURAOUI; GHARBI, 2019) deal with the accessibility creation of UI for mobile devices.

The MDD works that addressed usability implemented this through parameters for transformations rather than modeling them at an abstract level. Most of the DSLs used or created based on MDD approaches for generating UI do not support accessibility features.

Chapter 4 outlines the steps of our proposed MDD approach. We describe the extensions `AcIFML` and `AcFeat` used to support the proposal.

CHAPTER

4

MAMA - A MODEL-DRIVEN APPROACH MINDING ACCESSIBILITY

This chapter presents our proposal for a model-driven approach to generate accessible applications called "A **M**odel-driven **A**pproach **M**inding **A**ccessibility (**MAMA**)". The approach integrates accessibility recommendations and guidelines from the beginning of model-driven development by supporting this modeling in the UI language and providing tools to guide the modeling process. With this approach, we look after the thesis research aim *to develop a model-driven approach for generating accessible applications that consider accessibility requirements early in the development process*.

The chapter is organized as follows. The chapter begins with Section 4.1, which provides an overview of the proposed approach by addressing the research questions formulated in this thesis. The extensions to accommodate accessibility requirements in the IFML language are described in Section 4.2. Section 4.3 provides explanations of the conceptualization of the AcFeat Metamodel, which allows the modeling of presentation specifications and validations. Section 4.4 presents a feasible way to adopt the MAMA approach by integrating mobile accessibility recommendations to generate mobile apps. At the end, we discuss the final remarks in Section 4.5.

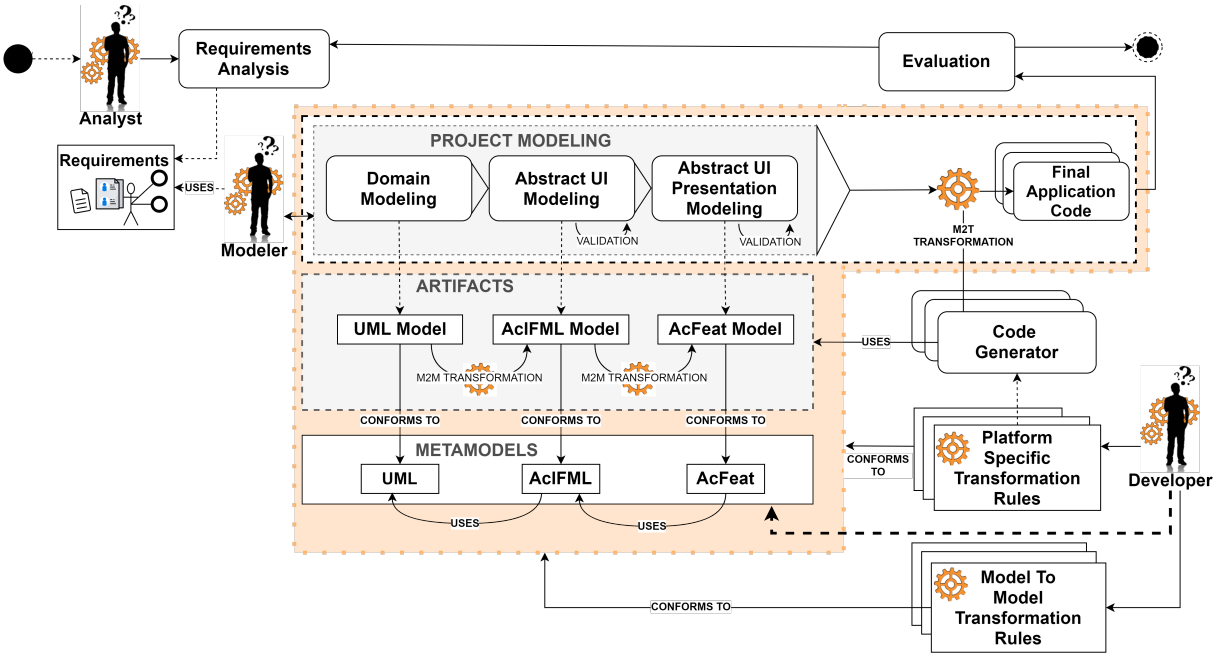
4.1. Approach Overview

The focus of this approach is on the front-end of the application, as most accessibility requirements will relate to the behavior associated with user interaction and usually requiring the use of Assistive Technologies (AT). However, to create a fully accessible application, requirements must be incorporated into the process from the beginning to deliver an architecture that properly supports accessibility. For the description of the approach, we return to our first research question, which is defined as follows:

[**RQ1.**] – *How can we support the modeling of user interfaces and interactions in applications to create more accessible systems using a Model-driven approach?*

MAMA aims to integrate accessibility recommendations and guidelines from the beginning of model-driven development by supporting this modeling in the UI language and providing tools to guide the modeling process. Figure 14 shows an overview of the approach with the steps, responsible persons and artifacts used during our proposed process.

Figure 14 – **M**odel-driven **A**pproach **M**inding **A**ccessibility (**MAMA**) Process



Source: Elaborated by the author.

The first step of the process is the "Requirements Analysis". In this phase, the system's key features that an analyst must be elicited to describe the application. The analyst must define the requirements that are fundamental to the modeling of any system. The main activities to be performed by the software analyst are the definition of functional requirements and tasks, the definition of user categories and abilities (types of disabilities for specific support), and the definition of accessibility requirements according to the defined tasks and user abilities, using any requirements template methodology. Once the requirements are defined, the next step is "Project Modeling". A modeler must perform three activities in this phase: Domain modeling, Abstract UI modeling, and Abstract UI presentation modeling.

In "Domain Modeling", the modeler must design a model according to the requirements, which stands for the relationships among the application's business domain objects and their attributes. This approach focuses on the basic structure needed to create a back-end model, as our focus is on user interfaces and interactions. We use the UML language to achieve this goal, with our primary artifact based on a class diagram. However, other models can be added to implement

more complex business behaviors, such as using the BPMN language, and other UML diagrams can be integrated into the process.

The artifact (UML model) created in the domain modeling phase is input to the "Abstract UI Modeling". It is used in the initial Model to Model (M2M) transformation to the IFML DomainModel when the modeler must select the model to start the creation of the `AcIFML` model via the model editor. After this step, the modeler starts specific abstract UI modeling activities. We assumed we must design an application for all users; however, allowing particularities description in the design at the abstract level when necessary, and the modeler can also model the `Context` (considering the user's abilities and the user and assistive technologies under concern).

So, the first is the general UI model for the application, and if necessary, the `Context` modeling and `Viewpoint` can be defined to provide UI customization for specific UI definitions. The modeler can use a different selection of components, a different organization of UI with user interactions and actions through the diverse viewpoints according to the defined contexts and interfaces by using the extended IFML to accommodate accessibility (`AcIFML`). This way, modelers can identify specific accessibility attribute requirements and propagate the necessary information to code generators. The UI model can be early validated concerning basic accessibility checks such as unadded descriptions and other recommendations.

Since IFML does not support presentation details, the next step for the modeler is to define an abstract presentation model using the `AcFeat` modeling language. At this stage ("Abstract UI Presentation Modeling"), the modeler can define abstract concerns related to the presentation properties of the application UI through themes and style definitions. In this way, the modeler can define presentation-related concepts such as the color palette, typography, decorations(non relevant for accessibility images) , and relative position that can later be mapped to specific platform concerns. Unlike the first M2M transformation, in this phase, the `AcIFML` model and the `AcFeat` model are instantiated together, and transformations occur at runtime on demand. This model is also validated according to some of the recommendations concerning presentation characteristics.

After all project modeling activities are complete, the modeler can trigger the "Code Generator", which performs a Model to Text (M2T) transformation to generate the final application code. The evaluation phase can be performed by automated tools, human specialists, tests with AT, and disabled users. Its results can impact requirement analysis, forcing a new iteration.

The infrastructure required to develop all stages of "Project Modeling" is the responsibility of the language developers who manage the metamodels. Another responsibility is to map the concerns among models and create the transformation rules and tools used by the modelers. The artifacts that are instantiated on the project modeling are created based on their metamodels. Three metamodels are used for the infrastructure of the approach:

(I.) the UML metamodel (modeling the domain),

(II.) the extended IFML (`AcIFML`) for (accommodating accessibility), and
(III.) the presentation metamodel `AcFeat`.

Table 5 summarizes the activities of the process with the responsible actor, the MDD objective, and the resources used and available for each activity. The `Analyst` responsible for defining the requirements, the `Developer` responsible for developing and maintaining metamodels, transformation rules, and model validation rules, and the `Modeler` working on developing the abstract UI model to represent the interfaces and user interactions using the `AcIFML` language.

Table 5 – **MAMA** approach activities and correspondent responsible persons

ACTIVITY	RESPONSIBLE	OBJECTIVE	RESOURCES
Requirement Analysis	Analyst	CIM	Requirements Documentations Diagrams
Metamodels	Developer	PIM	EMF (Metamodels/Editor) Eclipse Sirius (Editor) (UML, `AcIFML`, `AcFeat`)
Project Modeling	Modeler	PIM	Model Editor
Transformation Rules	Developer	PSM	Java (M2M) Acceleo (M2T)
Model Validation Rules	Developer	Accessibility validation	Java (Eclipse Sirius)

Source: Elaborated by the author.

As mentioned earlier, our focus is on UI and modeling user interactions for accessibility. To implement our proposal, we first focused on metamodels to provide a layer of DSLs that modelers can use to design accessible applications. In doing so, we initiated the integration of the necessary features from the recommendations and guidelines to answer our second research question, defined as follows:

> [**RQ2.**] – *How to abstract early on the model of user interface interaction flow to produce accessible applications?*

The initial design decision was to extend the IFML language. Section 4.2 shows the extended concepts on the language to accommodate the accessibility recommendations and guidelines that could be inserted at the AUI level.

4.2. `AcIFML` - Accommodating Accessibility on IFML

The IFML language, overviewed in Section 2.3, does not directly cover accessibility aspects. For this reason, we have proposed an extension to accommodate accessibility

requirements, recommendations, and guidelines. IFML is organized as a core set of concepts and a set of extensions that embody common features of many interactive applications. According to the documentation OMG (2015a), valid extensions should refine or adapt the core concept, or provided extensions, for specific cases and specialize their semantics without changing them, and not all possible extensions are allowed. To comply with the IFML standard, only the following concepts (and their extensions) are allowed for extensions: `ViewContainer`; `ViewComponent`; `ViewComponentPart`; `Event`; `DomainConcept and FeatureConcept`; `BehaviorConcept and BehavioralFeatureConcept`.

The documentation cites that extensions of other elements are not allowed. However, runtime ContextDimension may be specialized to represent other dimensions, such as user preferences, etc (OMG, 2015a). The language extension can be developed on the four artifacts provided in the specification. We used the IFML metamodel as the primary extension mechanism. For better understanding, we fragmented the extended components into the following four sections:

(a.) Accessible ViewContainers (`acViewContainers`) in Subsection 4.2.1 - contains the extended ViewContainers, which are first level elements of the interface that group other ViewContainers and/or ViewElements to display the content.

(b.) Accessible ViewComponents (`acViewComponents`) in Subsection 4.2.2 - the extended ViewComponents, which are user interface elements that display content or accept input and may be bound to a ContentBinding via its association with ViewComponentPart.

(c.) Accessible ViewComponentParts (`AcViewComponentParts`) in Subsection 4.2.3 - the extended ViewComponentPart which are interactive elements (InteractionFlowElement) that do not live outside a ViewComponent and may have incoming and outgoing flows.

(d.) Accessible ViewElementEvents, Context and ViewElementBehavior in Subsection 4.2.4 - the extended ViewElementEvents, that represent a user interaction event, a new concept "`AcViewElementBehavior`" used for modeling UI behavior such as the flow for a keyboard navigation and the extended context dimensions to model accessibility runtime context.

In this stage, we aimed to accommodate the accessibility requirements at a higher abstraction level so that the modelers could perceive them. They can also be propagated to the next steps of the MDD development process and be used in the transformation process. WAI guidelines such as WCAG and WAI-ARIA serve as the primary source, as almost all other documents use them as the primary reference, but the extension is not limited to them. Before adding the above extensions, it was a design decision to create a representation in the language metamodel with the abstractions that should be common to all accessible view elements. Since the core `ViewElement` is not extensible, we added an intermediate abstract component to the

language called "AcViewElement". Table 6 shows the semantics used for this component and the references that supported its creation, from which all accessible components will inherit the semantics.

Table 6 – Abstract AcViewElement

Concept	Semantics	Principle	Core References
AcViewElement	**description**: used for short description/ titles/ explicative texts	Perceivable	**WCAG 2.1** 1.1 - Text alternatives 1.1.1 - Non-text content Success Criterion 2.4.2 Page Titled Success Criterion 2.4.6 Headings and Labels Success Criterion 3.3.2 Labels or Instructions Success Criterion 2.4.10 Section Headings
	idiom: defines the idiom of page or parts	Understandable	3.1 - Readable 3.1.1 - Language of Page 3.1.2 - Language of Parts
	isFocusable: defines if the component is focusable (if it can receive keyboard focus) **isTouchFocusable**: defines if the component is focusable in touch mode	Operable	2.1 - Keyboard Accessible Success Criterion 2.1.2 No Keyboard Trap
	isDisabled: boolean	Operable State	**WAI ARIA 1.1** 2.2 - States and Properties 2.3 - Managing Focus
	isVisible: boolean	Operable State	

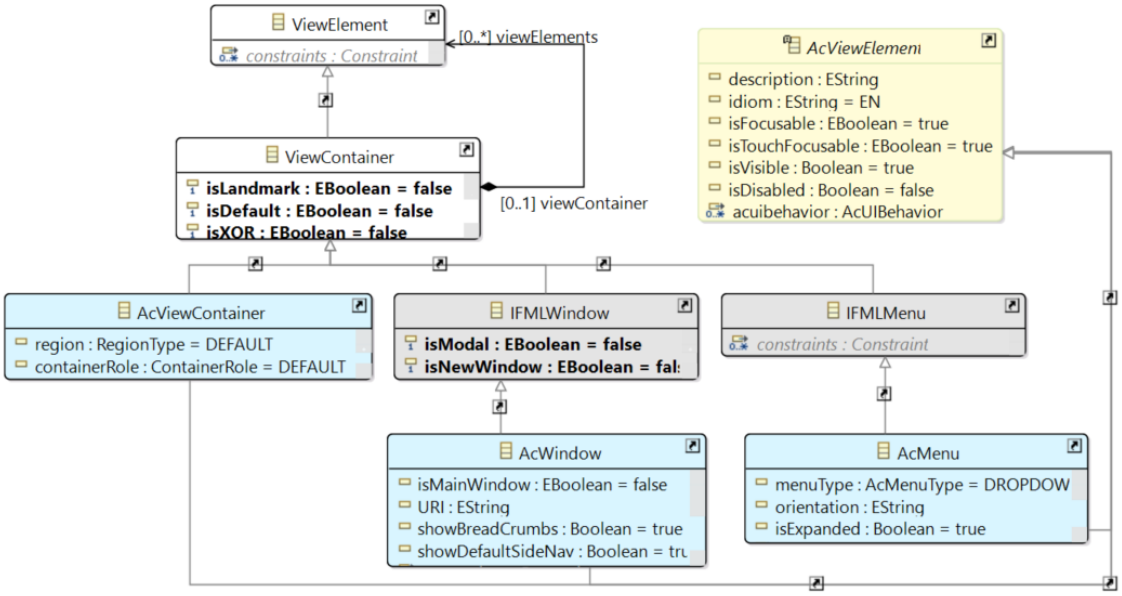
Source: Elaborated by the author.

All extended ViewContainers (Subsection 4.2.1), ViewComponents (Subsection 4.2.2), and ViewComponentParts (Subsection 4.2.3) inherited the concepts of AcViewElement. This component received five semantic attributes that are used to meet important accessibility recommendations of WCAG 2.1 and WAI-ARIA guidelines: description, idiom, isFocusable, isDisabled, and isVisible. IFML has a concept called ActivationExpression that is used to determine whether ViewElements, ViewComponentPart, and Event are enabled. This is a boolean expression and the decision to use a semantic attribute isDisabled was made to reduce subjectivity regarding the initial state of the elements. Figure 15 shows a fragment of the AcIFML metamodel where we can visualize the AcViewElement component.

4.2.1. Accessible ViewContainers

ViewContainers are first level components of the UI which groups other ViewContainers and/or ViewElements to display content. Figure 16 shows the components provided in this category minding accessibility based on the guidelines and research. We defined three components in this category: AcViewContainer, AcWindow, and AcMenu. These components are system generic, for example they are not mobile specific, can be used on different system context or if necessary also extended for a finer granularity.

Figure 15 – AcIFML Metamodel Fragment - Accessible ViewContainers



Source: Elaborated by the author.

Table 7 outlines the components, semantic attributes, and references. `AcViewContainer` and `AcWindow` have the same reference on the guidelines, but almost with different characteristics to satisfy different roles, platforms, and applications. `AcWindow` can be defined at the first level to implement concepts like mobile screens or windows. `AcViewContainers` can define region-specific containers such as headers, footers, page dividers, and toolbars.

Another defined component is `AcMenu`, which is is a composed component described in the extension closely related to the `AcMenuIten` and `AcSubmenu` view components. Antonelli, Silva and Fortes (2015) have inspired the `AcMenu` extension by defining an abstract structure and relationship to `AcMenuIten` and `AcSubmenu` (see Figure 16), which supports accessibility in terms of modeler requirements and automatic transformation, propagating even its roles.

Table 7 – Abstract `AcViewContainers`

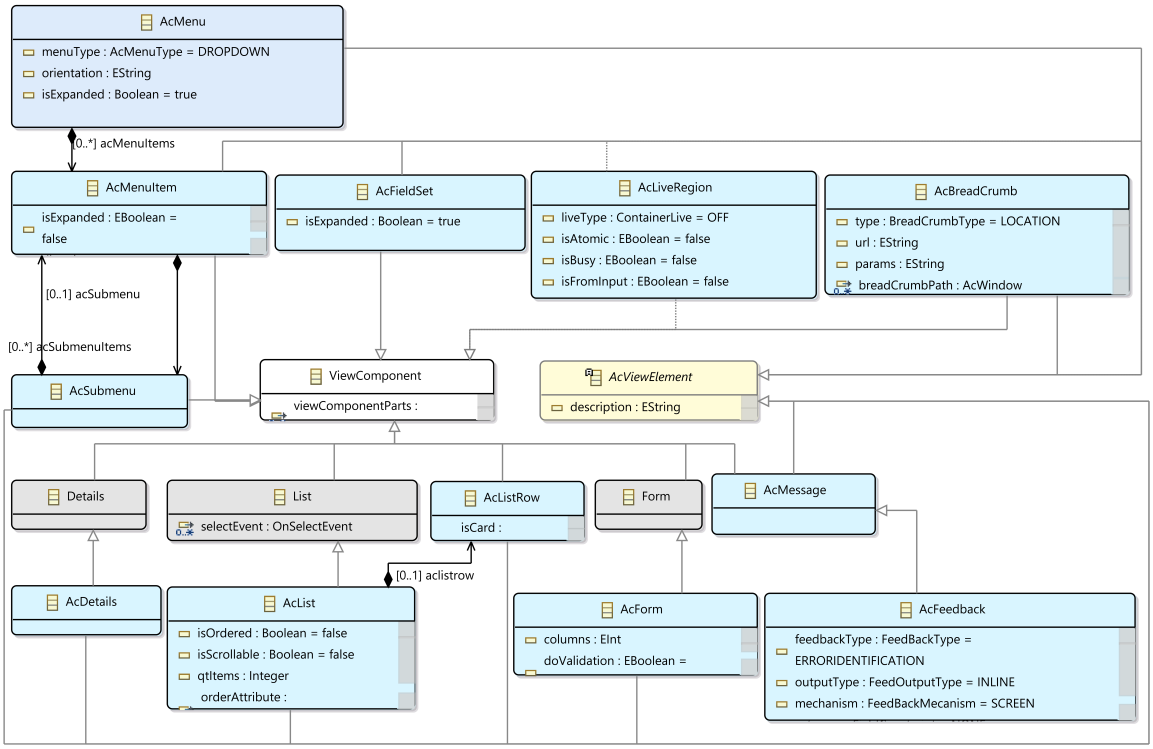
Concept	Semantics	Core References
`AcViewContainer`	**region**: RegionType **containerRole**: ContainerRole	**WCAG 2.1** – Success Criterion 1.3.1 Info and Relationships – Success Criterion 2.4.1 Bypass Blocks – Success Criterion 2.4.2 Page Titled **WCAG 2.1 Techniques** - G115: Using semantic elements to mark up structure (Sufficient) – G140: Separating information and structure from presentation to enable different presentations – H42: Using h1-h6 to identify headings (Sufficient) **WAI-ARIA 1.1** – 1.1 Authoring Practices – 4 Landmark Regions – 4.3 Landmark Roles
`AcWindow`	**isMainWindow**: boolean **showBreadCrumbs**: boolean **showDefaultSideNav**: boolean	**WCAG 2.1** – Success Criterion 1.3.1 Info and Relationships – Success Criterion 2.4.1 Bypass Blocks – Success Criterion 2.4.2 Page Titled **WCAG 2.1 Techniques** – G115: Using semantic elements to mark up structure (Sufficient) – G140: Separating information and structure from presentation to enable different presentations – H42: Using h1-h6 to identify headings (Sufficient) **WAI-ARIA 1.1** – 1.1 Authoring Practices – 4 Landmark Regions – 4.3 Landmark Roles
`AcMenu`	**menuType**: AcMenuType **orientation**: String **isExpanded**: boolean	**WCAG 2.1** – 3.2.3 Consistent Navigation – Success Criterion 4.1.2 Name, Role, Value **WAI-ARIA** – 2.2 States and Properties (Antonelli, Silva and Fortes (2015))

Source: Elaborated by the author.

4.2.2. Accessible ViewComponents

We have created eleven accessible ViewComponents (see Figure 16). All of them inherited the semantics of `AcViewElement` and some of them have special accessibility attributes to define their state, like isExpanded in the case of `AcMenuItem` and `AcFieldSet` components. The `AcList` component is a composite component that can accept a `AcListRow` as content. These component relationships were created to provide role awareness for modelers and automatic transformation and reference the WCAG guideline Success Criterion 1.3.1 regarding relationships and structure, which refers to meaningful associations and how parts are organized with each other.

Figure 16 – **AcIFML** Metamodel Fragment - Accessible ViewComponents



Source: Elaborated by the author.

Some of the extended ViewComponents such as `AcDetails`, `AcMessage`, where created for fine granularity. The components `AcSubMenu` and `AcMenuItem` are ViewComponents that compose the relations of the ViewContainer `AcMenu`. And `AcMenuItem` has isExpanded property to model its state. Table 8 shows the other components such as `AcLiveRegion`, `AcBreadCrumb`, `AcFeedback`, and `AcFieldSet` that were abstracted from the guidelines.

`AcLiveRegion` is the component used to define areas of dynamic content, such as those that are updated without reloading the page. This feature is important for ATs like screen readers, as it automatically announce it's content when it changes. `AcBreadCrumb` is the component that supports the breadcrumb trail, which consists of, for example, a list of links to parent pages. `AcFeedback` is a message component that can be used with various output mechanisms, such as screen or haptics, through system events such as sounds and vibration. `AcFieldSet` is a grouping-specific component that can be used, for example, to group content in a form or detail window, which is important, for example, to specify a Screen Reader way of reading the content of the UI.

Table 8 – Abstract `AcViewComponents: AcBreadCrumbs, AcFeedback, AcLiveRegion, AcFieldSet`

Concept	Semantics	Core References
`AcBreadCrumb`	**type**: BreadCrumbtype **uri**: String **params**: String **breadCrumbPath**: String	**WCAG 2.1** – 2.4 Navigable – Success Criterion 2.4.5 Multiple Ways – Success Criterion 2.4.8 Location – Success Criterion 3.2.3 Consistent Navigation **WCAG 2.1 Techniques** – G65 Providing a breadcrumb trail **WAI-ARIA 1.1** – 1.1 Authoring Practices – Design Patterns – Breadcrumb Pattern
`AcFeedback`	**feedbackType**: FeedBackType **outputType**: FeedOutputType **mecanism**: FeedbackMecanism	**WCAG 2.1** – Success Criterion 3.3.1 Error Identification – Success Criterion 3.2.2 On Input – Success Criterion 3.3.5 Help – Success Criterion 4.1.3 Status Messages **WCAG 2.1 Techniques** – G199: Providing success feedback when data is submitted successfully
`AcLiveRegion`	**liveType**: ContainerLive **isAtomic**: boolean **isBusy**: boolean **isFormInput**: boolean	**WCAG 2.1** – Success Criterion 3.3.1 (Error Identification) **WAI-ARIA 1.1** – 5.3.5 Live Region Roles
`AcFieldSet`	**isExpanded**: boolean	**WCAG 2.1** – Success Criterion 1.3.1 (Info and Relationships) – Success Criterion 2.4.10 Section Headings **WCAG 2.0 Techniques** – ARIA17 Using grouping roles to identify related form controls **WCAG 2.1 Techniques** – H71 Providing a description for groups of form controls using fieldset and legend elements – H82 Grouping form controls with FIELDSET and LEGEND – H85 Using OPTGROUP to group OPTION elements inside a SELECT

Source: Elaborated by the author.

4.2.3. Accessible ViewComponentParts

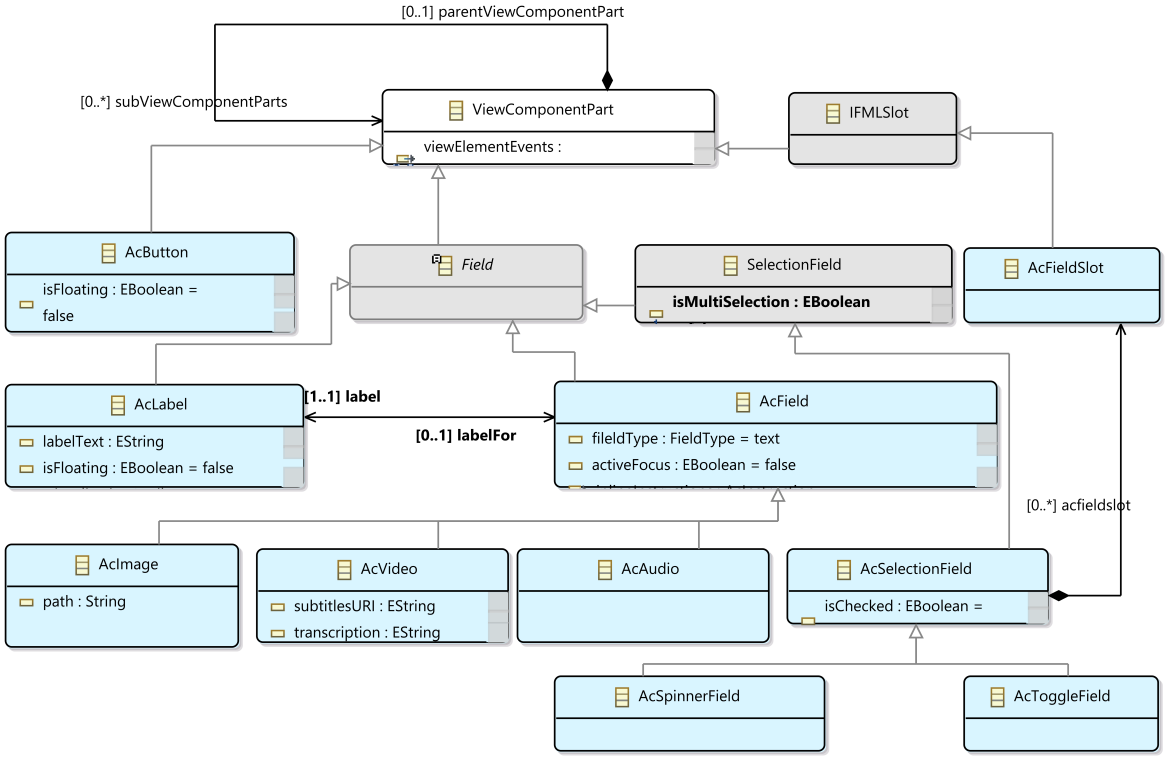
We derived about ten accessible ViewComponentParts (see Figure 17), that are elements that may not live outside the context of a ViewComponent, may trigger Events and have in and out "InteractionFlows".

`AcField` extends the `Field` component and adds to the concepts of `AcViewElement` the semantic attributes `fieldType`, which refer to the keyboard input type relevant for accessibility,

and a state property activeFocus, which is used to determine the initial state of the focus when needed.

We decided to create AcLabel, which is used to represent plain text, and establish a semantic relationship with AcField that specifies the mandatory need for a label for the accessible fields. We also extended SelectionField and added the state isChecked, and derived two more ViewComponentParts from it AcSpinnerField, to model spinners inputs, and AcToggleField to model two state inputs.

Figure 17 – **AcIFML** Metamodel Fragment - Accessible ViewComponentParts



Source: Elaborated by the author.

There are three extensions of AcField that are the components created to support accessible media: AcImage, AcVideo, AcAudio. It is important to remember that the attributes from the components such as path on AcImage, subtitlesURI and transcription from AcVideo are semantic and not final values, so their values may bind to be parameters from the domain model or even expressions and to comply with WCAG they must be predefined, this way coming from the domain model.

Table 9 shows the extended ViewComponetParts whose semantics have been expanded, associated with references from the guidelines: AcField, AcLabel, AcButton, and AcVideo.

Table 9 – Abstract `AcViewComponentParts`

Concept	Semantics	Core References
`AcField`	**inputType**:FieldType **isRequired**:Eboolean **activeFocus**:boolean **fieldPattern**:AcFieldPattern	**WCAG 2.1** – 2.5 Input Modalities – Success Criterion: 1.3.1 (Info and Relationships) – Success Criterion 1.3.5 (Identify Input Purpose) – Success Criterion 3.3.4 (Error Prevention) **WCAG 2.1 Techniques** – G83: Providing text descriptions to identify required fields that were not completed – G89: Providing expected data format and example
`AcLabel`	**isFloating**:boolean	**WCAG 2.1** – Success Criterion 2.4.10 Section Headings – Success Criterion 2.5.3 Label in Name – Success Criterion 3.3.2 (Labels or instructions) – Success Criterion 3.3.5 (Help) **WCAG 2.1 Techniques** – H44: Using label elements to associate text labels with form controls – G131: Providing descriptive labels – G162: Positioning labels to maximize predictability of relationships – H90: Indicating required form controls using label or legend
`AcButton`	**isFloating**:boolean	**WAI-ARIA Authoring Practices 1.1** – 3.5 Button
`AcVideo`	**subtitles**: String **transcrition**: String	**WCAG 2.1** – 1.2.1 Audio-only and Video-only (Prerecorded) – 1.2.2 Captions (Prerecorded) – 1.2.3 Audio Description or Media Alternative (Prerecorded) – 1.2.5 Audio Description (Prerecorded)

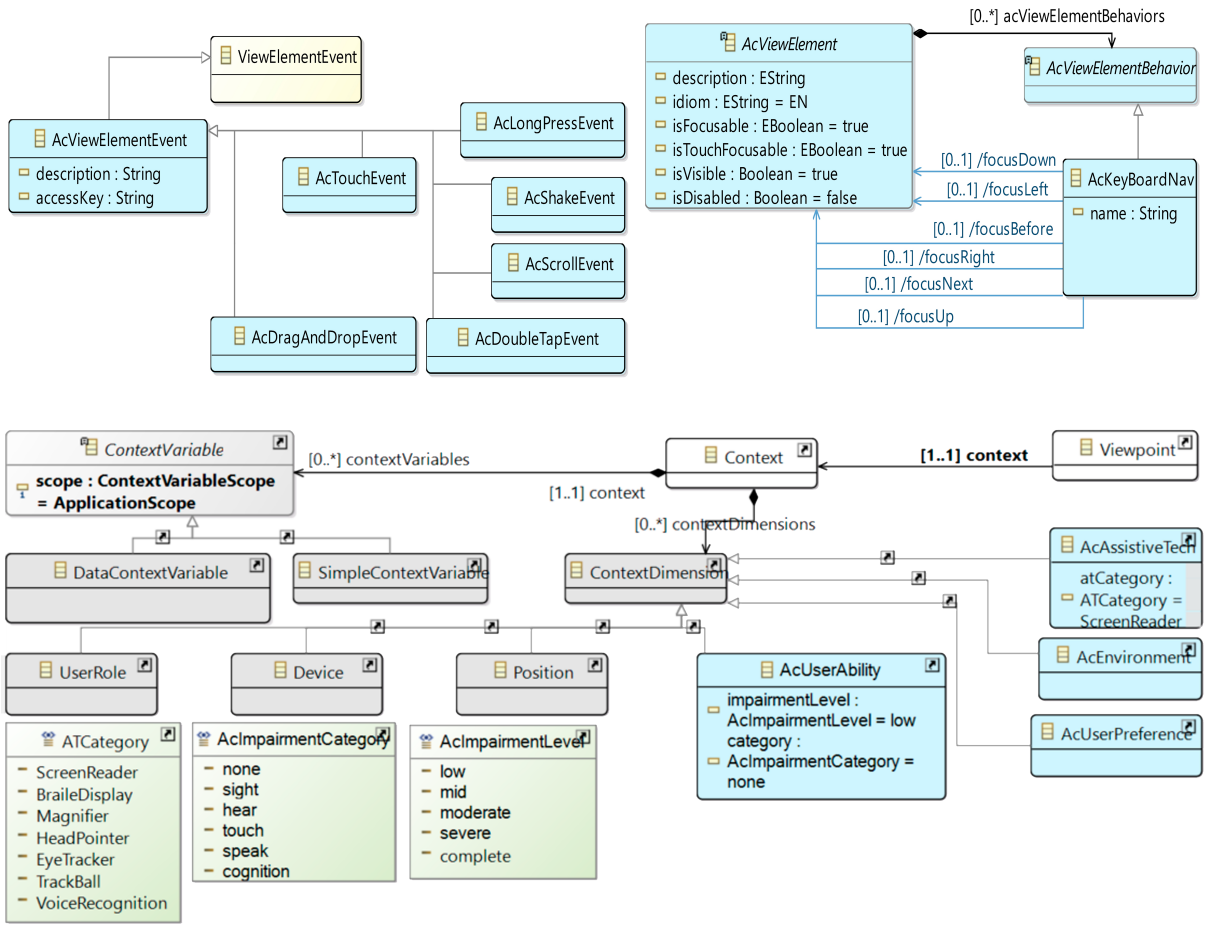
Source: Elaborated by the author.

`AcButton` is a part used to model a button pattern, which is a component of UI that can receive events. Developers can map `ViewElementEvents` such as `OnSubmitEvent`to buttons, anchors, or simple link events in the transformation rules. However, we created this abstraction to distinguish clickable action interactions from navigation, to bring the concept closer to modelers' intuition, and to design different final layouts for the same UI depending on the active `Context`. `AcButton` has a semantic property called isFloating, a boolean value that defines its state at UI and is relevant for accessibility configuration, e.g., keyboard accessibility.

4.2.4. Accessible ViewElementEvents, ViewElement Behavior, and Context

Figure 18 shows the concepts extended for user interaction events, the UI behavior, and runtime Context.

Figure 18 – AcIFML Metamodel Fragment - Extended Events, Context Dimensions and AcViewElementBehavior



Source: Elaborated by the author.

A ViewElementEvent represents a user interaction event that can be triggered by a ViewElement. It is an important element of UI, as it defines user interactions in the design and allows the modeling of user events. AcViewElementEvent extends the core ViewElementEvent meta-class of the IFML standard and includes references from the guidelines inserting on its semantics the description that can be used for accessibility description for the interaction and accessKey, which can be used to define keyboard shortcuts. The events OnSubmit and OnSelect have been extended with the AcOnSubmit and AcOnSelect components with the same properties for accessibility.

The following meta-classes were also added to the extension, inspired by the Brambilla, Mauri and Umuhoza (2014a) extension that extends `AcViewElementEvent`: `AcDragAndDropEvent`, `AcDoubletapEvent`, `AcSwipeEvent`, `AcPinchEvent`, `AcSpreadEvent`, `AcTouchEvent`, `AcLongPressEvent`, `AcScrollEvent`, `AcShakeEvent`, each representing a gesture event for the mobile context.

Table 10 – Abstract `AcViewElementEvent` and `AcKeyboardNav`

Concept	Semantics	Core References
`AcViewElementEvent`	**description**:String **accessKey**:String	**WCAG 2.1** 2.1.1 Keyboard **WCAG 2.1 Techniques** G202: Ensuring keyboard control for all functionality G90: Providing keyboard-triggered event handlers SCR20: Using both keyboard and other device-specific functions SCR2: Using redundant keyboard and mouse event handlers
`AcKeyboardNav`	**name**:String Derived References: *focusUp* *focusDown* *focusBefore* *focusRight* *focusNext* *focusUp*	**WCAG 2.1** 2.1 Keyboard Accessible **WCAG 2.1 - Success Criterion** 1.3.2 Meaningful Sequence 2.1.1 Keyboard 2.1.2 No Keyboard Trap 2.4.3 Focus Order 2.4.7 Focus Visible **WCAG 2.1 Techniques** H4: Creating a logical tab order through links, form controls, and objects

Source: Elaborated by the author.

We decided to create an abstract concept for modeling the behavior of UI called `AcViewElementBehavior` with the extended component `AcKeyboardNav`, which the modeler can use as needed to model the keyboard interaction sequence. By default, keyboard navigation follows a simple sequence. Still, sometimes it is necessary to model customized keyboard navigation, for example, to avoid common keyboard traps when UI presents complex components such as lists or date pickers that could trap the user, or even for complex UI designs.

Other relevant concept that was extended was the `Context` through the `ContextDimensions`. The `Context` has a special meaning for accessible applications. Although we strive to have one design for all users, it is sometimes necessary to create specific designs for a particular accessibility context. The defined dimensions can help the modeler to prepare specific UI and interactions according to the accessibility context. We have defined `ContextDimensions` extensions that allow user and accessibility specific runtime definitions.

`AcUserAbility` is a dimension that reflects the persona spectrum categories of impairment according to the affected sense in terms of permanent, temporary, and situational

disabilities (World Health Organization *et al.*, 2001; MICROSOFT, 2016; O'NEILL, 2021), grouped into four categories: sight, hear, touch, speak, and cognition. Other abstractions included the AcAssistiveTech to contexts corresponding to specific ATs, the AcEnvironment, and AcUserPreference. It is important to remember that these dimensions are associated to ViewPoint, this way to access a viewpoint all the dimensions defined on a context have to be satisfied. The modeler can define the necessary contexts for the UI design.

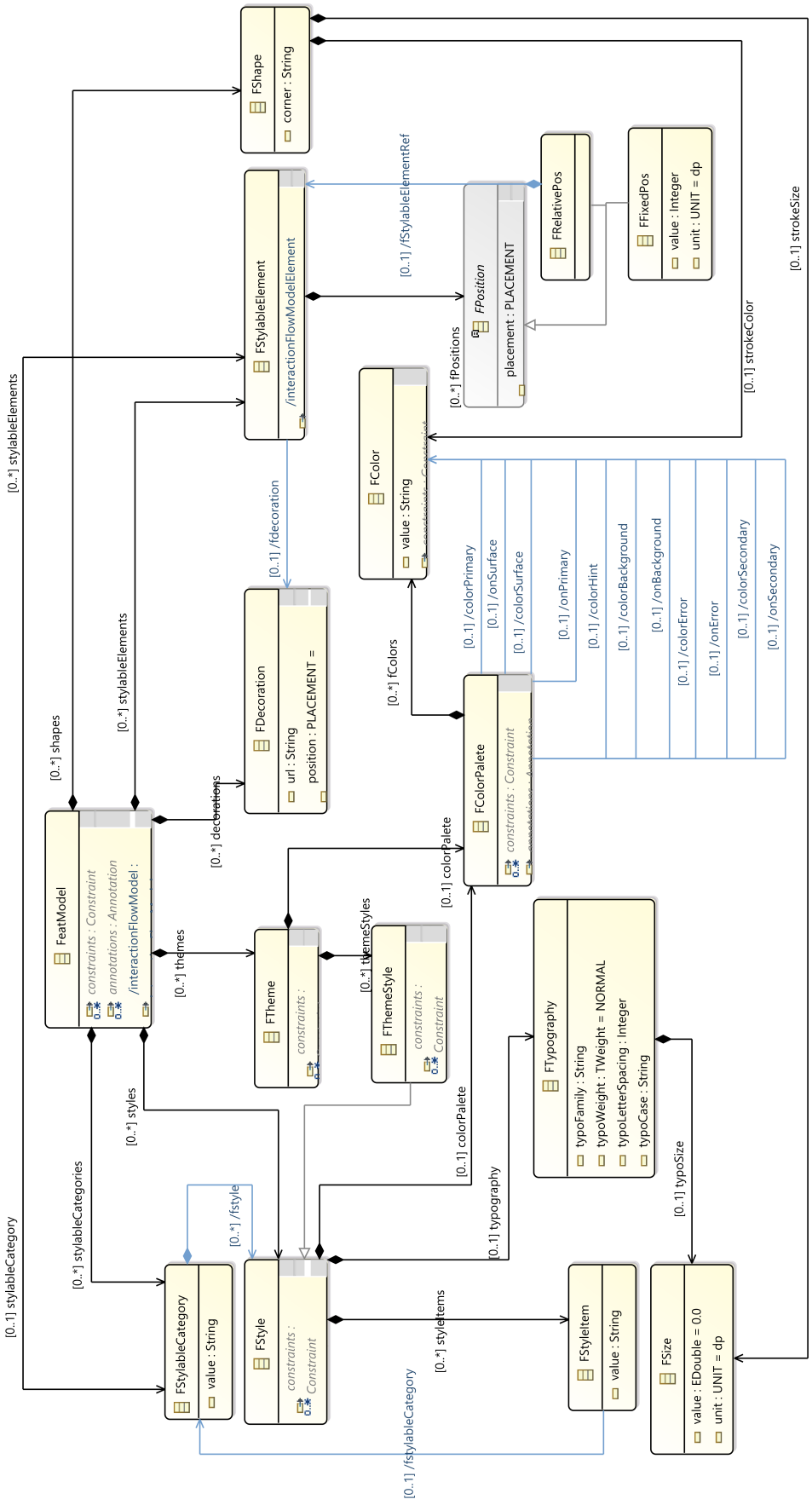
4.3. AcFeat Metamodel - Presentation Aspects

Since IFML does not support the presentation layer, we created a metamodel called AcFeat to provide more flexibility and accessibility to presentation specifications and validations. This metamodel is inspired by Material Design themes[1], which consists of three main actions: customization of the theme, applying across the design mocks, and using in code. Material Design[2] is a design system developed by Google to help teams create high-quality digital experiences for multiple platforms such as Android, iOS, Flutter, and the web. It defines the qualities that UI regions and components can express through concepts such as color, typography, iconography, shapes, and layout with the goal of consistency across platforms.

For this study, not all concepts of Material Design themes were considered, but we create a structure that can be extended. Figure 19 shows the concepts created for the presentation layer. The main meta-class of this metamodel is FeatModel, which refers to the "InteractionFlowModel" from the IFML core to provide presentation aspects. One FeatModel can contain FStyles, FStylabelCategories, FDecorations, FShapes, and FStylableElements.

[1] <https://material.io/design/material-theming/overview.html#material-theming>
[2] <https://material.io/design>

Figure 19 – **AcFeat** metamodel - UI presentation aspects



Source: Elaborated by the author.

FTheme is the concept that aggregates general presentation aspects for the application. When defining a theme, the modeler can specify styles for component categories (FStylableCategory) for general, directly assignable elements and specific customizations. Each theme defines its colors, which are grouped on the color palette (FColorPalete) and follow the material design color system. The other concepts of the metamodel correspond to typography concepts (FTypography), FDecorations for iconography, FShapes for defining UI shapes and a position system (FPosition) that can be fixed (FFixedPos) or relative to other components (FRelativePos).

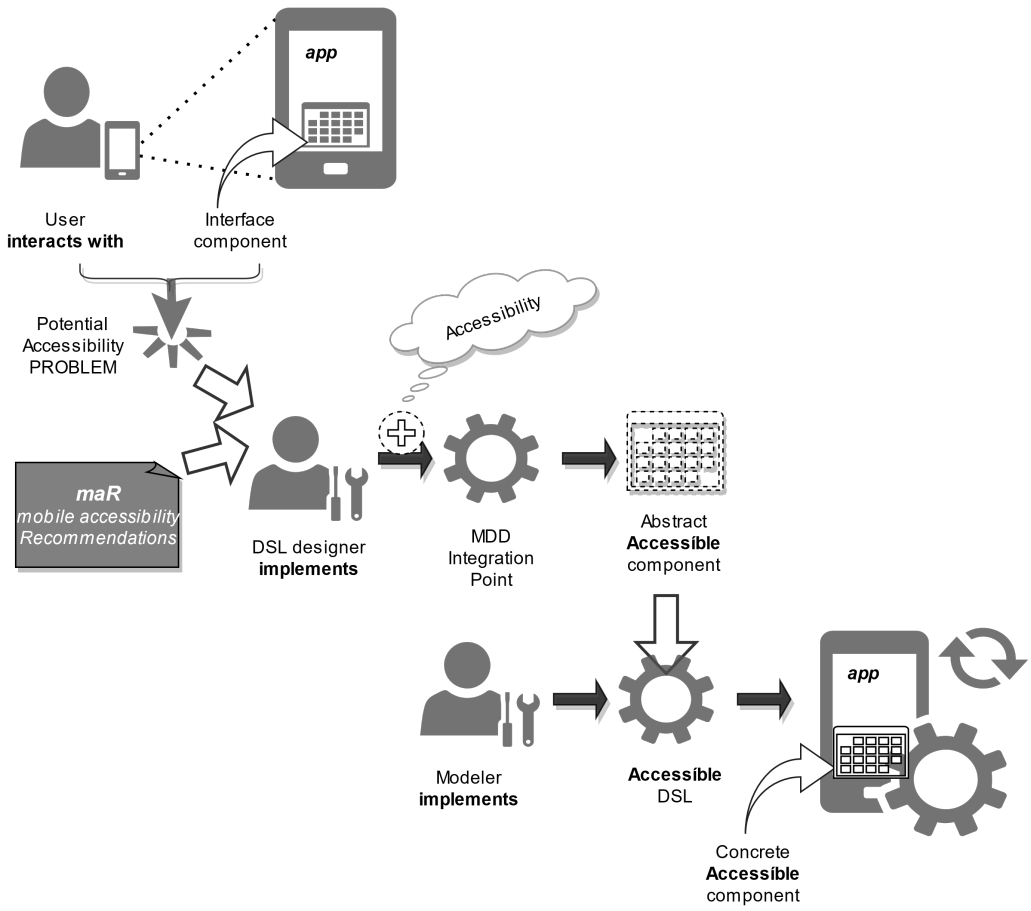
4.4. Integrating Accessibility Recommendations to produce MDD mobile applications

The proposed approach, **MAMA** (Figure 14), was presented providing details about the steps, responsible persons and artifacts used during an MDD process for developing accessible applications. We have also described the main rationale under each of the models and their extensions for addressing the necessary accessibility elements to produce the accessible applications.

Since MDD is a strategy that allows developers and stakeholders to communicate at different levels of abstraction, it is relevant to provide an overview of how to make **MAMA** feasible. We decided to focus our target on mobile application development, and we followed AccessMDD (DIAS, 2021), an MDD approach to stimulate Android developers' awareness of accessibility recommendations, focusing on blind and low vision users. Figure 20 shows an overview of AccessMDD.

In Figure 20 we can see a person (User) using a mobile application, and he/she interacts with UI components that can present potential accessibility problems. These problems are generally avoided by implementing technical accessibility recommendations, allowing the generated component to provide the appropriate accessibility resources for the user. To achieve this, a DSL designer, aware of the recommendations that can be applied to the interface components, implements the techniques in the respective integration points identified with the stages of the MDD process. Finally, these implementations allow a textsfModeler to follow the considerations suggested by the model editor and use the resources available in the language to generate a new mobile application with an accessible interface.

Figure 20 – Overview of AccessMDD approach



Source: Dias (2021)

In terms of feasibility, AccessMDD provides mobile application recommendations for the thirteen main accessibility problems that blind and low vision users face when interacting with mobile applications. The approach indicates 25 (twenty-five) accessibility recommendations, called mobile accessibility Recommendations (maRs), based on guidelines and documentations such as WCAG 2.1 (W3C, 2018b), BBC mobile (BBC, 2020), and GuAMA (SIDI, 2021; SIDI, 2019). To exemplify, the following is a brief description of maR13.

- maR13 - Navigation with ordered and visible focus.
  The application must support focus-based navigation, always visible, highlighting the focused component. It is important to ensure that keyboard focus navigation does not encounter barriers in the application, allowing all elements to be accessed without difficulty (W3C, 2018b; BBC, 2020; SIDI, 2021; Google LLC, 2021b).
  It helps to avoid: $\mathbf{P}_{11}$ - Confusing interaction and/or navigation.

Developers should integrate the recommendations (maRs) into the MDD process to produce accessible components. Table 11 shows the thirteen problems commonly faced by blind and low vision users, associated with the corresponding maRs compiled to avoid such problems.

Table 11 – Accessibility Problems ($\mathbf{P}_i$) related to the **maRs**

	Problem	Recommendation
$\mathbf{P}_1$	Insufficient text / image contrast Ballantyne *et al.* (2018), Acosta-Vargas *et al.* (2020), Alshayban, Ahmed and Malek (2020)	**maR07** | Color contrast
$\mathbf{P}_2$	Inappropriate touch target size Ballantyne *et al.* (2018), Acosta-Vargas *et al.* (2020), Alshayban, Ahmed and Malek (2020), Damaceno, Braga and Mena-Chalco (2018)	**maR17** | Touch target size **maR18** | Spacing
$\mathbf{P}_3$	Component missing label Carvalho *et al.* (2018a), Ballantyne *et al.* (2018), Alshayban, Ahmed and Malek (2020), Damaceno, Braga and Mena-Chalco (2018)	**maR10** | Component label
$\mathbf{P}_4$	Inadequate or redundant element description Acosta-Vargas *et al.* (2020), Alshayban, Ahmed and Malek (2020), Damaceno, Braga and Mena-Chalco (2018)	**maR24** | Contextual help
$\mathbf{P}_5$	Inadequate or unavailable feedback Carvalho *et al.* (2018a), Ballantyne *et al.* (2018), Damaceno, Braga and Mena-Chalco (2018)	**maR09** | Screen events **maR19** | Action Feedback **maR23** | Confirmation **maR25** | Error report
$\mathbf{P}_6$	Images / icons without textual alternative Carvalho *et al.* (2018a), Ballantyne *et al.* (2018)	**maR01** | Non-textual content
$\mathbf{P}_7$	Problems / Assistive Technology limitation Carvalho *et al.* (2018a), Ballantyne *et al.* (2018), Damaceno, Braga and Mena-Chalco (2018)	**maR06** | Resize content **maR22** | Assistive technology
$\mathbf{P}_8$	UI related elements non-grouped Alshayban, Ahmed and Malek (2020)	**maR15** | Grouping content
$\mathbf{P}_9$	Non-existent functionality Alshayban, Ahmed and Malek (2020)	**maR12** | Actionability
$\mathbf{P}_{10}$	Inconsistent or confusing content organisation Carvalho *et al.* (2018a)	**maR05** | Extensive content **maR08** | Paragraphs **maR20** | Data format
$\mathbf{P}_{11}$	Confusing interaction and / or navigation Carvalho *et al.* (2018a), Damaceno, Braga and Mena-Chalco (2018)	**maR04** | Visualization structure **maR13** | Navigation with ordered and visible focus **maR14** | Title of page **maR16** | App navigation
$\mathbf{P}_{12}$	Inadequate default presentation Carvalho *et al.* (2018a)	**maR02** | Content orientation **maR03** | Sensorial characteristics **maR11** | Keyboard mode
$\mathbf{P}_{13}$	Form structure with a confusing layout Ballantyne *et al.* (2018)	**maR21** | Form structure

Adapted from: Dias (2021)

AccessMDD indicates three Integration Points (IPs) to provide mobile accessible UI components which can be produced by adopting a MDD approach:

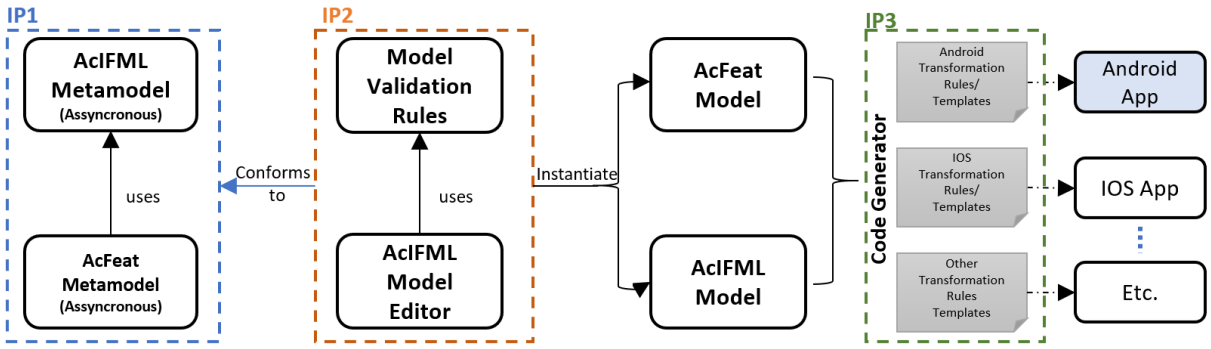
- **IP1** - Model (DSL): to include accessibility properties on the modeling language, providing

adequate accessibility support. For example, to define additional attributes, such as an "`alt`" attribute for inserting a textual alternative in an image

- **IP2** - Model Editor: enhance the model editor to provide warnings and hints to the modeler to remember or consider a specific accessibility attribute in a component. For example, in an image element, the editor can remind the modeler to provide an alternative text by displaying a warning

- **IP3** - Code Generator: transformation rules should map the abstract accessible components and properties, supporting and complementing the accessibility requirements and recommendations,to the corresponding components and accessibility properties of the target platform. For example, on the Android platform, the association of a form input component (`InputText`) with its respective label mapped to `TextView` and `EditText` components with the Android properties relevant for accessibility: `labelFor`, `inputType`.

Figure 21 – **MAMA** integration points for accessibility



Source: Elaborated by the author.

Figure 21 shows how we have used the AccessMDD integration points on the MAMA approach. We contemplate the **IP1** with the proposed `AcIFML` and `AcFeat` metamodels, the **IP2** with the provided properties and validation rules to guide the modeler on the model editor, and the **IP3** on the transformations rules used on our proof-of-concept target in Android mobile apps. Table 12 lists the maRs associated with the corresponding **IPs** the language developer has to consider to produce accessible mobile components according to MAMA artifacts.

The IFML language provides a visual syntax with graphical notations to represent elements, as we illustrated on Section 2.3. To provide tool support to the modeler during the project modeling phase of our approach, we extended the open source IFML editor[3]. The IFML editor is an Eclipse plugin based on Eclipse Modeling Framework (EMF)[4] and Sirius[5].

[3] IFML editor - <https://ifml.github.io/>
[4] EMF - <https://www.eclipse.org/modeling/emf/>
[5] Sirius - <https://www.eclipse.org/sirius/>

Table 12 – AccessMDD maRs and integration points on the **MAMA**

Mobile Accessibility Recommendations	**AcIFML** (IP1)	**AcFeat** (IP1)	**Model Editor** Validation (IP2)	**Transformation** (IP3)
– maR01 Non-textual content	✔			
– maR02 Content orientation	✔			
– maR03 Sensorial characteristics	✔			
– maR04 Visualization structure	✔			
– maR05 Extensive content			✔	
– maR06 Resize content				✔
– maR07 Color contrast		✔	✔	✔
– maR08 Paragraphs				✔
– maR09 Screen events	✔			
– maR10 Component label	✔		✔	✔
– maR11 Keyboard mode	✔			✔
– maR12 Actionability	✔	✔		
– maR13 Navigation with ordered and visible focus	✔			
– maR14 Title of page	✔		✔	
– maR15 Grouping content	✔			
– maR16 App navigation	✔			✔
– maR17 Touch target size		✔		
– maR18 Spacing		✔		✔
– maR19 Action Feedback	✔			✔
– maR20 Data format	✔		✔	
– maR21 Form structure	✔			
– maR22 Assistive technology	✔			
– maR23 Confirmation	✔			✔
– maR24 Contextual help	✔			
– maR25 Error report	✔			✔

Source: Elaborated by the author.

We have incorporated all of the extended `AcIFML` concepts into the editor, as well as the **IP2** maRs, to provide early information, warnings, and indications of potential accessibility modeling errors. For the **MAMA** approach, the modeler also relies on the tooling support to be guided, as needed, to provide an appropriate model with the necessary information for the transformation framework. We demonstrate these features, the **IP2** integration, and the inserted components in the model editor at Chapter 5.

4.5. Final Remarks

This chapter described our proposal for a model-driven approach to generate accessible applications whose focus relies on the front-end of the application minding accessibility called **MAMA**.

Initially, we outline the proposed approach to align with previously formulated research questions. Thus, we present a scheme representing the steps that constitute the approach (Figure 14). Next, we describe the extended metamodels (`AcIFML` and `AcFeat`) to provide the necessary elements to meet the accessibility requirements modeling when generating applications. And at the end, aiming to show the proposal's feasibility, we delimited the scope for mobile

apps and adopting MAMA. We discussed the integration of accessibility recommendations for mobile apps in the generation of accessible mobile apps.

Beyond, we can now present an update to Table 3 on Table 13. This was made possible because of our contributions to the components, in both IFML and MD^2, which have been improved with the inclusion of **accessibility features**.

Table 13 – Update of the UIDLs reviewed Bouraoui and Gharbi (2019)

Project	MDE/MBUI approach	Availability	Multi-platform/ Multi-device	Accessibility features	Code generation	Design/ Implementation/ Runtime
Context Toolkit Salber, Dey and Abowd (1999)	–	✔	Multi-platform Multi-device	–	No	Runtime
Cortex Duran-Limon *et al.* (2003)	–	–	Multi-platform Multi-device	–	No	Runtime
Compose Doukas and Antonelli (2013)	–	✔	Mobile	–	Partly	Implementation/ Runtime
UIML Helms *et al.* (2009)	✔	✔	Multi-platform	NDP [a]	Yes	Design
UsiXml Limbourg *et al.* (2005)	✔	–	Multi-platform	NDP [a]	Yes	Design
MD^2 Heitkötter, Majchrzak and Kuchen (2013a)	✔	✔	Mobile (Android/iOS)	✔	Yes	Design/Runtime
MariaXML Paterno', Santoro and Spano (2009)	✔	✔	Multi-platform Multi-device	CBA [b]	Yes	Design
IFML OMG (2015a)	✔	✔	Multi-platform Multi-device	✔	Yes	Design/Implementation
Supple Gajos, Weld and Wobbrock (2010)	–	–	Multi-platform Multi-device	Motor impairment	Yes	Design/Runtime
Transport App Krainz, Feiner and Fruhmann (2016)	✔	–	Mobile (Android)	Visual impairment	Partly	Implementation
Android XML Morris (2011)	–	✔	Mobile (Android)	✔	UI	Implementation
IP MM/AccUI MM/ EP MM Bouraoui and Gharbi (2019)	✔	✔	Multi-platform Multi-device	✔	Yes (Markup) Partly (PL)[c]	Design

✔ - Does correspond
– - Does not correspond
[a] [1] NDP: Not developed for this purpose
[b] [1] CBA: Could be applied
[c] [1] PL: Programming Languages

(BOURAOUI; GHARBI, 2019) Updated

Source: Elaborated by the author.

CHAPTER

5

PROOF OF CONCEPT

A Proof of Concept (PoC) is an exercise we use to test whether we can turn an idea into concrete reality. A proof of concept is used to determine an idea's feasibility or verify that the idea works as envisioned. It is sometimes also called a proof of principle.

In the **MAMA** approach presented in the previous chapter, the modeler counts with tool support to be guided as needed to provide a suitable model with the necessary information for the transformation framework. This chapter shows these features, the **IP2** integration, **IP3** integration, and the application of the inserted components by the model editor. We will use a PoC as an experiment that illustrates step-by-step the relevant elements of MAMA to demonstrate feasibility. The following sections describe an app prototype and its generation based on MAMA.

5.1. My Books Example

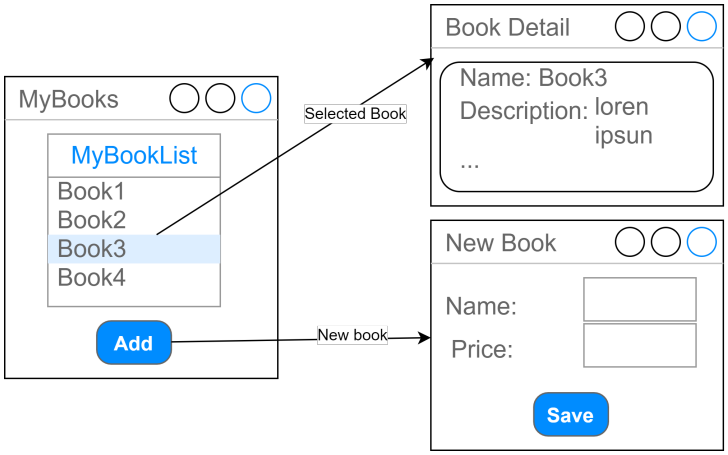
To demonstrate the applicability and expressiveness of the proposed approach, in this section we illustrate the MAMA process using a Proof of Concept (PoC). As a PoC, we developed a simple application that uses basic accessible abstract components and can generate multiple accessibility problems. We named the application *My books app*, Figure 22 shows a simple wireframe illustrating the application UI.

Despite simple interactions, we may encounter several of the problems described by Dias (2021), and others, in *My books app*, such as:

- $\mathbf{P}_2$ Inappropriate touch target size
- $\mathbf{P}_3$ Component missing labels
- $\mathbf{P}_5$ Inadequate or unavailable feedback
- $\mathbf{P}_8$ UI related elements non-grouped
- UI trap where the user cannot access the outside button to create a new book

Notwithstanding a simple application, this type of UI often encounters accessibility

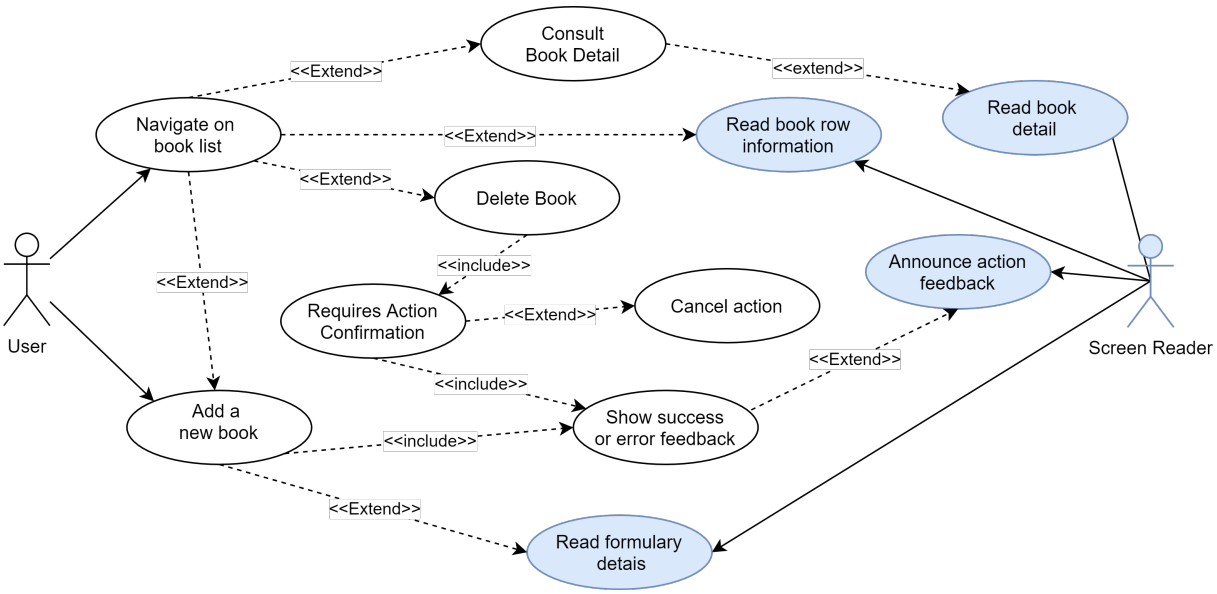
Figure 22 – Wireframe Fragment For *"My books App"*



Source: Elaborated by the author.

interaction problems. Figure 23 shows a simplified use case for this type of application. In *My books app*, books are in a list, and the application allows the user to interact with the list by navigating through the registered books, adding new books to the list, deleting books, and requesting detailed information about a book.

Figure 23 – "*My Books App*" Use Case

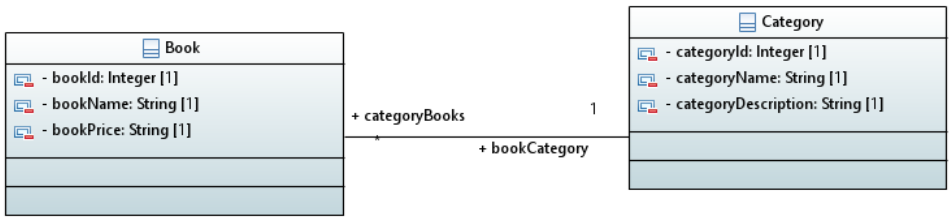


Source: Elaborated by the author.

Instead of coding, modelers must design graphical models to implement a complete application using MAMA. Before starting with the AUI model, the modeler must create at least one UML class diagram representing the intended domain model. For this example, a simple domain is shown on Figure 24. The model contains two classes `Book` and `Categories` related by an association N:1. This UML model will be used in the first transformation performed on MAMA. This model is then used as input for modeling with the AcIFML extension mapped to

the DomainModel concept through the model editor on EMF ModelWizzard, provided by the metamodel generator. The objects from the UML model are mapped to `AcIFML` using Java to accomplish the `DomainModel` with `DomainConcepts` and `DomainFeatures`.

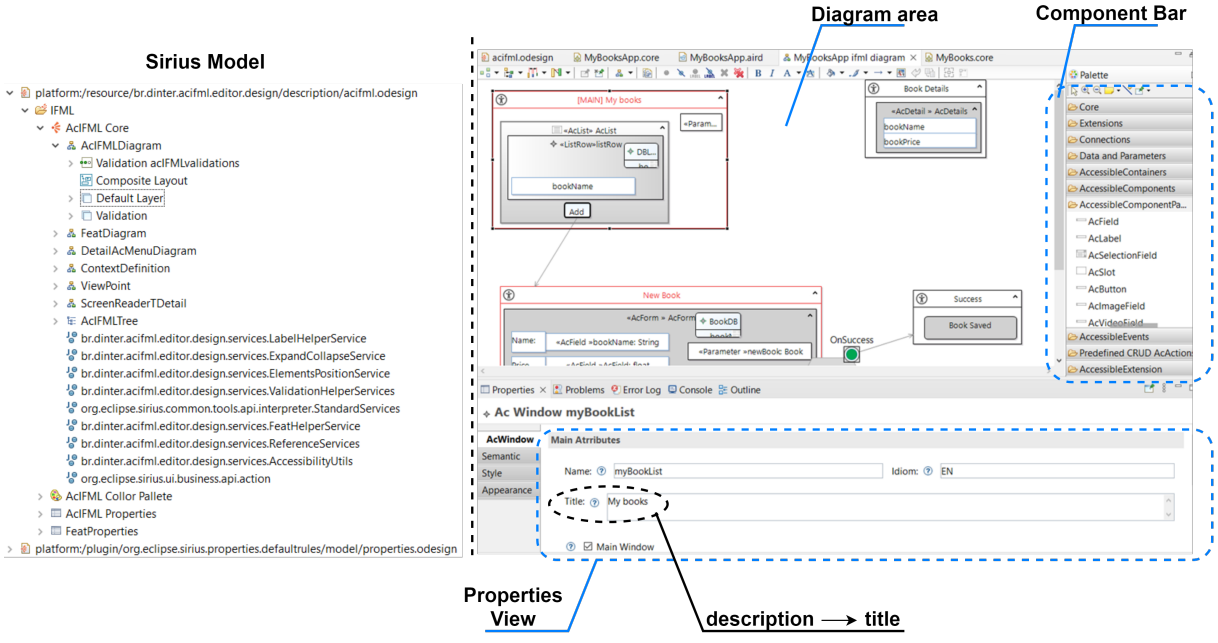
Figure 24 – Simple UML domain for *"My books App"*



Source: Elaborated by the author.

Once the domain model is completed, the modeler can begin the AUI design. The modeler must initiate an EMF model project on Eclipse, including the ".uml" domain file in the project, and create a ".core" file (`IFML` extension). At the time of creation, the model requests the setting of the ".uml" file and performs an M2M transformation using the domain information. It then opens the graphical Model Editor and enables the AUI design. Figure 25 presents the Sirius model definition file for the project on the left side, with the defined diagrams, layers, and properties reflected on the right side in the diagram area (main screen) of the modeling tool and the component bar. Modelers must graphically define the UI application elements, user interactions, and effects.

Figure 25 – **IFML** extended editor for `AcIFML` components



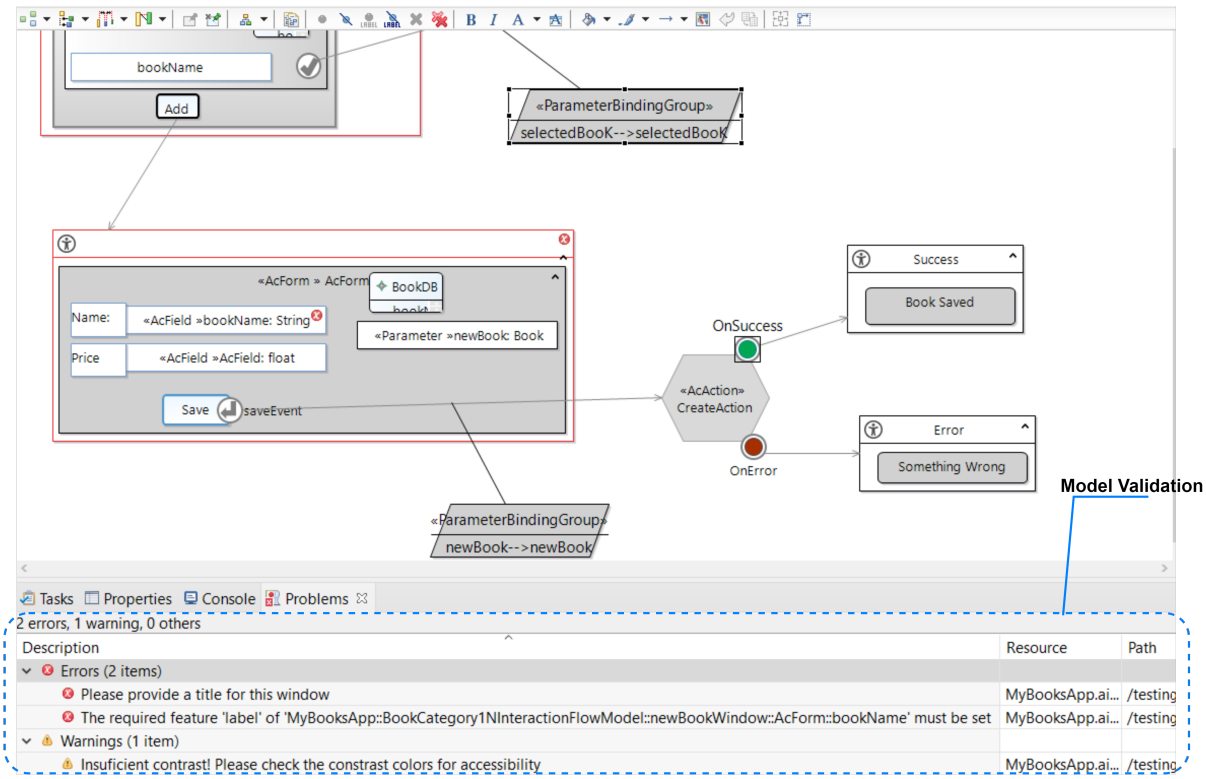
Source: Elaborated by the author.

In addition, modelers can define possible Contexts and their concepts (Figure 18). If needed, they can define ContextDimensions to model the UI and interactions for assistive technologies, user abilities, preferences defined in the extension, user role, screen position, and settings for a particular device. These contexts are related to predicted context variables used in runtime customization, which can be associated with new ViewPoints for the application.

To guide the modeler through the AUI modeling process, we have integrated IP2s maRs and also developed validation rules to ensure that the required information is propagated to the transformation framework. The modeler will be guided from the beginning when inserting components via the property editor, and during model validation. An example can also be found in Figure 25 where we have described the **description** property of an AUI `AcWindow` as **title** to make it easier to understanding and also with validation to prevent the property from not being filled in.

Figure 26 illustrates on the bottom some of the IP2 integrated into the model editor to provide early validation and avoid the aforementioned problems. We see two error messages (one for a window with no title, and one for a missing label on an input) and one warning (for low contrast in the `AcFeat` model color palette). These messages are provided for the modeler when he requests validation of the model.

Figure 26 – Example of accessibility early validation to the modeler

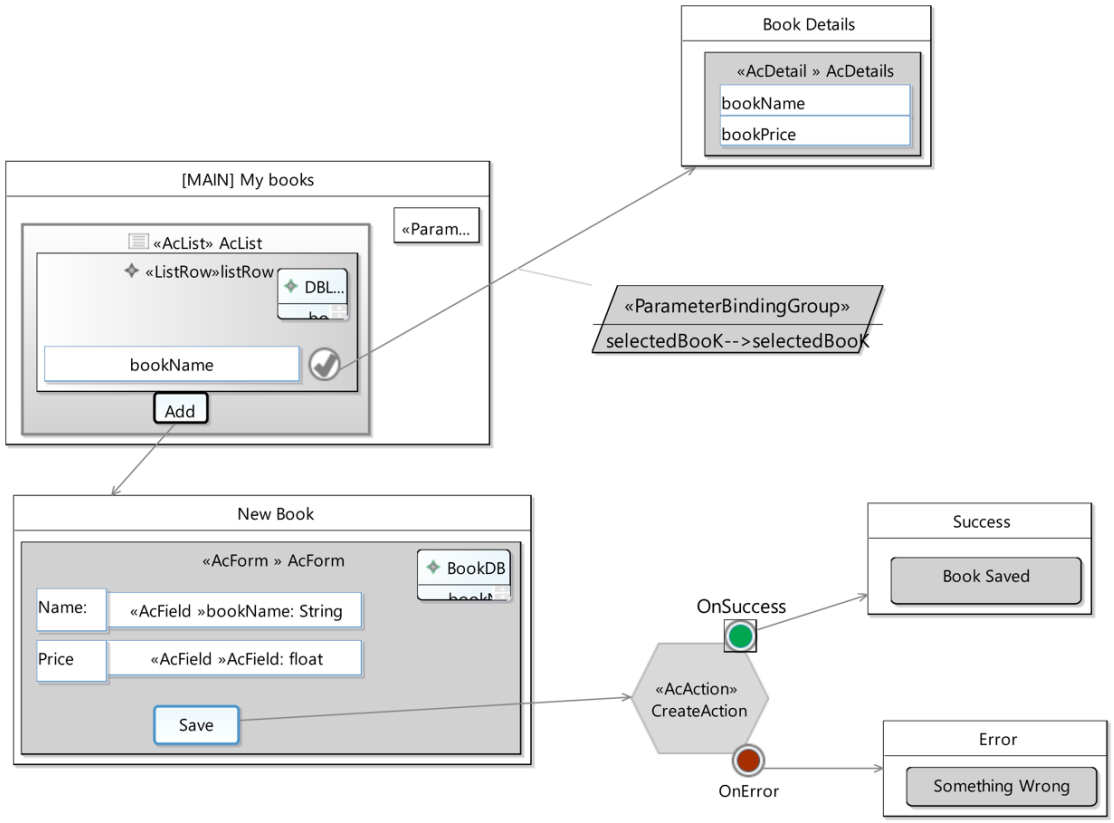


Source: Elaborated by the author.

Figure 27 presents a possible AUI modeling using the `AcIFML` extended components. The

components we have added are: `AcWindow`, `AcList` and `AcListRow`, `Databinding`, `VisualizationAttributes`, `IFMLParameters`, `AcLabel`, `AcField`, `AcDetails`, `AcCreateAction`, `ActionEvent` and `AcFeedback`.

Figure 27 – Possible AUI model with `AcIFML` components



Source: Elaborated by the author.

Another implementation in the model editor was the definition of predefined actions. Since we were not focused on the model business, one strategy was to define predefined "Create Read Update and Delete (CRUD)" actions that modelers could use for basic CRUD operations on the model. For example, when the modeler inserts an `AcCreateAction`, it inserts an `Action` with two `ActionEvents` *OnSuccess* and *OnFail* associated with success and failure messages `AcFeedback` to provide feedback to the user in at least these two cases ensuring action feedback modeling.

After validating the model, the modeler can request the M2M transformation to generate the code for the designed application using the transformation framework.

5.2. Model Transformation Framework

All abstract components that we have developed are integrated into the `AcIFML` editor. So, we can produce an abstract `AcIFML` model with any combination of them. However, since

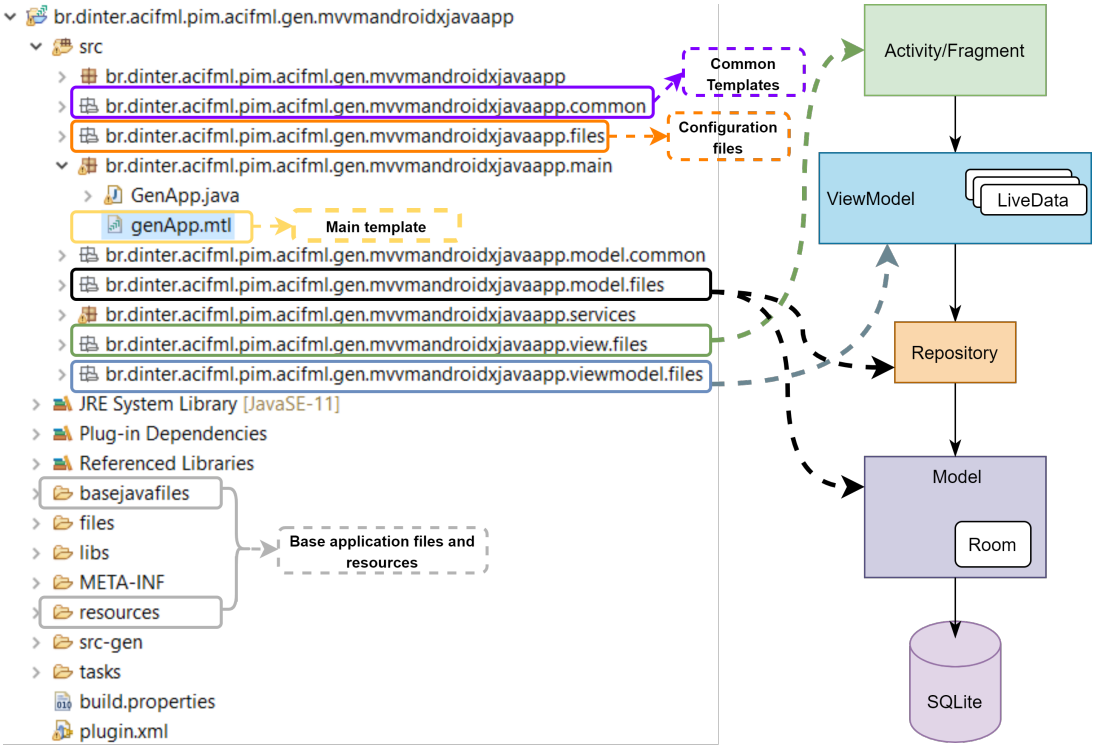
we focus on generating mobile accessible applications code, we have to provide the whole infrastructure for the generated application.Not all are integrated on the transformation framework set yet. In this section, we demonstrate the MAMA transformation framework and show some of the IP3s maRs we have integrated to meet accessibility requirements.

Regarding the generation of mobile applications on MAMA, we developed our transformation framework using Acceleo. It is an open-source template-based technology for creating custom code generators that implements OMG's MOF Model to Text Language (MTL) and use any EMF-based models. The idea is to generate mobile native accessible applications. As a first target, we decided to generate Android applications using Java language. For the architecture of the generated application, we have chosen the architecture pattern "Model View ViewModel (MVVM)". MVVM aims to facilitate the separation of graphical user interface development from back-end logic or business logic. Android provides a set of libraries and components for the MVVM architecture according to best design and implementation practices and provides good accessibility support and documentation. Despite separating the concerns of the architecture to test accessibility requirements, we implemented all architecture layers, but only SQLite local storage.

Figure 28 shows its structure for the Android generator using Java language. The framework is organized to reflect the MVVM architecture and Acceleo best practices recommendations[1]. At the bottom are the resources used to generate the application, such as the Java base files and the `.xml` resources. The prefix used for the packages is "br.dinte.acifm.pim.gen.mvvmandroid-xjavaapp", on ".viewmodel.files" are the templates to generate the viewmodel files; ".view.files" the templates to generate the views and view content (classes and .xml); ".model.files" the templates to generate the model (class and Data Access Object (DAO)) and the repository; ".files" the project configuration files and values like AndroidManifest, Graddle scripts, id.xml and strings.xml; ".common" the templates and queries common to all packages; ".services" the Java services wrapped by the queries.

[1] <https://www.obeosoft.com/en/acceleo-best-practices>

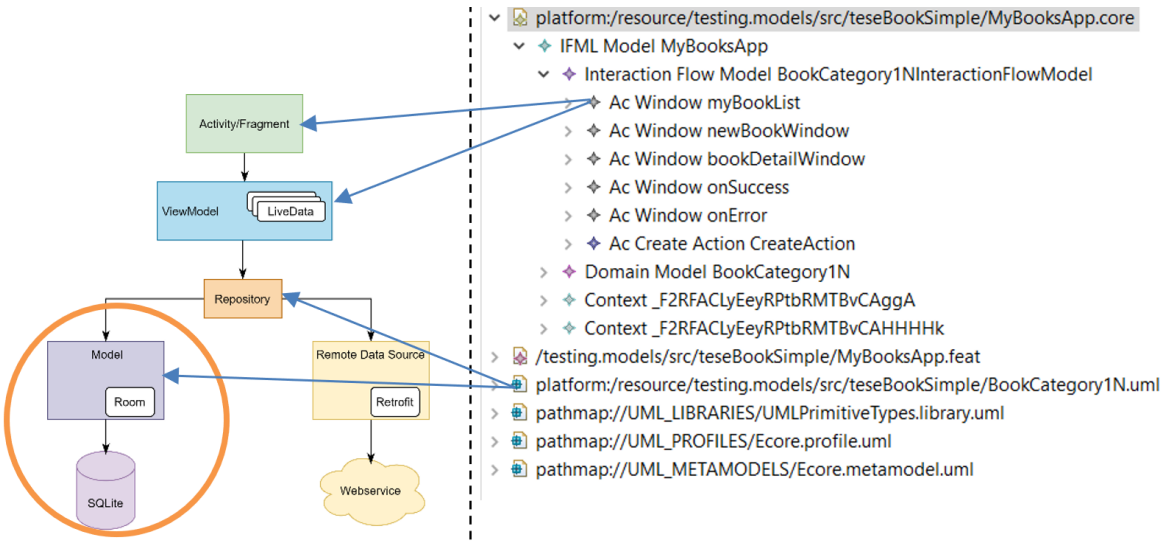
Figure 28 – Transformation Framework - From AcIFML to Android MVVM



Source: Elaborated by the author.

Figure 29 shows the Android MVVM architecture in terms of the first level components of the modeled *My books app*. The first level `ViewContainers` were mapped to Activities and also a ViewModel for basic CRUD operations. The Repository, and Model layer is generated based on the UML domain model, which remains proxy along with the `AcIFML` model. We have defined only local storage for the model with the Room database. We used the Data Binding library from Android Jetpack to allow the bind UI between components of the layouts to data sources.

Figure 29 – First Level mapping MVVM architecture **AcIFML** components



Source: Elaborated by the author.

Source code 1 shows a fragment of the template where we can demonstrate our strategy for generating part of the View layer that refers to the .xml file of Activity. All templates use almost the same strategy with a generic function defined for each component, as we can see on line 53 `content.genXMLNode`. For each `ViewElement`, `Event`, or `Action` that needs a representation on the .xml file we have to write a specific *genXMLNode*, according to its general component relation of the language or let the component use the general class function. The same approach is used for the other base files on the framework. So if the developer wants to add new widgets, he or she must write its corresponding template for each part of the application (Model, View, and ViewModel).

Source code 1 – Fragment Activity Template from ViewContainers (`AcWindow`) to Activities

```
21: [template public genLayoutFile(aViewContainer : ViewContainer)]
22: [file ('src-gen/'+aViewContainer.layoutPath()+aViewContainer.layoutName().concat(

49: [for (content : ViewElement | aViewContainer.viewElements->excluding(DataBinding))]
50:   [content.genXMLNode()/]
51:   [if (content.oclIsTypeOf(ViewComponent))]
52:     [for (vcp : ViewComponentPart | content.oclAsType(ViewComponent).
      viewComponentParts)]
53:       [content.genXMLNode()/]
54:     [/for]
55:   [/if]
56: [/for]
```

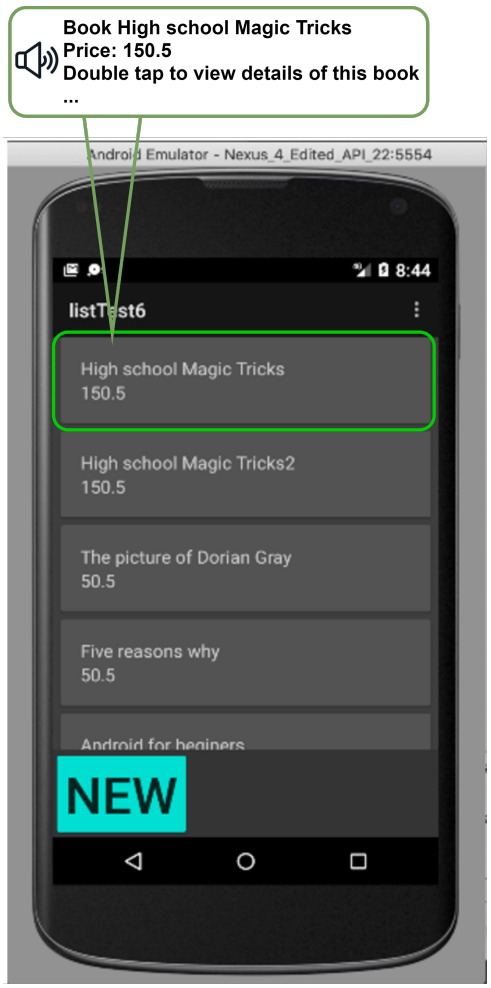
The main screen of the application show the list of books. This is a common layout on mobile applications and can be tricky for screen readers users, since they can be trapped on the list and never get access to the external button. To overcome this problem during the design the modeler received an error message being guided to fix this issue providing a `KeyboardNav`

specific for the AcButton on the screen providing an alternative navigation flow, fixing the trap problem before the code generation (Figure 30).

Figure 30 – Generated Main Screen of *My books app* - Accessible List Example

```
1: //accessibility presonalized
       announcements for select events of
       list rows
2: if (AcHelper.isAccEnabled([acList.
       listHolderAdapterName().
       toLowerFirst()//].[acList.
       listItemBindingName().toLowerFirst
       ()/].getRoot().getContext())){
3:   CustomAnnouncements
       [acList.listAdapterName().
       toLowerFirst().concat('Announcement
       ')/] = new CustomAnnouncements();
4:   [for (event : Event | acList.
       eAllContents(AcOnSelectEvent))]
5:     [if (not event.oclAsType(
       AcOnSelectEvent).description.
       oclIsUndefined())]
6:       [acList.listItemAnnouncemetObj()
       /].setTextToClick([acList.
       listHolderAdapterName().
       toLowerFirst()//].[acList.
       listItemBindingName().toLowerFirst
       ()/].getRoot().getContext().
       getResources().getString(R.string
       .[event.oclAsType(AcOnSelectEvent).
       idName().prefix('description_')/]));
7:       ViewCompat.
       setAccessibilityDelegate([acList.
       listHolderAdapterName().
       toLowerFirst()//].[acList.
       listItemBindingName().toLowerFirst
       ()/].[acList.
       listItemBindingRowContainer()/],
8:         [acList.listItemAnnouncemetObj
       ()/]);
9:     [/if]
10:
11:  [/for]
12: }
```

(a) Template fragment – Accessible List Adapter



(b) Generated - Accessible List

Source: Elaborated by the author.

Another IP3 improvement for accessibility is the grouping of AcListRow contents. The AcLabels added to the model have been mapped to the Android TextView component. The properties android:focusable android:focusableInTouchMode were both given the vale false for the book name and price. Only the AcListRow mapped to the CUI Android CardView component receives focus, allowing the screen reader to read it in a single speech flow (see Figure 30b), and also for the understand the line content. This is useful regardless of the content of the AcListRow.

Figure 30a shows a fragment of the template for the generating the personalized list announcements using the field descriptions. Line 2 shows that this considers whether the AT screen reader is enabled in the context. Line 6 shows the announcement if there are click events associated to the row.

We can see the .genXMLNode template fragment for `AcField` on Source code 2. For this mapping, we used a TextInputLayout from Android Material Design, with the options of using a floating label preserving screen accessibility, or using a fixed label next to the input. In line 11, we see the call to the labelFor specific node for this field. Line 16 shows a fixed IP3 recommendation that defines a minimum margin value of 16dp. Line 18 shows the placeholder attribute, which is a string that comes from the input description that we mapped with a prefix hint. Line 55 shows another IP3 implementation that comes from IP1 and IP2. This is the implementation of the keyboard mode by the modeled input type (text, decimal, email and others that exist on Android, for example).

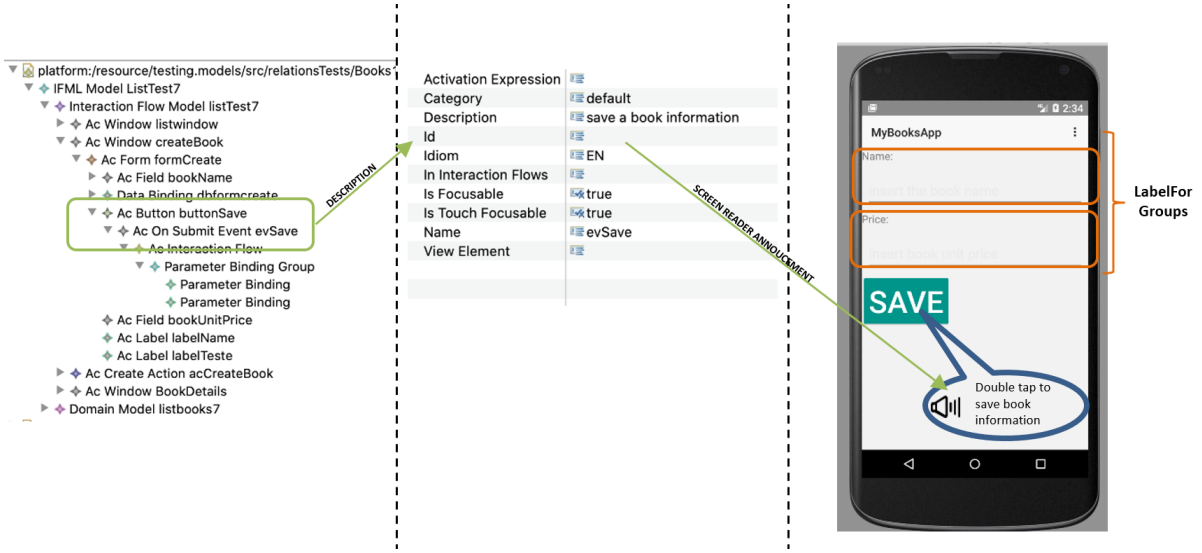
Source code 2 – Fragment genXMLNode for the widget `AcField` mapped to TextInputLayout

```
11: [widget.label.genXMLNodeLabelFor()/]
12: <com.google.android.material.textfield.TextInputLayout
13:     android:id="@+id/[widget.oclAsType(NamedElement).idName().concat('TextInputLayout
        ')/]"
14:     android:layout_width="match_parent"
15:     android:layout_height="wrap_content"
16:   android:layout_margin="16dp"
17: [comment hint will be mapped to label in other layout option -> Label isFloat=true /]
18:   app:placeholderText="@string/[widget.oclAsType(NamedElement).idName().prefix('hint_
        ')/]" >
19: <com.google.android.material.textfield.TextInputEditText
20:     android:layout_width="match_parent"
21:     android:layout_height="wrap_content"
22:   android:id="@+id/[widget.oclAsType(NamedElement).idName()/]"
23:   [comment if is part of a list individual objects must not be focusable, but the
        group on the row or detail/]
24:   [if (widget.ancestors(AcList)->size()>0)]
25:       android:focusable="false"
26:       android:focusableInTouchMode="false"

55:       android:inputType="[widget.fileldType.filterInputFieldType()/]"
56:     style="?textinputStyle"
57:     [widget.genXMLNav()/]
```

Figure 31 shows a fragment of the model for the `AcButton` component and the modeling of the `AcOnSubmitEvent` for the button. On the center we can see the description provided that will be announced by the screen reader, on the right side the generated screen for the *New Book* screen of the `AcIFML` model with the button and the related labelFor with the corresponding input demonstrating maR09, maR10, maR12, also with the implementation of maR19, maR11, and maR15 for the components on the mentioned screen.

Figure 31 – Generated UI *Create Book* - Event Description



Source: Elaborated by the author.

5.3. Final Remarks

This chapter demonstrates the applicability of the proposed approach **MAMA** for the generation of accessible applications by using an example application. First, we presented a simple application and some of the possible accessibility issues that can arise during interactions with the application when there are no accessibility concerns.

We presented our extension of the IFML model editor for `AcIFML` to provide tool support to modelers. We illustrated how the AccessMDD IP2s recommendations were used in the model editor to help modelers with the required information and actuate with the model aiming the propagation of the required accessibility features.

Finally, we presented our model transformation framework for MVVM Java Android applications that generates a complete MVVM application, exemplifying the implementations of recommendations with the transformation templates and the final application with accessibility in mind.

The MAMA approach allows the modeler to realize the design of applications with graphical components provided by the AcIFML extension in conjunction with AcFeat and incorporate accessibility requirements early in the development process. The approach guides the modeler and provides early validation of models. This way, we can avoid the mentioned accessibility problems and generate accessible mobile applications. Since the limitations arose from the previously planned route regarding the methodology for validation of the approach, we submitted the generated applications to automatic evaluation tools such as Android Accessibility Scanner, which did not return any accessibility problems.

CHAPTER

6

CONCLUSIONS

Accessibility is a requirement for any user to interact with software. Many applications still lack accessibility, and people with disabilities may find barriers when interacting with them. Given the importance of accessibility in applications, the developers of these apps must be aware of it and try to meet this requirement.

On the other hand, among the most important benefits of using model-driven approaches, we highlight the increase in productivity made possible from implementations of code transformations and between models, which optimize development time. By automating transformations, the same model can be transformed into code for different platforms and devices, avoiding repetitive tasks and reducing developer effort (LUCRÉDIO, 2009).

Due to the relevance of studies related to accessibility, as well as the lack of support favoring the adoption of approaches at a level of abstraction closer to user interfaces and interactions (which are subject to constant technological updates, such as frameworks and languages), this thesis proposes a model-driven approach that allows the design and generation of accessible applications from the beginning.

The main goal of this thesis was defined:

> ***To develop a model-driven approach for generating accessible applications that consider accessibility requirements early in the development process***.

The objective of this chapter is to summarize the achievements of the performed research activities pursuing the goal of this thesis.

6.1. Answering the Research Questions

There are two Research Questions, defined in this thesis, to boost our studies on accessibility issues, allowing the abstraction of concepts related to general accessibility barriers and to appropriately classify and integrate the recommendations and solutions at each level of abstraction of the MDD process development strategy. They are the following:

RQ1. *– How can we support the modeling of user interfaces and interactions in applications to create more accessible systems using a Model-driven approach?*

RQ2. *– How to abstract early on the model of user interface interaction flow to produce accessible applications?*

We have answered these two **RQs.**, and we could reach the goal of this thesis. The contributions of this work came with the development of the artifacts showing the constructiveness of the artifacts to those answers, beyond the knowledge to design rationale.

The answer to **RQ1.** is by proposing an approach, described in Section 4.1, `MAMA` (Figure 14). It aims to integrate accessibility recommendations and guidelines from the beginning of model-driven development by supporting this modeling in the UI language and providing tools to guide the modeling process.

We also have answered the **RQ2.** by abstracting the UI's accessibility requirements and modeling them as extensions of the IFML language. In Section 4.2, we presented the extended concepts on the language to accommodate the accessibility recommendations and guidelines that could be inserted at the AUI level.

6.2. Thesis Contributions

During the search for answers to the Research Questions and pursuing to accomplish the primary goal, we performed many activities that have increased the knowledge for the elaboration of the `MAMA` approach proposed in this thesis. `MAMA` is the first contribution with an architecture that considers accessibility requirements early in the development process. To avoid problems that blind and low vision users often face, we integrated accessibility recommendations into three points (metamodels, model editor, and transformation framework) following the `AccessMDD` approach. The contributions emerged in the artifacts generated for the architecture, with several IT constructs contributing to the design, implementation, and awareness of accessibility in applications for developers.

We have accommodated accessibility guidelines into the IFML language by providing an extension called `AcIFML` that supports the modeler with an accessibility-ready language with accessible abstract components. We also developed the AcFeat metamodel, which includes the

accessibility recommendations at an abstract level for the presentation layer inspired by Material Design.

Another contribution concerns graphical tool support with an environment that guides the modeler during the Project Modeling phase with early validation. In this way, the modeler must fill in the accessibility relevant information in the abstract model needed to generate concrete accessible components before generating the concrete application.

6.3. Thesis Publications

During the search for answers to the Research Questions and pursuing to accomplish the primary goal, we performed many activities that increased knowledge for elaborating the `MAMA` approach proposed in this thesis. Thus, we briefly describe two additional contributions of this thesis, obtained during this process, as follows:

1. **Collaboration** to a project SPRINT (#2018/08195-9[1]), which was supported by FAPESP. This activity brought contributions to the investigations of this thesis in the sense of:
   - It resulted in a publication: Rieger *et al.* (2020)
   - We could align the Model-Driven Development cross-platform approach (MD^2), since we assume that implementing various accessibility requirements will be facilitated by abstracting technical recommendations through high-level model implementations. Thus, the developer (modeler) doesn't need to have accessibility technical knowledge
   - It allowed a partnership with researchers of the University of Münster-Germany
   - This work extended the MD^2 framework with abstract-level accessibility features and also integration on transformations and model editor. In this way, Table 3 in Section 3.3 can be updated with MD^2 as an approach to support accessibility
   - It was one of the reasons that led us to focus the scope of our proof-of-concept on accessible mobile applications, although our approach can be used in other contexts and on other platforms.

2. **Cooperation** with a Master Dissertation (DIAS, 2021), specially related to the `AccessMDD`. This activity brought contributions to the investigations of this thesis in the sense of:
   - It resulted in a publication: Dias, Duarte and Fortes (2021)
   - It allowed contributing with development of `AccessMDD`, mentioned in Section 4.4. We have cooperated directly with creation of this MDD solution (`AccessMDD`) to generate accessible mobile applications. It focused on blind and low vision users to direct the development of the solution, and appointed recommendations for accessibility in the user interface components.

[1] <https://fapesp.br/en/sprint/call12018>

- It activated an investigation of the main accessibility problems that affect visually impaired users. A set of thirteen main recurring problems ($\mathbf{P}_i$) was identified, in many types of applications, to which mobile application developers should pay special attention. Table 11 listed the thirteen $\mathbf{P}_i$, associated with the corresponding maRs, which were also defined.
- It was another reason that led us to focus the scope of our proof-of-concept on accessible mobile applications, although our approach can be used in other contexts and on other platforms.

6.4. Related Works

The approach proposed in this thesis is related to and influenced by several works in different dimensions, as shown in our review of state of the art (Section 3.2). It is important to remember that our focus was not on adaptation, re-engineering, or accessibility evaluation. Our research aims ***To develop a model-driven approach for generating accessible mobile applications that consider accessibility requirements early in the development process*** lies at the intersection of three dimensions: model-driven development, mobile applications, and accessibility requirements early in the development process. In this way, related work may provide a link between two of these domains, but rarely between all three (at the time of writing). Accordingly, we considered revisit and discuss in this section works that also address the intersection of these three dimensions, summarized in Table 14.

Table 14 – Close Related Works

PROPOSAL	Integration Points	Guidelines	Modeling Aspects	Abstract UI DSL	Presentation Modeling	Modeling Tool Support	Target	Modeller Support
Miñón *et al.* (2014)	Metamodel/DSL	WCAG 2.0 (2.4 Guideline)	(UI, Behavior, Context)	UsiXML	No	Yes	Generic XHTML	No
Krainz and Miesenberger (2017a)	Metamodel/DSL Transformation Rules	WCAG 2.0	Accapto XML (UI)	Front-end	No	No	Native Android	No
Bouraoui and Gharbi (2019)	Metamodel/DSL Transformation Rules	WCAG 2.0	Data, Context, Behavior, UI	AccUI/IP/ EP MM	Yes	Yes	Native iOS	No
MAMA(This Approach)	Metamodel/DSL Model Editor Transformation Rules	WCAG 2.1	Data, Context, Behavior, UI	AcIFML AcFeat	**Yes**	**Yes**	Native Android	**Yes**

Source: Elaborated by the author.

Miñón *et al.* (2014) consider the three main dimensions we considered in our approach but focuses only on integrating the WCAG 2.0 guideline related to navigational issues (Guideline 2.4). They used and UI Description Language (UIDL), defined by several metamodels such as UsiXML, to describe abstract elements of UI such as widgets, controls, containers, modalities, interaction techniques. The SPA4USXML tool was used to associate abstract tasks and the AUI model with user decisions to support the generation of multiple UI from the same task definition. This work suggests that accessibility requirements must be integrated at multiple points in the user interface development methodology to meet requirements accessibility. Although the work

focuses on the WCAG 2.0 2.4 guideline, some of the success criteria (Page Titled (2.4.2), Headings and Labels (2.4.6), and Section Headings (2.4.10)) were not included. In contrast, our proposed approach is broader and covers success criteria from the various WCAG 2.1 recommendations. The Miñón *et al.* (2014) approach generates generic XHTML code and is included as mobile generation due to its UsiXML nature.

Krainz and Miesenberger (2017a) use an XML-based DSL in which developers must define the UI components on each screen to generate accessible UIs for Android. In this approach, accessibility is essentially enriched at runtime with a library of accessibility patterns instantiated according to the profiles and platforms defined in the XML application definition. Accapto automatically generates many accessibility-related features for the native Android XML UI layer(KRAINZ; MIESENBERGER, 2017a). However, developers still need to add programmatic features manually, and the approach does not support developers in doing so, and the DSL is not integrated into a modeling tool.

The Bouraoui and Gharbi (2019) approach focuses on the graphical and adaptation aspect for the semi-automatic generation of user interfaces. Their approach generates XHTML user interfaces based on a set of models, such as AccUI, which describes UI's basic dialog and presentation elements that meet accessibility and ergonomic criteria. The approach only considers a single context parameter (the platform). The concepts are tightly coupled in the metamodels, making it challenging to model UI for different contexts and user profiles with the components available for the presentation aspects of the user interface. Unlike our approach uses different models to facilitate the modeling of UI customization for context, profiles, assistive technology, and user abilities.

Given the characteristics of metamodels used to support `MAMA`, we can consider UI content with user ability, navigation path, user interactions, and events, business logic (Create, Read, Update, Delete (CRUD) tasks), and binding to the persistence layer, and presentation. We used UML class diagrams, AcIFML, and AcFeat metamodels to cover these aspects. Unlike all these approaches, we generated a complete Android MVVC application regarding the transformation framework. In this way, accessibility requirements are included on all application layers (Model, View, and ViewModel), not only in the XML mobile UI code. We can include the necessary coding to implement the accessibility recommendations using the integration points.

6.5. Approach Limitations

During the development of this doctoral work, we followed scientific rigor. However, despite answering the research questions and all the contributions, limitations arose that require future investigation and that we could address as we continue our research.

As we can see on Table 15, we identified WCAG 2.1 guidelines' Success Criteria limitations in our `MAMA` approach. One limitation identified is that even though we provided

early validations for the models, there is still no control over the content entered by the modeler. The models are validated, so the information is there. However, the content of the information provided, on descriptions and labels, is not validated, leaving it up to the modeler to provide contextually and domain-relevant information that can be truly useful. For this reason, we could not guarantee Success Criteria such as 2.5.3, 3.1.3, and 3.1.4 in our approach. In the same context, other success criteria that depend on the content developed for the application, such as 1.2.4, 1.2.7, 2.3.1, and 2.3.2, cannot be guaranteed even though the content can be modeled and generated.

Some of WCAG's 2.1 Success Criteria were outside the scope developed in this version of the approach for the generated applications, such as 2.2.4, 2.2.5, 2.2.6, and 3.3.4. Some need extra validation on different contexts, new components, and different devices, such as 1.4.10 and 2.1.12 but are valid for the components included on the abstract layer and the transformation framework. Others we identified as technology-dependent for the mobile context, such as 2.1.3, 2.5.6, and 3.1.6, were not possible to be abstracted and specified in any model of our MAMA approach.

Another limitation concerns the disabilities addressed. We follow the `AccessMDD` approach, which focuses on solving the most common accessibility problems that blind and low vision users face. However, we know that several aspects considered by the approach also resolve other types of disabilities interaction problems. Future extensions of the approach should consider and test other accessibility problems and disabilities to increase the scope of the developed applications.

An additional limitation is that the transformation framework does not implement all the features provided in the abstract models. Transformation rules should be extended to include their accessible generation to increase the scope and flexibility of the generated applications.

From the previously planned route regarding the methodology for validation of the approach, we also hoped to validate the proposal from broader perspectives: usability testing and interviews with professional developers. Therefore, these factors stimulate the next prospects for future work to continue this research.

6.6. Directions for Future Studies

Initially, `MAMA` should be validated as well as evaluated for the level of developer satisfaction, that is, we hope to analyze whether, in practice, the approach facilitates the process of developing accessible apps. Rather than measuring the effectiveness of our approach in terms of productivity (one of the benefits of the MDD), we recommend gathering a collection of opinions from developers regarding the use of the proposed approach in comparison with traditional approaches, in particular, regarding the assistance offered and the implementation details automatically added.

Table 15 – MAMA WCAG 2.1 Identified Limitations

WCAG 2.1 Guideline		Success Criteria	Status on MAMA
1.2 Time Based Media	A	1.2.4 Captions (live)	PC
	AAA	1.2.7 Extended Audio Description (Prerecorded)	CD
		1.2.9 Audio-only (Live)	CD
1.4 Distinguishable	AA	1.4.5 Images of Text	CD
		1.4.10 Reflow	CC
	AAA	1.4.7 Low or No Background Audio	CC
		1.4.9 Images of Text (No Exception)	CD
2.1 Keyboard Accessible		2.1.2 No Keyboard Trap	CC
	AAA	2.1.3 Keyboard AAA (No Exception)	TD
2.2 Enough Time	A	2.2.1 Timing Adjustable	PC
		2.2.2 Pause, Stop, Hide	PC
	AAA	2.2.3 No Timing	NC
		2.2.4 Interruptions	NC
		2.2.5 Re-authenticating	NC
		2.2.6 Timeouts	NC
2.3 Seizures & Physical Reactions	A	2.3.1 Three Flashes or Below Threshold	CD
	AAA	2.3.2 Three Flashes	CD
		2.3.3 Animation from Interactions	NC
2.5 Input Modalities	A	2.5.1 Pointer Gestures	PC
		2.5.3 Label in Name	CD
		2.5.4 Motion Actuation	CD
	AAA	2.5.6 Concurrent Input Mechanisms	TD
3.1 Readable	AAA	3.1.3 Unusual Words	CD
		3.1.4 Abbreviations	CD
		3.1.5 Reading Level	CD
		3.1.6 Pronunciation	TD
3.2 Predictable	AA	3.2.4 Consistent Identification	CD
3.3 Input Assistance	AA	3.3.4 Error Prevention (Legal, Financial, Data)	NC

CC – Correspond only on the tested Context
CD – Content/Modeler Dependent
NC – Does Not Correspond
PC – Partially Correspond
TD – Target/Technology Dependent

Source: Elaborated by the author.

In MAMA we did not describe a critical step, which refers to testing activities. Undoubtedly, this step is essential for developers to have safe conditions to verify and validate what is under development using the proposed approach. We predict that adopting a model-oriented testing approach would be quite adequate to evolve the MAMA proposal.

In this thesis, the components and transformers were implemented, with the appropriate accessibility issues, and thus we made possible both the integration with a method whose recommendations had been specified for mobile apps and use by blind users, as well as integrating interface components for Android. In future works, more components should be added to the transformers, aiming to complete all functionalities to be offered in interactions by users.

6.7. Final Remarks

In this chapter, we presented again the research questions that oriented the activities performed in this study and discussed them relating to achieving the main goal. We have also provided a brief overview of each of the contributions based on additional findings of this work during collaboration and cooperation with other related projects and publications. We have also highlighted the closely related works and limitations identified on MAMA and suggested topics to be investigated and explored in future studies.

The main contribution of this thesis was the proposal of `MAMA` jointly with the artifacts that constitute it. This proposed approach allows the implementation of technical requirements for accessibility in MDD from the early phases of a software process. Subsequently, the feasibility of our solution will be available to evolving and serving interested developers and researchers.

BIBLIOGRAPHY

A11Y. *Orange web accessibility guidelines*. 2021. Available: <https://a11y-guidelines.orange.com/en/>. Citations on pages 39 and 145.

ABRAHãO, S.; BOURDELEAU, F.; CHENG, B.; KOKALY, S.; PAIGE, R.; STÖERRLE, H.; WHITTLE, J. User Experience for Model-Driven Engineering: Challenges and Future Directions. In: **2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)**. [S.l.: s.n.], 2017. p. 229–236. Citation on page 145.

ACERBIS, R.; BONGIO, A.; BRAMBILLA, M.; BUTTI, S.; CERI, S.; FRATERNALI, P. Web applications design and development with webml and webratio 5.0. In: PAIGE, R.; MEYER, B. (Ed.). **Objects, Components, Models and Patterns**. Springer Berlin Heidelberg, 2008, (Lecture Notes in Business Information Processing, v. 11). p. 392–411. ISBN 978-3-540-69823-4. Available: <http://dx.doi.org/10.1007/978-3-540-69824-1_22>. Citations on pages 42 and 144.

ACERBIS, R.; BONGIO, A.; BRAMBILLA, M.; BUTTI, S. Model-Driven Development of Cross-Platform Mobile Applications with Web Ratio and IFML. In: **2015 2nd ACM International Conference on Mobile Software Engineering and Systems**. [S.l.: s.n.], 2015. p. 170–171. Citation on page 143.

ACOSTA-VARGAS, P.; SALVADOR-ULLAURI, L.; JADÁN-GUERRERO, J.; GUEVARA, C.; SANCHEZ-GORDON, S.; CALLE-JIMENEZ, T.; LARA-ALVAREZ, P.; MEDINA, A.; NUNES, I. L. Accessibility Assessment in Mobile Applications for Android. In: NUNES, I. L. (Ed.). **Advances in Human Factors and Systems Interaction**. Cham: Springer International Publishing, 2020. (Advances in Intelligent Systems and Computing), p. 279–288. ISBN 978-3-030-20040-4. Citations on pages 89 and 145.

AGGARWAL, P. K.; SHARMA, S.; Riya; JAIN, P.; Anupam. Gaps Identification for User Experience for Model Driven Engineering. In: **2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence)**. Noida, India: IEEE, 2021. p. 196–199. Available: <http://dx.doi.org/10.1109/Confluence51648.2021.9377178>. Citation on page 30.

AGH, H.; RAMSIN, R. A pattern-based model-driven approach for situational method engineering. **Information and Software Technology**, v. 78, p. 95 – 120, 2016. ISSN 0950-5849. Available: <http://www.sciencedirect.com/science/article/pii/S0950584916300969>. Citation on page 143.

AGIRRE, J. A.; SAGARDUI, G.; ETXEBERRIA, L. Evolving legacy model transformations to aggregate non functional requirements of the domain. In: **2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)**. [S.l.: s.n.], 2015. p. 437–448. Citation on page 143.

AGUSTIN, J. L. H.; BARCO, P. C. d. A model-driven approach to develop high performance web applications. **Journal of Systems and Software**, v. 86, n. 12, p. 3013 – 3023, 2013. ISSN 0164-1212. Available: <http://www.sciencedirect.com/science/article/pii/S0164121213001751>. Citation on page 143.

AHMED, S.; ASHRAF, G. Model-based user interface engineering with design patterns. **Journal of Systems and Software**, v. 80, n. 8, p. 1408 – 1422, 2007. ISSN 0164-1212. Available: <http://www.sciencedirect.com/science/article/pii/S016412120600286X>. Citations on pages 59 and 144.

ALI, A.; RASHID, M.; AZAM, F.; RASHEED, Y.; ANWAR, M. W. A Model-Driven Framework for Android Supporting Cross-Platform GUI Development. In: **2021 National Computing Colleges Conference (NCCC)**. [S.l.: s.n.], 2021. p. 1–6. Citation on page 146.

ALMEIDA, F.; BERNARDO, N.; LACERDA, R. chapter, **Practical Approach for Apps Design in Compliance With Accessibility, Usability, and User Experience**. 2022. ISBN: 9781799878483 Pages: 109-134 Publisher: IGI Global. Available: <https://www.igi-global.com/chapter/practical-approach-for-apps-design-in-compliance-with-accessibility-usability-and-user-experience/www.igi-global.com/chapter/practical-approach-for-apps-design-in-compliance-with-accessibility-usability-and-u287256>. Citation on page 146.

ALSHAYBAN, A.; AHMED, I.; MALEK, S. Accessibility issues in android apps: State of affairs, sentiments, and ways forward. In: **Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering**. New York, NY, USA: Association for Computing Machinery, 2020. (ICSE '20), p. 1323–1334. ISBN 9781450371216. Available: <https://doi.org/10.1145/3377811.3380392>. Citation on page 89.

ALULEMA, D.; IRIBARNE, L.; CRIADO, J. A DSL for the Development of Heterogeneous Applications. In: **2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)**. [S.l.: s.n.], 2017. p. 251–257. Citation on page 145.

AMELLER, D.; FRANCH, X.; GÓMEZ, C.; ARAUJO, J.; SVENSSON, R. B.; BIFFL, S.; CABOT, J.; CORTELLESSA, V.; DANEVA, M.; FERNÁNDEZ, D. M.; MOREIRA, A.; MUCCINI, H.; VALLECILLO, A.; WIMMER, M.; AMARAL, V.; BRUNELIèRE, H.; BURGUEÑO, L.; GOULãO, M.; SCHÄTZ, B.; TEUFL, S. Handling non-functional requirements in Model-Driven Development: An ongoing industrial survey. In: **2015 IEEE 23rd International Requirements Engineering Conference (RE)**. [S.l.: s.n.], 2015. p. 208–213. Citation on page 143.

AMELLER, D.; FRANCH, X.; GÓMEZ, C.; MARTÍNEZ-FERNÁNDEZ, S.; ARAÚJO, J.; BIFFL, S.; CABOT, J.; CORTELLESSA, V.; FERNÁNDEZ, D. M.; MOREIRA, A.; MUCCINI, H.; VALLECILLO, A.; WIMMER, M.; AMARAL, V.; BÖHM, W.; BRUNELIERE, H.; BURGUEÑO, L.; GOULÃO, M.; TEUFL, S.; BERARDINELLI, L. Dealing with Non-Functional Requirements in Model-Driven Development: A Survey. **IEEE Transactions on Software Engineering**, v. 47, n. 4, p. 818–835, Apr. 2021. ISSN 1939-3520. Conference Name: IEEE Transactions on Software Engineering. Citation on page 40.

AMMAR, L. B.; MAHFOUDHI, A. Early usability evaluation in model driven framework. p. 23–30, 2013. Citation on page 144.

AMMAR, L. B.; TRABELSI, A.; MAHFOUDHI, A. Incorporating usability requirements into model transformation technologies. **Requirements Engineering**, v. 20, n. 4, p. 465–479, Nov. 2015. ISSN 1432-010X. Available: <https://doi.org/10.1007/s00766-014-0213-z>. Citations on pages 59, 60, 68, and 144.

AMMAR, L. B.; TRABELSI, A.; MAHFOUDHI, A. A model-driven approach for usability engineering of interactive systems. **Software Quality Journal**, v. 24, n. 2, p. 301–335, Jun. 2016. ISSN 1573-1367. Available: <https://doi.org/10.1007/s11219-014-9266-y>. Citations on pages 60, 68, and 145.

ANDRADE, W. T.; BRANCO, R. G. d.; CAGNIN, M. I.; PAIVA, D. M. B. Incorporating Accessibility Elements to the Software Engineering Process. **Advances in Human-Computer Interaction**, v. 2018, p. e1389208, Dec. 2018. ISSN 1687-5893. Publisher: Hindawi. Available: <https://www.hindawi.com/journals/ahci/2018/1389208/>. Citations on pages 58, 68, and 145.

ANJOS, M.; PELLEGRINI, F.; FLORENTIN, F.; ALVES, V.; ZANDONA, D.; RIBEIRO, B.; CORREIA, W.; QUINTINO, J.; MACÊDO, J. How to overcome the challenges of developing responsive and accessible websites–a case study. In: SPRINGER. **International Conference on Applied Human Factors and Ergonomics**. [S.l.], 2019. p. 260–270. Citation on page 145.

ANJOS, M.; PELLEGRINI, F.; FLORENTIN, F.; ALVES, V.; ZANDONA, D.; RIBEIRO, B.; CORREIA, W.; QUINTINO, J.; MACêDO, J. How to Overcome the Challenges of Developing Responsive and Accessible Websites – A Case Study. In: AHRAM, T.; FALCÃO, C. (Ed.). **Advances in Usability and User Experience**. Cham: Springer International Publishing, 2020. p. 260–270. ISBN 978-3-030-19135-1. Citations on pages 30, 39, and 145.

ANTONELLI, H. L.; SILVA, E. A. N. da; FORTES, R. P. M. A model-driven development for creating accessible web menus. **Procedia Computer Science**, v. 67, p. 95–104, 2015. ISSN 1877-0509. Proceedings of the 6th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion. Available: <https://www.sciencedirect.com/science/article/pii/S1877050915030999>. Citations on pages 77, 78, and 143.

Apple Inc. **Accessibility**. 2021. Access 04/08/2021. Available: <https://www.apple.com/accessibility/>. Citation on page 39.

AZENKOT, S.; HANLEY, M. J.; BAKER, C. M. How Accessibility Practitioners Promote the Creation of Accessible Products in Large Companies. **Proceedings of the ACM on Human-Computer Interaction**, v. 5, n. CSCW1, p. 1–27, Apr. 2021. ISSN 2573-0142. Available: <https://dl.acm.org/doi/10.1145/3449222>. Citation on page 146.

BAČÍKOVÁ, M.; PORUBÄN, J. DSL-driven generation of Graphical User Interfaces. **Central European Journal of Computer Science**, v. 4, n. 4, p. 204–221, Dec. 2014. ISSN 2081-9935. Available: <https://doi.org/10.2478/s13537-014-0210-9>. Citation on page 143.

BALLANTYNE, M.; JHA, A.; JACOBSEN, A.; HAWKER, J. S.; EL-GLALY, Y. N. Study of Accessibility Guidelines of Mobile Applications. In: **Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia**. Cairo Egypt: ACM, 2018. p. 305–315. ISBN 978-1-4503-6594-9. Available: <https://dl.acm.org/doi/10.1145/3282894.3282921>. Citations on pages 89 and 145.

BBC. **Accessibility for Products - Mobile Accessibility Guidelines**. 2020. Access 02/08/2021. Available: <http://www.bbc.co.uk/guidelines/futuremedia/accessibility/mobile>. Citations on pages 39, 88, and 145.

BENOUDA, H.; AZIZI, M.; ESBAI, R.; MOUSSAOUI, M. MDA Approach to Automate Code Generation for Mobile Applications. In: KIM, K. J.; WATTANAPONGSAKORN, N.; JOUKOV, N. (Ed.). **Mobile and Wireless Technologies 2016**. Singapore: Springer, 2016. (Lecture Notes in Electrical Engineering), p. 241–250. ISBN 978-981-10-1409-3. Citation on page 145.

BENOUDA, H.; AZIZI, M.; MOUSSAOUI, M.; ESBAI, R. Automatic code generation within MDA approach for cross-platform mobiles apps. In: **2017 First International Conference on Embedded Distributed Systems (EDiS)**. [S.l.: s.n.], 2017. p. 1–5. Citation on page 145.

BERNERS-LEE, T. World wide web consortium (w3c) launches international web accessibility initiative. **Web Accessibility Initiative (WAI)**, 1997. Available: <https://www.w3.org/Press/WAI-Launch.html>. Citation on page 28.

BÉZIVIN, J. Model Driven Engineering: An Emerging Technical Space. In: LÄMMEL, R.; SARAIVA, J.; VISSER, J. (Ed.). **Generative and Transformational Techniques in Software Engineering: International Summer School, GTTSE 2005, Braga, Portugal, July 4-8, 2005. Revised Papers**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 36–64. ISBN 978-3-540-46235-4. Available: <https://doi.org/10.1007/11877028_2>. Citations on pages 40 and 144.

BI, T.; XIA, X.; LO, D.; GRUNDY, J. C.; ZIMMERMANN, T.; FORD, D. Accessibility in Software Practice: A Practitioner's Perspective. **CoRR**, abs/2103.08778, 2021. Available: <https://arxiv.org/abs/2103.08778>. Citations on pages 29, 30, and 146.

BIN, Y.; ZHI, J.; XIAOHONG, C. An Approach for Selecting Implementation Strategies of Non-functional Requirements. In: **Proceedings of the Fourth Asia-Pacific Symposium on Internetware**. New York, NY, USA: ACM, 2012. (Internetware '12), p. 20:1–20:7. ISBN 978-1-4503-1888-4. Available: <http://doi.acm.org/10.1145/2430475.2430495>. Citation on page 143.

BLANCO, J. Z.; LUCRÉDIO, D. A holistic approach for cross-platform software development. **Journal of Systems and Software**, v. 179, p. 110985, Sep. 2021. ISSN 0164-1212. Available: <https://www.sciencedirect.com/science/article/pii/S0164121221000820>. Citation on page 146.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML Guia do Usuario**. 2rd. ed. [S.l.]: Editora Campus, 2005. Citations on pages 47 and 48.

BOURAOUI, A.; GHARBI, I. Model driven engineering of accessible and multi-platform graphical user interfaces by parameterized model transformations. **Science of Computer Programming**, v. 172, p. 63–101, March 2019. ISSN 0167-6423. Available: <https://www.sciencedirect.com/science/article/pii/S0167642318304301>. Citations on pages 19, 62, 63, 68, 69, 92, 108, 109, and 145.

BOURIMI, M.; TESORIERO, R. Non-Functional Requirements for Distributable User Interfaces in Agile Processes. In: **Proceedings of the 2014 Workshop on Distributed User Interfaces and Multimodal Interaction**. New York, NY, USA: ACM, 2014. (DUI '14), p. 54–66. ISBN 978-1-60558-724-0. Available: <http://doi.acm.org/10.1145/2677356.2677668>. Citation on page 143.

BRAJNIK, G.; HARPER, S. Detaching Control from Data Models in Model-Based Generation of User Interfaces. In: CIMIANO, P.; FRASINCAR, F.; HOUBEN, G.-J.; SCHWABE, D. (Ed.). **Engineering the Web in the Big Data Era**. Cham: Springer International Publishing, 2015. (Lecture Notes in Computer Science), p. 697–700. ISBN 978-3-319-19890-3. Citation on page 144.

BRAMBILLA, M. **Interaction Flow Modeling Language in the IIoT context**. 2015. 43 p. Available: <https://www.omg.org/news/meetings/tc/ma-15/special-events/iiot-pdf/Brambilla.pdf>. Citations on pages 43 and 144.

BRAMBILLA, M.; FRATERNALI, P. **Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML**. [S.l.]: Morgan Kaufmann, 2014. Citations on pages 29, 43, 45, 46, and 144.

____. Large-scale Model-Driven Engineering of web user interaction: The WebML and WebRatio experience. **Science of Computer Programming**, v. 89, p. 71 – 87, 2014. ISSN 0167-6423. Available: <http://www.sciencedirect.com/science/article/pii/S0167642313000701>. Citation on page 143.

BRAMBILLA, M.; MAURI, A.; UMUHOZA, E. Extending the Interaction Flow Modeling Language (IFML) for Model Driven Development of Mobile Applications Front End. In: AWAN, I.; YOUNAS, M.; FRANCH, X.; QUER, C. (Ed.). **Mobile Web Information Systems**. Cham: Springer International Publishing, 2014. (Lecture Notes in Computer Science), p. 176–191. ISBN 978-3-319-10359-4. Citations on pages 64, 68, and 84.

____. Extending the interaction flow modeling language (ifml) for model driven development of mobile applications front end. In: AWAN, I.; YOUNAS, M.; FRANCH, X.; QUER, C. (Ed.). **Mobile Web Information Systems**. Springer International Publishing, 2014, (Lecture Notes in Computer Science, v. 8640). p. 176–191. ISBN 978-3-319-10358-7. Available: <http://dx.doi.org/10.1007/978-3-319-10359-4_15>. Citation on page 143.

BRAMBILLA, M.; UMUHOZA, E.; ACERBIS, R. Model-driven development of user interfaces for IoT systems via domain-specific components and patterns. **Journal of Internet Services and Applications**, v. 8, n. 1, p. 14, Sep. 2017. ISSN 1869-0238. Available: <https://doi.org/10.1186/s13174-017-0064-1>. Citation on page 145.

BRANCO, R. G. D.; CAGNIN, M. I.; PAIVA, D. M. B. AccTrace: Accessibility in Phases of Requirements Engineering, Design, and Coding Software. In: **2014 14th International Conference on Computational Science and Its Applications**. [S.l.: s.n.], 2014. p. 225–228. Citations on pages 29, 58, and 143.

BREINER, K.; SEISSLER, M.; MEIXNER, G.; FORBRIG, P.; SEFFAH, A.; KLÖCKNER, K. PEICS: Towards HCI Patterns into Engineering of Interactive Systems. In: **Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems**. New York, NY, USA: ACM, 2010. (PEICS '10), p. 1–3. ISBN 978-1-4503-0246-3. Available: <http://doi.acm.org/10.1145/1824749.1824750>. Citation on page 144.

BRUNELIERE, H.; BURGER, E.; CABOT, J.; WIMMER, M. A feature-based survey of model view approaches. **Software & Systems Modeling**, v. 18, n. 3, p. 1931–1952, Jun. 2019. ISSN 1619-1366, 1619-1374. Available: <http://link.springer.com/10.1007/s10270-017-0622-9>. Citation on page 145.

BUARQUE, A.; CASTRO, J.; ALENCAR, F. The Role of NFRs when Transforming I* Requirements Models into OO-method Models. In: **Proceedings of the 28th Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2013. (SAC '13), p. 1305–1306. ISBN 978-1-4503-1656-9. Available: <http://doi.acm.org/10.1145/2480362.2480605>. Citation on page 143.

BUCCHIARONE, A.; CABOT, J.; PAIGE, R. F.; PIERANTONIO, A. Grand challenges in model-driven engineering: an analysis of the state of the research. **Software and Systems Modeling**, v. 19, n. 1, p. 5–13, Jan. 2020. ISSN 1619-1374. Available: <https://doi.org/10.1007/s10270-019-00773-6>. Citations on pages 41 and 145.

BUSTOS, B.; MARTÍN, A.; CECHICH, A. Diseño de interfaces guiado por restricciones de accesibilidad web. In: **XIII Congreso Americano en Software Engineering**. [S.l.: s.n.], 2010. p. 229–242. Citations on pages 59 and 144.

CALVARY, G.; COUTAZ, J.; THEVENIN, D.; LIMBOURG, Q.; BOUILLON, L.; VANDERDONCKT, J. A Unifying Reference Framework for multi-target user interfaces. **Interacting with Computers**, v. 15, n. 3, p. 289 – 308, 2003. ISSN 0953-5438. Available: <http://www.sciencedirect.com/science/article/pii/S0953543803000109>. Citations on pages 59, 60, 61, and 144.

CALVO, R.; SEYEDARABI, F.; SAVVA, A. Beyond web content accessibility guidelines: Expert accessibility reviews. In: **Proceedings of the 7th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion**. New York, NY, USA: ACM, 2016. (DSAI 2016), p. 77–84. ISBN 978-1-4503-4748-8. Available: <http://doi.acm.org.ez67.periodicos.capes.gov.br/10.1145/3019943.3019955>. Citations on pages 27 and 145.

CAO, R. K.; LIU, X. IFML-Based Web Application Modeling. **Procedia Computer Science**, v. 166, p. 129–133, Jan. 2020. ISSN 1877-0509. Available: <https://www.sciencedirect.com/science/article/pii/S1877050920301563>. Citation on page 145.

CARVALHO, M. C. N.; DIAS, F. S.; REIS, A. G. S.; FREIRE, A. P. Accessibility and Usability Problems Encountered on Websites and Applications in Mobile Devices by Blind and Normal-vision Users. In: **Proceedings of the 33rd Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2018. (SAC '18), p. 2022–2029. ISBN 978-1-4503-5191-1. Available: <http://doi.acm.org/10.1145/3167132.3167349>. Citation on page 89.

____. Accessibility and Usability Problems Encountered on Websites and Applications in Mobile Devices by Blind and Normal-vision Users. In: **Proceedings of the 33rd Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2018. (SAC '18), p. 2022–2029. ISBN 978-1-4503-5191-1. Event-place: Pau, France. Available: <http://doi.acm.org/10.1145/3167132.3167349>. Citation on page 145.

CERI, S.; FRATERNALI, P.; BONGIO, A. Web modeling language (webml): a modeling language for designing web sites. **Computer Networks**, v. 33, n. 1D6, p. 137 – 157, 2000. ISSN 1389-1286. Available: <http://www.sciencedirect.com/science/article/pii/S1389128600000402>. Citations on pages 42 and 144.

CHANNONTHAWAT, T.; LIMPIYAKORN, Y. Model driven development of android application prototypes from windows navigation diagrams. In: **2016 International Conference on Software Networking (ICSN)**. [S.l.: s.n.], 2016. p. 1–4. Citation on page 143.

CHEON, Y.; BARUA, A. Model driven development for android apps. In: THE STEERING COMMITTEE OF THE WORLD CONGRESS IN COMPUTER SCIENCE, COMPUTER .... **Proceedings of the International Conference on Software Engineering Research and Practice (SERP)**. [S.l.], 2018. p. 17–22. Citation on page 145.

CHIOU, P. T.; ALOTAIBI, A. S.; HALFOND, W. G. J. Detecting and localizing keyboard accessibility failures in web applications. In: **Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering**. Athens Greece: ACM, 2021. p. 855–867. ISBN 978-1-4503-8562-6. Available: <https://dl.acm.org/doi/10.1145/3468264.3468581>. Citation on page 146.

CHMIELEWSKI, J.; FLOTYńSKI, J.; RUMIńSKI, D.; WÓJTOWICZ, A. Declarative GUI descriptions for device-independent applications. **Personal and Ubiquitous Computing**, v. 20, n. 2, p. 185–194, Apr. 2016. ISSN 1617-4917. Available: <https://doi.org/10.1007/s00779-016-0903-2>. Citations on pages 62 and 145.

CHUNG, L.; LEITE, J. C. S. do P. On non-functional requirements in software engineering. In: ____. **Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 363–379. ISBN 978-3-642-02463-4. Available: <http://dx.doi.org/10.1007/978-3-642-02463-4_19>. Citation on page 144.

CHUNG, L.; NIXON, B.; YU, E.; MYLOPOULOS, J. Non-functional requirements in software engineering–kluwer academic publishers. **Massachusetts, USA**, 2000. Citation on page 144.

CIVIL, C. Lei nº 13.146, de 6 de julho 2015. **Institui a Lei Brasileira de Inclusão da Pessoa com Deficiência (Estatuto da Pessoa com Deficiência). Brasília**, 2015. Available: <http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2015/Lei/L13146.htm>. Citation on page 35.

CLERCKX, T.; LUYTEN, K.; CONINX, K. DynaMo-AID: A Design Process and a Runtime Architecture for Dynamic Model-Based User Interface Development. In: BASTIDE, R.; PALANQUE, P.; ROTH, J. (Ed.). **Engineering Human Computer Interaction and Interactive Systems: Joint Working Conferences EHCI-DSVIS 2004, Hamburg, Germany, July 11-13, 2004, Revised Selected Papers**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 77–95. ISBN 978-3-540-31961-0. DOI: 10.1007/11431879_5. Available: <http://dx.doi.org/10.1007/11431879_5>. Citation on page 144.

COMPUTING, A. *et al.* An architectural blueprint for autonomic computing. **IBM White Paper**, IBM Corporation Hawthorne, NY, v. 31, n. 2006, p. 1–6, 2006. Citation on page 66.

COOK, T. M. The americans with disabilities act: The move to integration. **Temp. LR**, HeinOnline, v. 64, p. 393, 1991. Citation on page 35.

COSTA, S. L. d.; NETO, V. V. G.; OLIVEIRA, J. L. d. A User Interface Stereotype to Build Web Portals. In: **2014 9th Latin American Web Congress**. [S.l.: s.n.], 2014. p. 10–18. Citation on page 143.

DAMACENO, R. J. P.; BRAGA, J. C.; MENA-CHALCO, J. P. Mobile device accessibility for the visually impaired: problems mapping and recommendations. **Universal Access in the Information Society**, v. 17, n. 2, p. 421–435, Jun 2018. ISSN 1615-5297. Available: <https://doi.org/10.1007/s10209-017-0540-1>. Citation on page 89.

DAVIES, J.; GIBBONS, J.; WELCH, J.; CRICHTON, E. Model-driven engineering of information systems: 10 years and 1000 versions. **Science of Computer Programming**, Elsevier, v. 89, p. 88–104, 2014. Available: <https://reader.elsevier.com/reader/sd/pii/S0167642313000270?token=9841AF6849F3FA6A075ECEC3BD090090900B66344F7BC0FEE3CE29AA2997F1B1DFEA3A8B7A219D originRegion=us-east-1&originCreation=20210504175546>. Citation on page 144.

DELGADO, A.; ESTEPA, A.; TROYANO, J. A.; ESTEPA, R. Reusing {UI} elements with Model-Based User Interface Development. **International Journal of Human-Computer Studies**, v. 86, p. 48 – 62, 2016. ISSN 1071-5819. Available: <http://www.sciencedirect.com/science/article/pii/S1071581915001470>. Citations on pages 58 and 143.

DESRUELLE, H.; ISENBERG, S.; BOTSIKAS, A.; VERGORI, P.; GIELEN, F. Accessible user interface support for multi-device ubiquitous applications: architectural modifiability considerations. **Universal Access in the Information Society**, v. 15, n. 1, p. 5–19, Mar. 2016. ISSN 1615-5297. Available: <https://doi.org/10.1007/s10209-014-0373-0>. Citation on page 143.

DE', R.; PANDEY, N.; PAL, A. Impact of digital surge during Covid-19 pandemic: A viewpoint on research and practice. **International Journal of Information Management**, v. 55, p. 102171, Dec. 2020. ISSN 0268-4012. Available: <https://www.sciencedirect.com/science/article/pii/S0268401220309622>. Citations on pages 27 and 145.

DIAS, A. L. **Um processo para sistemas web com foco em acessibilidade e usabilidade.** Phd Thesis (PhD Thesis) — São Carlos : Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2014. Citations on pages 29, 57, 67, 68, and 144.

DIAS, A. L.; FORTES, R. P. M.; MASIERO, P. C. Increasing the quality of web systems: By inserting requirements of accessibility and usability. In: **2012 Eighth International Conference on the Quality of Information and Communications Technology**. Lisbon, Portugal: IEEE, 2012. p. 224–229. Available: <https://ieeexplore.ieee.org/document/6511814>. Citations on pages 57, 67, 68, and 143.

DIAS, F.; DUARTE, L.; FORTES, R. P. M. AccessMDD: an MDD approach for generating accessible mobile applications. In: **In The 39th ACM International Conference on Design of Communication (SIGDOC '21), October 12–14, 2021, Virtual Event, USA. ACM, New York, NY, USA,**. [S.l.: s.n.], 2021. Citations on pages 107 and 146.

DIAS, F. S. **AccessMDD: uma abordagem MDD para geração de aplicativos móveis acessíveis**. 156 f. Master's Thesis (Master Dissertation in Computer Science and Computational Mathematics) — ICMC/USP, São Carlos - SP, 2021. Available: <https://www.teses.usp.br/teses/disponiveis/55/55134/tde-17082021-135910/en.php>. Accessed: 10 sept. 2021. Citations on pages 87, 88, 89, 93, and 107.

DINGSØYR, T.; MOE, N. B.; FÆGRI, T. E.; SEIM, E. A. Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. **Empirical Software Engineering**, Springer, v. 23, n. 1, p. 490–520, 2018. Citation on page 145.

DOUKAS, C.; ANTONELLI, F. Compose: Building smart amp; context-aware mobile applications utilizing iot technologies. In: **Global Information Infrastructure Symposium - GIIS 2013**. [S.l.: s.n.], 2013. p. 1–6. Citation on page 92.

DUARTE, C.; MATOS, I.; VICENTE, J. a.; SALVADO, A.; DUARTE, C. M.; CARRIçO, L. Development technologies impact in web accessibility. In: **Proceedings of the 13th Web for All Conference**. New York, NY, USA: Association for Computing Machinery, 2016. (W4A '16). ISBN 9781450341387. Available: <https://doi.org/10.1145/2899475.2899498>. Citations on pages 28, 39, and 145.

DURAN-LIMON, H. A.; BLAIR, G. S.; FRIDAY, A.; GRACE, P.; SAMARTZIDIS, G.; SIVAHARAN, T.; WU, M. **Context-Aware Middleware for Pervasive and Ad Hoc Environments**. Lancaster, UK, 2003. 9 p. Citations on pages 62, 92, and 144.

ECKHARDT, J.; VOGELSANG, A.; FERNÁNDEZ, D. M. Are "Non-functional" Requirements Really Non-functional?: An Investigation of Non-functional Requirements in Practice. In: **Proceedings of the 38th International Conference on Software Engineering**. New

York, NY, USA: ACM, 2016. (ICSE '16), p. 832–842. ISBN 978-1-4503-3900-1. Available: <http://doi.acm.org/10.1145/2884781.2884788>. Citation on page 143.

EMBLEY, D. W.; THALHEIM, B. (Ed.). **Handbook of Conceptual Modeling**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. ISBN 978-3-642-15864-3 978-3-642-15865-0. Available: <http://link.springer.com/10.1007/978-3-642-15865-0>. Citation on page 144.

ENGEL, J. A Model- and Pattern-based Approach for Development of User Interfaces of Interactive Systems. In: **Proceedings of the 2Nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems**. New York, NY, USA: ACM, 2010. (EICS '10), p. 337–340. ISBN 978-1-4503-0083-4. Available: <http://doi.acm.org/10.1145/1822018.1822075>. Citation on page 144.

European Commission. **Persons with disabilities.** European Commission, 2021. Available: <https://ec.europa.eu/social/main.jsp?catId=1137&langId=en>. Citation on page 28.

FATIMA, I.; ANWAR, M. W.; AZAM, F.; MAQBOOL, B.; TUFAIL, H. Extending Interaction Flow Modeling Language (IFML) for Android User Interface Components. In: DAMAŠEVIČIUS, R.; VASILJEVIENė, G. (Ed.). **Information and Software Technologies**. Cham: Springer International Publishing, 2019. (Communications in Computer and Information Science), p. 76–89. ISBN 978-3-030-30275-7. Citations on pages 64, 68, and 145.

FAVRE, J.-M. Towards a basic theory to model model driven engineering. In: CITESEER. **3rd Workshop in Software Model Engineering, WiSME**. [S.l.], 2004. p. 262–271. Citations on pages 40 and 144.

____. Foundations of model (driven)(reverse) engineering: Models–episode i: stories of the fidus papyrus and of the solarus. In: SCHLOSS DAGSTUHL-LEIBNIZ-ZENTRUM FÜR INFORMATIK. **Dagstuhl Seminar Proceedings**. [S.l.], 2005. Citation on page 144.

____. Megamodelling and etymology. In: SCHLOSS DAGSTUHL-LEIBNIZ-ZENTRUM FÜR INFORMATIK. **Dagstuhl Seminar Proceedings**. [S.l.], 2006. Citation on page 144.

FERREIRA, D. de A.; SILVA, A. R. da. RSLingo: An information extraction approach toward formal requirements specifications. In: **2012 Second IEEE International Workshop on Model-Driven Requirements Engineering (MoDRE)**. [S.l.: s.n.], 2012. p. 39–48. Citation on page 144.

FOGLI, D.; PROVENZA, L. P.; BERNAREGGI, C. A universal design resource for rich Internet applications based on design patterns. **Universal Access in the Information Society**, v. 13, n. 2, p. 205–226, 2014. ISSN 1615-5297. Available: <http://dx.doi.org/10.1007/s10209-013-0291-6>. Citation on page 144.

FORBRIG, P.; SAURIN, M. Supporting the HCI Aspect of Agile Software Development by Tool Support for UI-Pattern Transformations. In: BOGDAN, C.; GULLIKSEN, J.; SAUER, S.; FORBRIG, P.; WINCKLER, M.; JOHNSON, C.; PALANQUE, P.; BERNHAUPT, R.; KIS, F. (Ed.). **Human-Centered and Error-Resilient Systems Development: IFIP WG 13.2/13.5 Joint Working Conference, 6th International Conference on Human-Centered Software Engineering, HCSE 2016, and 8th International Conference on Human Error, Safety, and System Development, HESSD 2016, Stockholm, Sweden, August 29-31, 2016, Proceedings**. Cham: Springer International Publishing, 2016. p. 17–29. ISBN 978-3-319-44902-9. DOI: 10.1007/978-3-319-44902-9_2. Available: <http://dx.doi.org/10.1007/978-3-319-44902-9_2>. Citation on page 143.

FORWARD, A.; LETHBRIDGE, T.; BADREDDIN, O. Problems and opportunities for model-centric vs code-centric development: A survey of software professionals. In: **5th Workshop" From code centric to model centric: Evaluating the effectiveness of MDD (C2M: EEMDD)", University Pierre & Marie Curie, Paris**. [S.l.: s.n.], 2010. Citation on page 144.

FRANCESE, R.; RISI, M.; SCANNIELLO, G.; TORTORA, G. Model-driven development for multi-platform mobile applications. In: ABRAHAMSSON, P.; CORRAL, L.; OIVO, M.; RUSSO, B. (Ed.). **Product-Focused Software Process Improvement**. Cham: Springer International Publishing, 2015. p. 61–67. ISBN 978-3-319-26844-6. Citation on page 143.

FREIRE, A. P.; RUSSO, C. M.; FORTES, R. P. M. A Survey on the Accessibility Awareness of People Involved in Web Development Projects in Brazil. In: **Proceedings of the 2008 International Cross-disciplinary Conference on Web Accessibility (W4A)**. New York, NY, USA: ACM, 2008. (W4A '08), p. 87–96. ISBN 978-1-60558-153-8. Available: <http://doi.acm.org/10.1145/1368044.1368064>. Citations on pages 39 and 144.

GAGGI, O.; QUADRIO, G.; BUJARI, A. Accessibility for the Visually Impaired: State of the Art and Open Issues. In: **2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)**. [S.l.: s.n.], 2019. p. 1–6. ISSN: 2331-9860. Citations on pages 29 and 145.

GAJOS, K. Z.; WELD, D. S.; WOBBROCK, J. O. Automatically generating personalized user interfaces with Supple. **Artificial Intelligence**, v. 174, n. 12, p. 910–950, Aug. 2010. ISSN 0004-3702. Available: <https://www.sciencedirect.com/science/article/pii/S0004370210000822>. Citations on pages 62, 92, and 144.

GAMITO, I.; SILVA, A. R. da. From Rigorous Requirements and User Interfaces Specifications into Software Business Applications. In: SHEPPERD, M.; ABREU, F. Brito e; SILVA, A. Rodrigues da; PÉREZ-CASTILLO, R. (Ed.). **Quality of Information and Communications Technology**. Cham: Springer International Publishing, 2020. (Communications in Computer and Information Science), p. 459–473. ISBN 978-3-030-58793-2. Citations on pages 30 and 145.

GAOUAR, L.; BENAMAR, A.; BENDIMERAD, F. T. Model driven approaches to cross platform mobile development. In: **Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication**. New York, NY, USA: Association for Computing Machinery, 2015. (IPAC '15). ISBN 9781450334587. Available: <https://doi.org/10.1145/2816839.2816882>. Citation on page 143.

GARĆIA-BORGOÑÓN, L.; BARCELONA, M. A.; GARĆIA-GARĆIA, J. A.; ESCALONA, M. J. Software Process Accessibility in Practice: A Case Study. **Procedia Computer Science**, v. 27, p. 292 – 301, 2014. ISSN 1877-0509. Available: <http://www.sciencedirect.com/science/article/pii/S1877050914000349>. Citation on page 143.

GHARAAT, M.; SHARBAF, M.; ZAMANI, B.; HAMOU-LHADJ, A. ALBA: a model-driven framework for the automatic generation of android location-based apps. **Automated Software Engineering**, v. 28, n. 1, p. 2, Jan. 2021. ISSN 1573-7535. Available: <https://doi.org/10.1007/s10515-020-00278-3>. Citation on page 146.

GIACHETTI, G.; MARÍN, B.; LÓPEZ, L.; FRANCH, X.; PASTOR, O. Verifying goal-oriented specifications used in model-driven development processes. **Information Systems**, v. 64, p. 41 – 62, 2017. ISSN 0306-4379. Available: <http://www.sciencedirect.com/science/article/pii/S0306437915300478>. Citation on page 143.

GIBBS, I.; DASCALU, S.; HARRIS, F. C.; Jr. A separation-based {UI} architecture with a {DSL} for role specialization. **Journal of Systems and Software**, v. 101, p. 69 – 85, 2015. ISSN 0164-1212. Available: <http://www.sciencedirect.com/science/article/pii/S0164121214002702>. Citation on page 143.

GIL, A. C. Como elaborar projetos de pesquisa. são paulo: Atlas, 2002. **Métodos e técnicas de pesquisa social**, v. 6, p. 22–23, 2009. Citation on page 32.

GONÇALVES, R.; MARTINS, J.; PEREIRA, J.; OLIVEIRA, M. A.-Y.; FERREIRA, J. J. P. Enterprise Web Accessibility Levels Amongst the Forbes 250: Where Art Thou O Virtuous Leader? **Journal of Business Ethics**, v. 113, n. 2, p. 363–375, Mar. 2013. ISSN 1573-0697. Available: <https://doi.org/10.1007/s10551-012-1309-3>. Citation on page 144.

GONZÁLEZ-HUERTA, J.; INSFRAN, E.; ABRAHãO, S.; MCGREGOR, J. D. Non-functional Requirements in Model-driven Software Product Line Engineering. In: **Proceedings of the Fourth International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages**. New York, NY, USA: ACM, 2012. (NFPinDSML '12), p. 6:1–6:6. ISBN 978-1-4503-1807-5. Available: <http://doi.acm.org/10.1145/2420942.2420948>. Citation on page 143.

Google LLC. **Build more accessible apps**. 2021. Access 08/04/2021. Available: <https://developer.android.com/guide/topics/ui/accessibility>. Citation on page 39.

____. **Build more accessible apps**. 2021. Access 27/05/2021. Available: <https://developer.android.com/guide/topics/ui/accessibility>. Citation on page 88.

GRECO, G. M. Accessibility Studies: Abuses, Misuses and the Method of Poietic Design. In: STEPHANIDIS, C. (Ed.). **HCI International 2019 – Late Breaking Papers**. Cham: Springer International Publishing, 2019. (Lecture Notes in Computer Science), p. 15–27. ISBN 978-3-030-30033-3. Citations on pages 29 and 145.

GREENFIELD, J.; SHORT, K. Software factories: Assembling applications with patterns, models, frameworks and tools. In: **Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications**. New York, NY, USA: ACM, 2003. (OOPSLA '03), p. 16–27. ISBN 1-58113-751-6. Available: <http://doi.acm.org/10.1145/949344.949348>. Citations on pages 41 and 144.

GRILLO, F. D. N.; FORTES, R. P. d. M. Accessible Modeling on the Web: A Case Study. **Procedia Computer Science**, v. 27, p. 460 – 470, 2014. ISSN 1877-0509. Available: <http://www.sciencedirect.com/science/article/pii/S1877050914000520>. Citation on page 143.

GROUP, O. M. **MDA Specifications**. Object Management Group, 2021. [Online; accessed 16. Feb. 2020]. Available: <https://www.omg.org/mda/specs.htm>. Citation on page 30.

GSA, U. G. S. A. **Glossary of Section 508 Terms | Section508.gov**. 2020. Available: <https://www.section508.gov/content/glossary>. Citation on page 36.

____. **IT Accessibility Laws and Policies | Section508.gov**. 2020. Available: <https://www.section508.gov/manage/laws-and-policies>. Citation on page 39.

GUERRERO-GARCIA, J.; GONZALEZ-CALLEROS, J. M.; VANDERDONCKT, J.; MUNOZ-ARTEAGA, J. A Theoretical Survey of User Interface Description Languages: Preliminary Results. In: **2009 Latin American Web Congress**. [S.l.: s.n.], 2009. p. 36–43. Citations on pages 62 and 144.

HAILPERN, B.; TARR, P. Model-driven development: The good, the bad, and the ugly. **IBM Systems Journal**, v. 45, n. 3, p. 451–461, 2006. ISSN 0018-8670. Conference Name: IBM Systems Journal. Citation on page 144.

HAMDANI, M.; BUTT, W. H.; ANWAR, M. W.; AZAM, F. A systematic literature review on interaction flow modeling language (ifml). In: **Proceedings of the 2018 2nd International Conference on Management Engineering, Software Engineering and Service Sciences**. Association for Computing Machinery, 2018. (ICMSS 2018), p. 134–138. ISBN 978-1-4503-5431-8. Available: <https://doi.org/10.1145/3180374.3181333>. Citation on page 145.

HAYAT, Z.; RASHID, M.; AZAM, F.; RASHEED, Y.; ANWAR, M. W. Extension of interaction flow modeling language for geographical information systems. In: **2021 10th International Conference on Software and Computer Applications**. [S.l.: s.n.], 2021. p. 186–192. Citation on page 146.

HAZZAN, O.; KRAMER, J. The Role of Abstraction in Software Engineering. p. 2, 2008. Citation on page 144.

HEITKÖTTER, H.; MAJCHRZAK, T. A.; KUCHEN, H. Cross-platform model-driven development of mobile applications with md2. In: **Proceedings of the 28th Annual ACM Symposium on Applied Computing**. New York, NY, USA: Association for Computing Machinery, 2013. (SAC '13), p. 526–533. ISBN 9781450316569. Available: <https://doi.org/10.1145/2480362.2480464>. Citations on pages 61, 62, 68, and 92.

____. Cross-platform model-driven development of mobile applications with md^2. In: **Proceedings of the 28th Annual ACM Symposium on Applied Computing**. New York, NY, USA: Association for Computing Machinery, 2013. (SAC '13), p. 526–533. ISBN 978-1-4503-1656-9. Available: <https://doi.org/10.1145/2480362.2480464>. Citation on page 144.

HELMS, J.; SCHAEFER, R.; LUYTEN, K.; VERMEULEN, J.; ABRAMS, M.; COYETTE, A.; VANDERDONCKT, J. Human-centered engineering of interactive systems with the user interface markup language. In: ____. **Human-Centered Software Engineering: Software Engineering Models, Patterns and Architectures for HCI**. London: Springer London, 2009. p. 139–171. ISBN 978-1-84800-907-3. Available: <https://doi.org/10.1007/978-1-84800-907-3_7>. Citations on pages 62, 92, and 144.

HENTATI, M.; AMMAR, L. B.; TRABELSI, A.; MAHFOUDHI, A. Model-driven engineering for optimizing the usability of user interfaces. In: **Proceedings of the 18th International Conference on Enterprise Information Systems**. Setubal, PRT: SCITEPRESS - Science and Technology Publications, Lda, 2016. (ICEIS 2016), p. 459–466. ISBN 9789897581878. Available: <https://doi.org/10.5220/0005838004590466>. Citations on pages 60, 68, and 145.

____. An approach for incorporating the usability optimization process into the model transformation. In: MADUREIRA, A. M.; ABRAHAM, A.; GAMBOA, D.; NOVAIS, P. (Ed.). **Intelligent Systems Design and Applications**. Cham: Springer International Publishing, 2017. p. 879–888. ISBN 978-3-319-53480-0. Citation on page 145.

HENTATI, M.; TRABELSI, A.; AMMAR, L. B.; MAHFOUDHI, A. Towards optimizing the usability of user interface generated with model-driven development process. In: **2015 8th International Conference on Human System Interaction (HSI)**. [S.l.: s.n.], 2015. p. 206–212. Citations on pages 60, 68, and 144.

HONG, J. Accessible Human-Error Interactions in AI Applications for the Blind. In: **Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers**. New York, NY, USA: Association for Computing Machinery, 2018. (UbiComp '18), p. 522–528. ISBN 978-1-4503-5966-5. Available: <https://doi.org/10.1145/3267305.3267321>. Citation on page 145.

HORKOFF, J.; YU, E. Comparison and evaluation of goal-oriented satisfaction analysis techniques. **Requirements Engineering**, v. 18, n. 3, p. 199–222, 2013. ISSN 1432-010X. Available: <http://dx.doi.org/10.1007/s00766-011-0143-y>. Citation on page 144.

HOUTENVILLE, A.; BOEGE, S. 2019 annual report on people with disabilities in america. **Institute on Disability, University of New Hampshire**, ERIC, 2020. Available: <https://disabilitycompendium.org/sites/default/files/user-uploads/2019%20Annual%20Report%20---%20FINAL%20ALL.pdf>. Citations on pages 28 and 145.

HUANG, A.; PAN, M.; ZHANG, T.; LI, X. Static extraction of IFML models for Android apps. In: **Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings**. New York, NY, USA: Association for Computing Machinery, 2018. (MODELS '18), p. 53–54. ISBN 978-1-4503-5965-8. Available: <https://doi.org/10.1145/3270112.3278185>. Citation on page 145.

IBGE. **Censo Demográfico 2010**. 2010. Available: <ftp://ftp.ibge.gov.br/Censos/Censo_Demografico_2010/Caracteristicas_Gerais_Religiao_Deficiencia/tab1_3.pdf>. Citation on page 28.

IBM. **Home – IBM Accessibility**. 2021. Access 31/08/2021. Available: <https://www.ibm.com/able/>. Citations on pages 39 and 146.

INC., I. G. L. C. **Accessibility | IMS Global Learning Consortium**. 2021. Access 08/04/2021. Available: <http://www.imsglobal.org/activity/accessibility>. Citations on pages 39 and 146.

ISO. **ISO - ISO 9241-171:2008 - Ergonomics of human-system interaction — Part 171: Guidance on software accessibility**. 2008. Access 01/05/2019. Available: <https://www.iso.org/standard/39080.html>. Citations on pages 39 and 57.

ISO. **ISO/IEC 24751-1:2008**. 2008. Available: <https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/04/15/41521.html>. Citation on page 39.

____. **ISO 14289-1:2014**. 2014. Available: <https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/45/64599.html>. Citation on page 39.

____. **ISO/IEC Guide 71:2014**. 2014. Available: <https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/05/73/57385.html>. Citation on page 39.

____. **ISO/IEC 29138-1:2018**. 2018. Available: <https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/19/71953.html>. Citation on page 39.

ISO 12207. **ISO - ISO/IEC 12207:2008 - Systems and software engineering — Software life cycle processes**. 2008. Available: <https://www.iso.org/standard/43447.html>. Citation on page 58.

ISO 9241-11. **Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts**. [S.l.], 2018. Available: <https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en>. Citations on pages 28, 35, 39, and 57.

JONES, C.; JIA, X. The axiom model framework: Transforming requirements to native code for cross-platform mobile applications. In: **2014 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)**. [S.l.: s.n.], 2014. p. 1–12. Citation on page 143.

JURISTO, N.; MORENO, A. M.; SANCHEZ-SEGURA, M.-I. Analysing the impact of usability on software design. **Journal of Systems and Software**, v. 80, n. 9, p. 1506 – 1516, 2007. ISSN 0164-1212. Available: <http://www.sciencedirect.com/science/article/pii/S0164121207000088>. Citation on page 144.

KAINDL, H. Model-Based Transition from Requirements to High-Level Software Design. In: **2013 20th Asia-Pacific Software Engineering Conference (APSEC)**. [S.l.: s.n.], 2013. v. 2, p. 81–82. Citation on page 143.

KAUFHOLD, M.-A.; REUTER, C.; COMES, T.; MIRBABAIE, M.; STIEGLITZ, S. 2nd workshop on mobile resilience: Designing mobile interactive systems for crisis response. In: **Adjunct Publication of the 23rd International Conference on Mobile Human-Computer Interaction**. New York, NY, USA: Association for Computing Machinery, 2021. (MobileHCI '21 Adjunct). ISBN 9781450383295. Available: <https://doi.org/10.1145/3447527.3474869>. Citations on pages 27 and 146.

KAWAS, S.; VONESSEN, L.; KO, A. J. Teaching accessibility: A design exploration of faculty professional development at scale. In: **Proceedings of the 50th ACM Technical Symposium on Computer Science Education**. New York, NY, USA: Association for Computing Machinery, 2019. (SIGCSE '19), p. 983–989. ISBN 9781450358903. Available: <https://doi.org/10.1145/3287324.3287399>. Citations on pages 36 and 145.

KEMP, S. **Digital 2021: Global Overview Report**. 2021. Available: <https://datareportal.com/reports/digital-2021-global-overview-report>. Citations on pages 27 and 146.

KHADDAM, I.; VANDERDONCKT, J. Model Voyager: Visualization of Cameleon Reference Framework UI models. In: . [s.n.], 2020. Available: <https://dial.uclouvain.be/pr/boreal/object/boreal:230049>. Citation on page 145.

KHAN, M.; AZAM, F.; RASHID, M.; SAMEA, F.; ANWAR, M.; MUZAFFAR, A.; HAIDER, W. A Retargetable Model-Driven Framework for the Development of Mobile User Interfaces. **Journal of Circuits, Systems and Computers**, v. 31, Jan. 2021. Citation on page 146.

KHATTER, K.; KALIA, A. Integration of non-functional requirements in model-driven architecture. In: **Fifth International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2013)**. [S.l.: s.n.], 2013. p. 359–364. Citation on page 143.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. 2007. Citation on page 52.

KRAINZ, E.; FEINER, J.; FRUHMANN, M. Accelerated development for accessible apps – model driven development of transportation apps for visually impaired people. In: BOGDAN, C.; GULLIKSEN, J.; SAUER, S.; FORBRIG, P.; WINCKLER, M.; JOHNSON, C.; PALANQUE, P.; BERNHAUPT, R.; KIS, F. (Ed.). **Human-Centered and Error-Resilient Systems Development**. Cham: Springer International Publishing, 2016. p. 374–381. ISBN 978-3-319-44902-9. Citations on pages 62, 92, and 145.

KRAINZ, E.; MIESENBERGER, K. Accapto, a Generic Design and Development Toolkit for Accessible Mobile Apps. **Studies in health technology and informatics**, v. 242, p. 660—664, 2017. ISSN 0926-9630. Available: <http://europepmc.org/abstract/MED/28873868>. Citations on pages 61, 68, 69, 108, and 109.

____. Accapto, a Generic Design and Development Toolkit for Accessible Mobile Apps. **Harnessing the Power of Technology to Improve Lives**, p. 660–664, 2017. Publisher: IOS Press. Available: <https://ebooks.iospress.nl/doi/10.3233/978-1-61499-798-6-660>. Citation on page 145.

KRAINZ, E.; MIESENBERGER, K.; FEINER, J. Can We Improve App Accessibility with Advanced Development Methods? p. 64–70, 06 2018. Citation on page 62.

____. Can we improve app accessibility with advanced development methods? In: SPRINGER. **International Conference on Computers Helping People with Special Needs**. [S.l.], 2018. p. 64–70. Citation on page 145.

KRAMER, J. Is abstraction the key to computing? **COMMUNICATIONS OF THE ACM**, v. 50, n. 4, p. 6, 2005. Citation on page 144.

____. Is abstraction the key to computing? **Communications of the ACM**, v. 50, n. 4, p. 36–42, Apr. 2007. ISSN 0001-0782. Available: <https://doi.org/10.1145/1232743.1232745>. Citation on page 144.

KÜHNE, T. What is a model? In: BEZIVIN, J.; HECKEL, R. (Ed.). **Language Engineering for Model-Driven Software Development**. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2005. (Dagstuhl Seminar Proceedings, 04101). ISSN 1862-4405. Available: <http://drops.dagstuhl.de/opus/volltexte/2005/23>. Citation on page 40.

LAAZ, N.; MBARKI, S. Combining Ontologies and IFML Models Regarding the GUIs of Rich Internet Applications. In: DICHEV, C.; AGRE, G. (Ed.). **Artificial Intelligence: Methodology, Systems, and Applications: 17th International Conference, AIMSA 2016, Varna, Bulgaria, September 7-10, 2016, Proceedings**. Cham: Springer International Publishing, 2016. p. 226–236. ISBN 978-3-319-44748-3. DOI: 10.1007/978-3-319-44748-3_22. Available: <http://dx.doi.org/10.1007/978-3-319-44748-3_22>. Citation on page 143.

____. Integrating IFML models and owl ontologies to derive UIs web-Apps. In: **2016 International Conference on Information Technology for Organizations Development (IT4OD)**. Fez, Morocco: IEEE, 2016. p. 1–6. Citations on pages 64, 65, 68, and 143.

____. OntoIFML: Automatic Generation of Annotated Web Pages from IFML and Ontologies using the MDA Approach: A Case Study of an EMR Management Application:. In: **Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development**. Prague, Czech Republic: SCITEPRESS - Science and Technology Publications, 2019. p. 353–361. ISBN 978-989-758-358-2. Available: <http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0007402203530361>. Citations on pages 65, 68, and 145.

LACHGAR, M.; ABDALI, A. MODELING AND GENERATING THE USER INTERFACE OF MOBILE DEVICES AND WEB DEVELOPMENT WITH DSL. **. Vol.**, p. 9, 2005. Citation on page 144.

____. Modeling and generating native code for cross-platform mobile applications using DSL. **Intelligent Automation & Soft Computing**, v. 23, n. 3, p. 445–458, Jul. 2017. ISSN 1079-8587. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10798587.2016.1239392. Available: <https://doi.org/10.1080/10798587.2016.1239392>. Citation on page 145.

LAZAR, J. Web accessibility policy and law. In: **Web Accessibility**. [S.l.]: Springer, 2019. p. 247–261. Citations on pages 28 and 39.

LEITE, M. V. R.; SCATALON, L. P.; FREIRE, A. P.; ELER, M. M. Accessibility in the mobile development industry in Brazil: Awareness, knowledge, adoption, motivations and barriers. **Journal of Systems and Software**, v. 177, Jul. 2021. ISSN 0164-1212. Available: <https://www.sciencedirect.com/science/article/pii/S016412122100039X>. Citations on pages 28, 29, 30, and 146.

Level Access. **Accessibility Laws & Standards**. 2021. [Online; accessed 07. Jul. 2021]. Available: <https://www.levelaccess.com/accessibility-regulations/>. Citations on pages 28 and 145.

LI, N.; HUA, Q.; WANG, S.; YU, K.; WANG, L. Research on a Pattern-Based User Interface Development Method. In: **2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics**. [S.l.: s.n.], 2015. v. 1, p. 443–447. Citation on page 143.

LIMA, J. F.; CARAN, G. M.; MOLINARO, L. F. R.; GARROSSINI, D. F. Analysis of Accessibility Initiatives Applied to the Web. **Procedia Technology**, v. 5, p. 319 – 326, 2012. ISSN 2212-0173. Available: <http://www.sciencedirect.com/science/article/pii/S2212017312004665>. Citation on page 143.

LIMBOURG, Q.; VANDERDONCKT, J.; MICHOTTE, B.; BOUILLON, L.; LÓPEZ-JAQUERO, V. Usixml: a language supporting multi-path development of user interfaces. In: SPRINGER. **International Workshop on Design, Specification, and Verification of Interactive Systems**. [S.l.], 2004. p. 200–220. Citations on pages 61 and 144.

____. Usixml: A language supporting multi-path development of user interfaces. In: BASTIDE, R.; PALANQUE, P.; ROTH, J. (Ed.). **Engineering Human Computer Interaction and Interactive Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 200–220. ISBN 978-3-540-31961-0. Citations on pages 62 and 92.

LINAJE, M.; LOZANO-TELLO, A.; PEREZ-TOLEDANO, M. A.; PRECIADO, J. C.; RODRIGUEZ-ECHEVERRIA, R.; SANCHEZ-FIGUEROA, F. Providing RIA user interfaces with accessibility properties. **Journal of Symbolic Computation**, v. 46, n. 2, p. 207 – 217, 2011. ISSN 0747-7171. Available: <http://www.sciencedirect.com/science/article/pii/S0747717110001380>. Citation on page 144.

LOWE, D.; PRESSMAN, S. **Engenharia Web**. [S.l.]: Rio de Janeiro: LTC Editoria SA, 2009. Citations on pages 57 and 144.

LTD., P. **Perform Systematic Literature Reviews**. 2014. Available: <https://parsif.al/about/>. Citation on page 52.

LUCRÉDIO, D. **Uma abordagem orientada a modelos para reutilização de software**. 277 f. Phd Thesis (Doutorado em Ciências de Computação e Matemática Computacional) — ICMC/USP, São Carlos - SP, 2009. Available: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-01072008-143726/>. Accessed: 10 mai. 2018. Citations on pages 40, 41, 105, and 144.

LUDEWIG, J. Models in software engineering–an introduction. **Software and Systems Modeling**, Springer, v. 2, n. 1, p. 5–14, 2003. Citations on pages 40 and 144.

LUMERTZ, P. R.; RIBEIRO, L.; DUARTE, L. M. User interfaces metamodel based on graphs. **Journal of Visual Languages & Computing**, v. 32, p. 1 – 34, 2016. ISSN 1045-926X. Available: <http://www.sciencedirect.com/science/article/pii/S1045926X1500083X>. Citation on page 143.

MACIK, M.; CERNY, T.; SLAVIK, P. Context-sensitive, cross-platform user interface generation. **Journal on Multimodal User Interfaces**, v. 8, n. 2, p. 217–229, Jun. 2014. ISSN 1783-8738. Available: <https://doi.org/10.1007/s12193-013-0141-0>. Citation on page 143.

MAIA, L. S. **Um processo para o desenvolvimento de aplicações Web acessíveis**. Master's Thesis (Master's Thesis) — Faculdade de Computação – UFMS, 2010. Citations on pages 58 and 144.

MARTÍN, A.; CECHICH, A.; GORDILLO, S.; ROSSI, G. A Three-Layered Approach to Model Web Accessibility for Blind Users. In: **2007 Latin American Web Conference (LA-WEB 2007)**. Santiago, Chile: IEEE, 2007. p. 76–83. Citations on pages 59 and 144.

MARTÍN, A.; CECHICH, A.; ROSSI, G. Accessibility at Early Stages: Insights from the Designer Perspective. In: **Proceedings of the International Cross-Disciplinary Conference on Web Accessibility**. New York, NY, USA: ACM, 2011. (W4A '11), p. 9:1–9:9. ISBN 978-1-4503-0476-4. Available: <https://dl.acm.org/doi/abs/10.1145/1969289.1969302>. Citations on pages 29, 59, 68, and 144.

MARTÍN, A.; MAZALU, R.; CECHICH, A. Supporting an Aspect-Oriented Approach to Web Accessibility Design. In: **2010 Fifth International Conference on Software Engineering Advances**. Nice, France: IEEE, 2010. p. 20–25. Citations on pages 59 and 144.

MARTÍN, A.; ROSSI, G.; CECHICH, A.; GORDILLO, S. Engineering Accessible Web Applications. An Aspect-Oriented Approach. **World Wide Web**, v. 13, n. 4, p. 419–440, 2010. ISSN 1573-1413. Available: <http://dx.doi.org/10.1007/s11280-010-0091-3>. Citations on pages 59 and 144.

MARTINS, B. F.; SOUZA, V. E. S. A Model-Driven Approach for the Design of Web Information Systems based on Frameworks. In: **Proceedings of the 21st Brazilian Symposium on Multimedia and the Web**. New York, NY, USA: Association for Computing Machinery, 2015. (WebMedia '15), p. 41–48. ISBN 978-1-4503-3959-9. Available: <https://doi.org/10.1145/2820426.2820439>. Citation on page 143.

MATEUS, D. A.; SILVA, C. A.; ELER, M. M.; FREIRE, A. P. Accessibility of mobile applications: Evaluation by users with visual impairment and by automated tools. In: **Proceedings of the 19th Brazilian Symposium on Human Factors in Computing Systems**. New York, NY, USA: Association for Computing Machinery, 2020. (IHC '20). ISBN 9781450381727. Available: <https://doi.org/10.1145/3424953.3426633>. Citation on page 145.

MELLOR, S. J.; CLARK, A. N.; FUTAGAMI, T. Focusguest editors' introduction. **IEEE SOFTWARE**, p. 5, 2003. Citation on page 144.

MELLOR, S. J.; CLARK, T.; FUTAGAMI, T. Model-driven development: guest editors' introduction. IEEE Software, 20 (5). pp. 14-18. ISSN 0740-7459. **IEEE software**, v. 20, n. 5, p. 14–18, 2003. Publisher: IEEE Computer Society. Citation on page 144.

MELLOR, S. J.; SCOTT, K.; UHL, A.; WEISE, D. **MDA distilled: principles of Model-Driven Architecture**. 1st. ed. Boston: Addison-Wesley Professional, 2004. Citation on page 40.

MELOUK, M.; RHAZALI, Y.; YOUSSEF, H. An Approach for Transforming CIM to PIM up To PSM in MDA. **Procedia Computer Science**, v. 170, p. 869–874, Jan. 2020. ISSN 1877-0509. Available: <https://www.sciencedirect.com/science/article/pii/S1877050920305603>. Citation on page 145.

MICROSOFT. **Inclusive Microsoft Design**. 2016. Available: <https://download.microsoft.com/download/b/0/d/b0d4bf87-09ce-4417-8f28-d60703d672ed/inclusive_toolkit_manual_final.pdf>. Citations on pages 36, 39, and 85.

____. **Microsoft Design**. 2018. Available: <https://www.microsoft.com/design/inclusive/>. Citation on page 39.

____. **Accessibility fundamentals - Learn**. 2021. Access 31/08/2021. Available: <https://docs.microsoft.com/en-us/learn/paths/accessibility-fundamentals/>. Citation on page 39.

____. **Accessibility in Windows 10 - Windows apps | Microsoft Docs**. 2021. Access 31/08/2021. Available: <https://docs.microsoft.com/en-us/windows/apps/accessibility>. Citation on page 39.

MILLER, J.; MUKERJI, J. MDA Guide Version 1.0.1. p. 62, 2003. Citation on page 144.

MIÑÓN, R.; MORENO, L.; MARTÍNEZ, P.; ABASCAL, J. An approach to the integration of accessibility requirements into a user interface development method. **Science of Computer Programming**, v. 86, p. 58 – 73, 2014. ISSN 0167-6423. Available: <http://www.sciencedirect.com/science/article/pii/S0167642313001032>. Citations on pages 36, 61, 68, 69, 108, 109, and 143.

MIÑÓN, R.; PATERNò, F.; ARRUE, M.; ABASCAL, J. Integrating adaptation rules for people with special needs in model-based UI development process. **Universal Access in the Information Society**, v. 15, n. 1, p. 153–168, 2016. ISSN 1615-5297. Available: <http://dx.doi.org/10.1007/s10209-015-0406-3>. Citation on page 143.

MIRANDA, D. G.; ARAUJO, J. A framework for integrating web accessibility requirements in agile methodologies. 2020. Citation on page 145.

MITROVIC, N.; BOBED, C.; MENA, E. A review of user interface description languages for mobile applications. In: **Tenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies UBICOMM**. [S.l.: s.n.], 2016. Citations on pages 62 and 145.

MOLINA, A. I.; GIRALDO, W. J.; ORTEGA, M.; REDONDO, M. A.; COLLAZOS, C. A. Model-driven development of interactive groupware systems: Integration into the software development process. **Science of Computer Programming**, v. 89, Part C, p. 320 – 349, 2014. ISSN 0167-6423. Available: <http://www.sciencedirect.com/science/article/pii/S016764231400104X>. Citation on page 143.

MOLINA, F.; TOVAL, A. Integrating usability requirements that can be evaluated in design time into Model Driven Engineering of Web Information Systems. **Advances in Engineering Software**, v. 40, n. 12, p. 1306 – 1317, 2009. ISSN 0965-9978. Available: <http://www.sciencedirect.com/science/article/pii/S096599780900043X>. Citations on pages 29 and 144.

MORENO, L.; MARTINEZ, P. The harmonization of accessibility standards for public policies. **Computer**, IEEE, v. 52, n. 7, p. 57–66, 2019. Citations on pages 28, 39, and 145.

MORENO, L.; THESE, P. D. Awa, methodological framework in the accessibility domain for web application development. **Universidad Carlos III de Madrid**, 2010. Available: <http://www.sigaccess.org/2010/03/awa-methodological-framework-in-the-accessibility-domain-for-web-application-development/>. Citations on pages 61 and 144.

MORENO, L.; VALENCIA, X.; PÉREZ, J.; ARRUE, M. An exploratory study of web adaptation techniques for people with low vision. **Universal Access in the Information Society**, v. 20, n. 2, p. 223–237, 2021. ISSN 1615-5289. Citation on page 146.

MORRIS, J. **Android user interface development**. [S.l.]: Packt Publishing Birmingham, 2011. Citation on page 92.

NASCIMENTO, M. D. d.; BRANDãO, A. A. F.; BRANDãO, L. d. O.; OLIVEIRA, F. C. d. M. B. Overcoming Accessibility Barriers for People with Severe Vision Impairment in Web-based Learning Environments: A Literature Review. In: **2019 IEEE Frontiers in Education Conference (FIE)**. [S.l.: s.n.], 2019. p. 1–8. ISSN: 2377-634X. Citation on page 145.

NOUREEN, A.; AMJAD, A.; AZAM, F. Model driven architecture-issues, challenges and future directions. **J. Softw.**, v. 11, n. 9, p. 924–933, 2016. Available: <http://www.jsoftware.us/index.php?m=content&c=index&a=show&catid=172&id=2678>. Citation on page 145.

NÚÑEZ, M.; BONHAURE, D.; GONZÁLEZ, M.; CERNUZZI, L. A model-driven approach for the development of native mobile applications focusing on the data layer. **Journal of Systems and Software**, v. 161, p. 110489, Mar. 2020. ISSN 0164-1212. Available: <https://www.sciencedirect.com/science/article/pii/S0164121219302638>. Citation on page 145.

OLIVEIRA, R.; SILVA, L.; LEITE, J. C. S. P.; MOREIRA, A. Eliciting Accessibility Requirements an Approach Based on the NFR Framework. In: **Proceedings of the 31st Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2016. (SAC '16), p. 1276–1281. ISBN 978-1-4503-3739-7. Available: <http://doi.acm.org/10.1145/2851613.2851759>. Citations on pages 29 and 143.

OLIVEIRA, R. C. de; FREIRE, A. P.; PAIVA, D. M. B.; CAGNIN, M. I.; RUBINSZTEJN, H. A framework to facilitate the implementation of technical aspects of web accessibility. In: STEPHANIDIS, C.; ANTONA, M. (Ed.). **Universal Access in Human-Computer Interaction. Design for All and Accessibility Practice**. Cham: Springer International Publishing, 2014. p. 3–13. ISBN 978-3-319-07509-9. Citations on pages 58 and 144.

OLIVEIRA, R. F. de; MOURA, A. M. da M.; LEITE, J. C. S. P. Reengineering for accessibility: A strategy based on software awareness. In: **Proceedings of the 17th Brazilian Symposium on Software Quality**. [s.n.], 2018. p. 180–189. Available: <https://dl.acm.org/doi/abs/10.1145/3275245.3275265>. Citation on page 145.

OMG. **Documents Associated With Interaction Flow Modeling Language - IFML**. 2015. Available: <http://www.omg.org/spec/IFML/1.0/>. Accessed: 19 Set. 2020. Citations on pages 29, 42, 43, 44, 46, 47, 62, 75, and 92.

____. **IFML - The standard Interaction Flow Modeling Language**. 2015. Available: <http://www.ifml.org>. Accessed: 19 Set. 2020. Citations on pages 42, 44, and 47.

____. **An OMG® Executable UML® PublicationSemantics of a Foundational Subset for Executable UML Models (fUML)**. 2020. Available: <https://www.omg.org/spec/FUML/1.5/Beta1/PDF>. Accessed: 2 Jun. 2020. Citation on page 47.

ORDOÑEZ, K.; HILERA, J.; CUEVA, S. Model-driven development of accessible software: a systematic literature review. **Universal Access in the Information Society**, Sep. 2020. ISSN 1615-5297. Available: <https://doi.org/10.1007/s10209-020-00751-6>. Citations on pages 30, 38, 39, and 145.

O'NEILL, J. L. Accessibility for All Abilities: How Universal Design, Universal Design for Learning, and Inclusive Design Combat Inaccessibility and Ableism. **Journal of Open Access to Law**, v. 9, n. 1, p. 15–15, Feb. 2021. ISSN 2372-7152. Number: 1. Available: <https://ojs.law.cornell.edu/index.php/joal/article/view/107>. Citations on pages 36, 85, and 146.

PACIELLO, M. **Web accessibility for people with disabilities**. 1. ed. Boca Raton, FL USA: CRC Press Taylor & Francis Group, 2000. Citation on page 28.

PAIVA, D. M. B.; FREIRE, A. P.; FORTES, R. P. d. M. Accessibility and Software Engineering Processes: A Systematic Literature Review. **Journal of Systems and Software**, v. 171, p. 110819, 2021. ISSN 0164-1212. Available: <https://www.sciencedirect.com/science/article/pii/S0164121220302168>. Citations on pages 29 and 146.

PANACH, J. I.; AQUINO, N.; PASTOR, O. A proposal for modelling usability in a holistic MDD method. **Science of Computer Programming**, v. 86, p. 74 – 88, 2014. ISSN 0167-6423. Available: <http://www.sciencedirect.com/science/article/pii/S0167642313001573>. Citations on pages 29 and 143.

PANACH, J. I.; DIESTE, O.; MARÍN, B.; ESPAÑA, S.; VEGAS, S.; PASTOR, O.; JURISTO, N. Evaluating Model-Driven Development Claims with Respect to Quality: A Family of Experiments. **IEEE Transactions on Software Engineering**, v. 47, n. 1, p. 130–145, Jan. 2021. ISSN 1939-3520. Conference Name: IEEE Transactions on Software Engineering. Citation on page 146.

PANACH, J. I.; JURISTO, N.; VALVERDE, F.; PASTOR, O. A framework to identify primitives that represent usability within Model-Driven Development methods. **Information and Software Technology**, v. 58, p. 338 – 354, 2015. ISSN 0950-5849. Available: <http://www.sciencedirect.com/science/article/pii/S0950584914001566>. Citation on page 143.

PATEL, R.; BRETON, P.; BAKER, C. M.; EL-GLALY, Y. N.; SHINOHARA, K. Why software is not accessible: Technology professionals' perspectives and challenges. In: . New York, NY, USA: Association for Computing Machinery, 2020. (CHI EA '20), p. 1–9. ISBN 9781450368193. Available: <https://doi.org/10.1145/3334480.3383103>. Citations on pages 28, 29, 30, 39, and 145.

PATERNO', F.; SANTORO, C.; SPANO, L. D. Maria: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. **ACM Trans. Comput.-Hum. Interact.**, Association for Computing Machinery, New York, NY, USA, v. 16, n. 4, Nov. 2009. ISSN 1073-0516. Available: <https://doi.org/10.1145/1614390.1614394>. Citations on pages 62, 92, and 144.

PEISSNER, M.; HÄBE, D.; JANSSEN, D.; SELLNER, T. MyUI: Generating Accessible User Interfaces from Multimodal Design Patterns. In: **Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems**. New York, NY, USA: ACM, 2012. (EICS '12), p. 81–90. ISBN 978-1-4503-1168-7. Available: <http://doi.acm.org/10.1145/2305484.2305500>. Citation on page 143.

PEREIRA, L. S.; ARCHAMBAULT, D. Web widgets barriers for visually impaired users. **Studies in health technology and informatics**, v. 242, p. 836–842, 2017. Citation on page 145.

PETRASCH, R. Model Based User Interface Development with HCI Patterns: Variatio Delectat. In: **Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems**. New York, NY, USA: ACM, 2010. (PEICS '10), p. 10–11. ISBN 978-1-4503-0246-3. Available: <http://doi.acm.org/10.1145/1824749.1824752>. Citation on page 144.

PIMAR. 2021. Available: <https://www.uni-marburg.de/de/fb12/arbeitsgruppen/swt/projekte/pimar>. Citation on page 146.

PLANAS, E.; DANIEL, G.; BRAMBILLA, M.; CABOT, J. Towards a model-driven approach for multiexperience AI-based user interfaces. **Software and Systems Modeling**, v. 20, n. 4, p. 997–1009, Aug. 2021. ISSN 1619-1374. Available: <https://doi.org/10.1007/s10270-021-00904-y>. Citation on page 146.

QUEIROZ, R.; MARQUES, A. B.; LOPES, A.; OLIVEIRA, E.; CONTE, T. Evaluating usability of ifml models: How usability is perceived and propagated. In: **Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems**. [S.l.: s.n.], 2018. p. 1–10. Citations on pages 64 and 145.

RADEKE, F.; FORBRIG, P.; SEFFAH, A.; SINNIG, D. PIM Tool: Support for Pattern-Driven and Model-Based UI Development. In: CONINX, K.; LUYTEN, K.; SCHNEIDER, K. A. (Ed.). **Task Models and Diagrams for Users Interface Design: 5th International Workshop, TAMODIA 2006, Hasselt, Belgium, October 23-24, 2006. Revised Papers**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 82–96. ISBN 978-3-540-70816-2. DOI: 10.1007/978-3-540-70816-2_7. Available: <http://dx.doi.org/10.1007/978-3-540-70816-2_7>. Citation on page 144.

RAMASWAMY, A.; MONSUEZ, B.; TAPUS, A. Model Driven Software Development for Human-machine Interaction Systems. In: **Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction**. New York, NY, USA: ACM, 2014. (HRI '14), p. 270–271. ISBN 978-1-4503-2658-2. Available: <http://doi.acm.org/10.1145/2559636.2559824>. Citation on page 143.

RANEBURGER, D.; MEIXNER, G.; BRAMBILLA, M. Platform-Independence in Model-Driven Development of Graphical User Interfaces for Multiple Devices. In: CORDEIRO, J.; SINDEREN, M. van (Ed.). **Software Technologies: 8th International Joint Conference, ICSOFT 2013, Reykjavik, Iceland, July 29-31, 2013, Revised Selected Papers**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. p. 180–195. ISBN 978-3-662-44920-2. DOI: 10.1007/978-3-662-44920-2_12. Available: <http://dx.doi.org/10.1007/978-3-662-44920-2_12>. Citation on page 143.

RASHEED, Y.; AZAM, F.; ANWAR, M. W.; TUFAIL, H. A Model-Driven Approach for Creating Storyboards of Web Based User Interfaces. In: **Proceedings of the 2019 7th International Conference on Computer and Communications Management**. New York, NY, USA: Association for Computing Machinery, 2019. (ICCCM 2019), p. 169–173. ISBN 978-1-4503-7195-7. Available: <https://doi.org/10.1145/3348445.3348465>. Citation on page 145.

RHAZALI, Y.; HADI, Y.; MOULOUDI, A. A methodology for transforming CIM to PIM through UML: From business view to information system view. In: **2015 Third World Conference on Complex Systems (WCCS)**. [S.l.: s.n.], 2015. p. 1–6. Citation on page 144.

____. A model transformation in MDA from CIM to PIM represented by web models through SoaML and IFML. In: **2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)**. [S.l.: s.n.], 2016. p. 116–121. Citation on page 143.

____. A new methodology CIM to PIM transformation resulting from an analytical survey. In: **2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)**. [S.l.: s.n.], 2016. p. 266–273. Citation on page 145.

RIBEIRO, A. N.; ARAÚJO, C. R. An Automated Model Based Approach to Mobile UI Specification and Development. In: KUROSU, M. (Ed.). **Human-Computer Interaction. Theory, Design, Development and Practice : 18th International Conference, HCI International 2016, Toronto, ON, Canada, July 17-22, 2016. Proceedings, Part I**. Cham: Springer International Publishing, 2016. p. 523–534. ISBN 978-3-319-39510-4. DOI: 10.1007/978-3-319-39510-4_48. Available: <http://dx.doi.org/10.1007/978-3-319-39510-4_48>. Citation on page 143.

RIEGER, C.; KUCHEN, H. A process-oriented modeling approach for graphical development of mobile business apps. **Computer Languages, Systems & Structures**, v. 53, p. 43–58, Sep. 2018. ISSN 1477-8424. Available: <https://www.sciencedirect.com/science/article/pii/S1477842417301215>. Citations on pages 66, 67, 68, and 145.

____. A Model-Driven Cross-Platform App Development Process for Heterogeneous Device Classes. In: . [s.n.], 2019. ISBN 978-0-9981331-2-6. Accepted: 2019-01-03T01:00:48Z. Available: <http://scholarspace.manoa.hawaii.edu/handle/10125/60180>. Citation on page 145.

____. A process-oriented modeling approach for graphical development of mobile business apps. **Computer Languages, Systems & Structures**, v. 53, p. 43–58, Jan. 2019. ISSN 1477-8424. Available: <https://www.sciencedirect.com/science/article/pii/S1477842417301215>. Citations on pages 66 and 68.

RIEGER, C.; LUCRéDIO, D.; FORTES, R. P. M.; KUCHEN, H.; DIAS, F.; DUARTE, L. A model-driven approach to cross-platform development of accessible business apps. In: **Proceedings of the 35th Annual ACM Symposium on Applied Computing**. New York, NY, USA: Association for Computing Machinery, 2020. (SAC '20), p. 984–993. ISBN 9781450368667. Available: <https://doi.org/10.1145/3341105.3375765>. Citations on pages 28, 30, 39, 107, and 145.

RODRÍGUEZ, F. D.; ACUÑA, S. T.; JURISTO, N. Design and programming patterns for implementing usability functionalities in web applications. **Journal of Systems and Software**, v. 105, p. 107 – 124, 2015. ISSN 0164-1212. Available: <http://www.sciencedirect.com/science/article/pii/S0164121215000795>. Citation on page 143.

ROSA, J. R. d. S.; VALENTIM, N. M. C. Accessibility, usability and user experience design for visually impaired people: a systematic mapping study. In: **Proceedings of the 19th Brazilian Symposium on Human Factors in Computing Systems**. Diamantina Brazil: ACM, 2020. p. 1–10. ISBN 978-1-4503-8172-7. Available: <https://dl.acm.org/doi/10.1145/3424953.3426626>. Citations on pages 30 and 145.

ROUBI, S.; ERRAMDANI, M.; MBARKI, S. A model driven approach to generate graphical user interfaces for Rich Internet Applications using Interaction Flow Modeling Language. In: **2015 15th International Conference on Intelligent Systems Design and Applications (ISDA)**. [S.l.: s.n.], 2015. p. 272–276. Citation on page 143.

____. Extending graphical part of the Interaction Flow Modeling Language to Generate Rich Internet Graphical User Interfaces. In: **2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)**. Rome, Italy: IEEE, 2016. p. 161–167. Citations on pages 64, 65, 68, and 145.

____. Modeling and generating graphical user interface for MVC Rich Internet Application using a model driven approach. In: **2016 International Conference on Information Technology for Organizations Development (IT4OD)**. [S.l.: s.n.], 2016. p. 1–6. Citation on page 143.

SALBER, D.; DEY, A. K.; ABOWD, G. D. The context toolkit: Aiding the development of context-enabled applications. In: **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems**. New York, NY, USA: Association for Computing Machinery, 1999. (CHI '99), p. 434–441. ISBN 0201485591. Available: <https://doi.org/10.1145/302979.303126>. Citations on pages 62, 92, and 144.

SÁNCHEZ-GORDÓN, M.-L.; MORENO, L. Toward an Integration of Web Accessibility into Testing Processes. **Procedia Computer Science**, Elsevier Masson SAS, v. 27, n. Dsai 2013, p. 281–291, 2014. ISSN 18770509. Available: <http://www.sciencedirect.com/science/article/pii/S1877050914000337>. Citation on page 144.

SANCHEZ-GORDON, S.; SÁNCHEZ-GORDÓN, M.; YILMAZ, M.; O'CONNOR, R. V. Integration of accessibility design patterns with the software implementation process of iso/iec 29110. **Journal of Software: Evolution and Process**, Wiley Online Library, v. 31, n. 1, p. e1987, 2019. Citation on page 145.

SAUER, S. Applying Meta-Modeling for the Definition of Model-Driven Development Methods of Advanced User Interfaces. In: HUSSMANN, H.; MEIXNER, G.; ZUEHLKE, D. (Ed.). **Model-Driven Development of Advanced User Interfaces**. Berlin, Heidelberg: Springer, 2011, (Studies in Computational Intelligence). p. 67–86. ISBN 978-3-642-14562-9. Available: <https://doi.org/10.1007/978-3-642-14562-9_4>. Citation on page 144.

SCHMIDT, D. Guest Editor's Introduction: Model-Driven Engineering. **Computer**, v. 39, n. 2, p. 25–31, 2006. Citation on page 40.

SCHMIDT, D. C. Model-driven engineering. Citeseer, 2006. Citation on page 144.

SEBASTIÁN-LOMBRAÑA, A.; GUERRA, E.; LARA, J. d. Positioning-Based Domain-Specific Modelling through Mobile Devices. In: **2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)**. [S.l.: s.n.], 2020. p. 150–157. Citation on page 145.

SEFFAH, A. Hci design patterns as a building block in model-driven engineering. In: ____. **Patterns of HCI Design and HCI Design of Patterns: Bridging HCI Design and Model-Driven Software Engineering**. Cham: Springer International Publishing, 2015. p. 35–58. ISBN 978-3-319-15687-3. Available: <https://doi.org/10.1007/978-3-319-15687-3_3>. Citation on page 143.

SEISSLER, M.; MEIXNER, G.; BREINER, K. Using HCI Patterns within the Model-Based Development of Run-Time Adaptive User Interfaces. **IFAC Proceedings Volumes**, v. 43, n. 13, p. 477 – 482, 2010. ISSN 1474-6670. Available: <http://www.sciencedirect.com/science/article/pii/S1474667015325799>. Citation on page 144.

SELIC, B. The pragmatics of model-driven development. **IEEE Software**, v. 20, n. 5, p. 19–25, Sep. 2003. ISSN 1937-4194. Conference Name: IEEE Software. Citation on page 144.

SENDALL, S.; KOZACZYNSKI, W. Model transformation: the heart and soul of model-driven software development. **IEEE Software**, v. 20, n. 5, p. 42–45, Sep. 2003. ISSN 1937-4194. Conference Name: IEEE Software. Citation on page 144.

SHAMSUJJOHA, M.; GRUNDY, J.; LI, L.; KHALAJZADEH, H.; LU, Q. Developing mobile applications via Model Driven Development: A Systematic Literature Review. **Information and Software Technology**, p. 106693, Jul. 2021. ISSN 0950-5849. Available: <https://www.sciencedirect.com/science/article/pii/S0950584921001488>. Citation on page 146.

SHINOHARA, K.; WOBBROCK, J. O.; PRATT, W. Incorporating Social Factors in Accessible Design. In: **Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility**. New York, NY, USA: Association for Computing Machinery, 2018. (ASSETS '18), p. 149–160. ISBN 978-1-4503-5650-3. Available: <https://doi.org/10.1145/3234695.3236346>. Citation on page 145.

SHIROGANE, J. Support method to Elicit Accessibility Requirements. In: ZOWGHI, D.; JIN, Z. (Ed.). **Requirements Engineering**. Berlin, Heidelberg: Springer, 2014. (Communications in Computer and Information Science), p. 210–223. ISBN 978-3-662-43610-3. Citations on pages 57 and 143.

SIDI. **Mobile Accessibility - Guide to the Development of Accessible Mobile Applications**. 2019. Access 31/08/2021. Available: <http://www.sidi.org.br/guiadeacessibilidade/en/index.html>. Citations on pages 39 and 88.

____. **Guide to the Development of Accessible Mobile Applications**. 2021. Available: <http://www.sidi.org.br/guiadeacessibilidade/en/index.html>. Citations on pages 88 and 146.

SIEGEL, J. **MDA Guide, revision 2.0**. 2014. Available: <http://www.omg.org/cgi-bin/doc?ormsc/14-06-01>. Citations on pages 41 and 47.

SILVA, A. R. d. Model-driven engineering: A survey supported by the unified conceptual model. **Computer Languages, Systems & Structures**, v. 43, p. 139 – 155, 2015. ISSN 1477-8424. Available: <http://www.sciencedirect.com/science/article/pii/S1477842415000408>. Citation on page 143.

SILVA, A. R. da. Rigorous Specification of Use Cases with the RSL Language. **International Conference on Information Systems Development (ISD)**, Dec. 2019. Available: <https://aisel.aisnet.org/isd2014/proceedings2019/ISDMethodologies/14>. Citation on page 145.

SILVA, A. R. da; PAIVA, A. C. R.; SILVA, V. E. R. da. A Test Specification Language for Information Systems Based on Data Entities, Use Cases and State Machines. In: HAMMOUDI, S.; PIRES, L. F.; SELIC, B. (Ed.). **Model-Driven Engineering and Software Development**. Cham: Springer International Publishing, 2019. p. 455–474. ISBN 978-3-030-11030-7. Citation on page 145.

SILVA, C.; ELER, M. M.; FRASER, G. A survey on the tool support for the automatic evaluation of mobile accessibility. In: **Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion**. Thessaloniki Greece: ACM, 2018. p. 286–293. ISBN 978-1-4503-6467-6. Available: <https://dl.acm.org/doi/10.1145/3218585.3218673>. Citation on page 145.

SILVA, J. de Sousa e; GONçALVES, R.; PEREIRA, A. Accessibility in the software life cycle a maieutic exercise in software engineering. In: . [S.l.: s.n.], 2017. p. 1–5. Citations on pages 39 and 145.

SILVA, J. S. e.; GONçALVES, R.; BRANCO, F.; PEREIRA, A.; AU-YONG-OLIVEIRA, M.; MARTINS, J. Accessible software development: a conceptual model proposal. **Universal Access in the Information Society**, v. 18, n. 3, p. 703–716, Aug. 2019. ISSN 1615-5297. Available: <https://doi.org/10.1007/s10209-019-00688-5>. Citations on pages 58, 67, 68, and 145.

SKJERVE, R.; GIANNOUMIS, G. A.; NASEEM, S. An intersectional perspective on web accessibility. In: ____. **Designing Around People: CWUAAT 2016**. Cham: Springer International Publishing, 2016. p. 13–22. ISBN 978-3-319-29498-8. Available: <http://dx.doi.org/10.1007/978-3-319-29498-8_2>. Citations on pages 28, 39, and 145.

SNIDER, S.; II, W. L. S.; TREWIN, S. Accessibility information needs in the enterprise. **ACM Trans. Access. Comput.**, Association for Computing Machinery, New York, NY, USA, v. 12, n. 4, Jan. 2020. ISSN 1936-7228. Available: <https://doi.org/10.1145/3368620>. Citations on pages 39 and 145.

SOTTET, J. S.; VAGNER, A. Defining Domain Specific Transformations in Human-Computer interfaces development. In: **2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)**. [S.l.: s.n.], 2014. p. 246–253. Citation on page 143.

SOUZA, R. X. de O.; OLIVEIRA, A. A. de; NASCIMENTO, R. P. C. do. ModelER: Abordagem Baseada Em Modelos Aplicada Ao Processo De Elicitação De Requisitos. In: **Proceedings of the 7th Euro American Conference on Telematics and Information Systems**. New York, NY, USA: ACM, 2014. (EATIS '14), p. 11:1–11:7. ISBN 978-1-4503-2435-9. Available: <http://doi.acm.org/10.1145/2590651.2590663>. Citation on page 143.

SPRAGUE, P.; SCHAHCZENSKI, C. Abstraction the key to cs 1. **Journal of Computing Sciences in Colleges**, v. 17, n. 3, p. 211–218, 2002. Citation on page 144.

STATISTA. **Topic: Internet usage worldwide**. 2021. [Online; accessed 15. Mar. 2021]. Available: <https://www.statista.com/topics/1145/internet-usage-worldwide>. Citation on page 27.

STEEN-HANSEN, L.; FAGERNES, S. The Importance of Process-Oriented Accessibility Guidelines for Web Developers. **Universal Design 2016: Learning from the Past, Designing for the Future**, p. 439–449, 2016. Publisher: IOS Press. Available: <https://ebooks.iospress.nl/doi/10.3233/978-1-61499-684-2-439>. Citations on pages 57, 67, 68, and 145.

STEINBERG, D.; BUDINSKY, F.; PATERNOSTRO, M.; MERKS, E. **EMF: Eclipse Modeling Framework**. 2rd. ed. USA: Addison-Wesley Professional, 2008. Citation on page 41.

STEINEBACH, T. info:eu-repo/semantics/masterThesis, **Web Accessibility : incorporating user requirements into a guide for usable web accessibility**. 2020. Publisher: University of Twente. Available: <http://essay.utwente.nl/82776/>. Citation on page 145.

THANH-DIANE, N.; VANDERDONCKT, J.; SEFFAH, A. UIPLML: Pattern-based engineering of user interfaces of multi-platform systems. In: **2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)**. [S.l.: s.n.], 2016. p. 1–12. Citation on page 143.

TOMANDL, J.; HEINMÜLLER, S.; SELB, M.; GRAESSEL, E.; FREIBERGER, E.; KÜHLEIN, T.; HUEBER, S.; BOOK, S.; GOTTHARDT, S. Laying the foundation for a Core Set of the International Classification of Functioning, Disability and Health for community-dwelling older adults in primary care: relevant categories of their functioning from the research perspective, a scoping review. **BMJ Open**, v. 11, n. 2, p. e037333, Feb. 2021. ISSN 2044-6055, 2044-6055. Publisher: British Medical Journal Publishing Group Section: Geriatric medicine. Available: <https://bmjopen.bmj.com/content/11/2/e037333>. Citation on page 146.

TORCHIANO, M.; TOMASSETTI, F.; RICCA, F.; TISO, A.; REGGIO, G. Benefits from modelling and mdd adoption: expectations and achievements. In: **Proceedings of the Second Edition of the International Workshop on Experiences and Empirical Studies in Software Modelling**. [S.l.: s.n.], 2012. p. 1–6. Citation on page 144.

TSADIMAS, A.; NIKOLAIDOU, M.; ANAGNOSTOPOULOS, D. Extending SysML to Explore Non-functional Requirements: The Case of Information System Design. In: **Proceedings of the 27th Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2012. (SAC '12), p. 1057–1062. ISBN 978-1-4503-0857-1. Available: <http://doi.acm.org/10.1145/2245276.2231941>. Citation on page 143.

TUFAIL, H.; AZAM, F.; ANWAR, M. W.; QASIM, I. Model-driven development of mobile applications: A systematic literature review. In: IEEE. **2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)**. [S.l.], 2018. p. 1165–1171. Citation on page 145.

UMUHOZA, E.; BRAMBILLA, M. Model Driven Development Approaches for Mobile Applications: A Survey. In: YOUNAS, M.; AWAN, I.; KRYVINSKA, N.; STRAUSS, C.; THANH, D. v. (Ed.). **Mobile Web and Intelligent Information Systems**. Cham: Springer International Publishing, 2016. (Lecture Notes in Computer Science), p. 93–107. ISBN 978-3-319-44215-0. Citation on page 145.

USMAN, M.; IQBAL, M. Z.; KHAN, M. U. A model-driven approach to generate mobile applications for multiple platforms. In: IEEE. **2014 21st Asia-Pacific Software Engineering Conference**. [S.l.], 2014. v. 1, p. 111–118. Citation on page 144.

VAUPEL, S.; TAENTZER, G.; GERLACH, R.; GUCKERT, M. Model-driven development of mobile applications for Android and iOS supporting role-based app variability. **Software & Systems Modeling**, v. 17, n. 1, p. 35–63, Feb. 2018. ISSN 1619-1374. Available: <https://doi.org/10.1007/s10270-016-0559-4>. Citations on pages 62, 68, and 145.

VILAIN, P.; SCHWABE, D.; SOUZA, C. S. de. A Diagrammatic Tool for Representing User Interaction in UML. In: EVANS, A.; KENT, S.; SELIC, B. (Ed.). **UML 2000 — The Unified Modeling Language: Advancing the Standard Third International Conference York, UK, October 2–6, 2000 Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. p. 133–147. ISBN 978-3-540-40011-0. DOI: 10.1007/3-540-40011-7_10. Available: <http://dx.doi.org/10.1007/3-540-40011-7_10>. Citation on page 144.

VÖLTER, M. Best practices for dsls and model-driven development. **Journal of Object Technology**, Citeseer, v. 8, n. 6, p. 79–102, 2009. Citation on page 144.

W3C. **Start with Accessibility - Education & Outreach**. 2014. [Online; accessed 2. Feb. 2021]. Available: <https://www.w3.org/WAI/EO/wiki/Start_with_Accessibility>. Citation on page 29.

____. **W3C Issues Improved Accessibility Guidance for Websites and Applications**. 2018. Available: <https://www.w3.org/2018/06/pressrelease-wcag21.html.en>. Citation on page 37.

____. **Web Content Accessibility Guidelines (WCAG) 2.1**. 2018. [Online; accessed 2. Feb. 2021]. Available: <https://www.w3.org/TR/WCAG21/>. Citations on pages 38 and 88.

____. **Introduction to Web Accessibility**. 2019. Available: <https://www.w3.org/WAI/fundamentals/accessibility-intro/>. Citations on pages 28 and 36.

____. **Home**. 2021. [Online; accessed 2. Feb. 2021]. Available: <https://www.w3.org/WAI>. Citation on page 37.

____. **Mobile Accessibility at W3C**. 2021. Available: <https://www.w3.org/WAI/standards-guidelines/mobile/>. Citation on page 38.

____. **Web Content Accessibility Guidelines (WCAG) Overview**. 2021. [Online; accessed 2. Feb. 2021]. Available: <https://www.w3.org/WAI/standards-guidelines/wcag>. Citations on pages 37 and 38.

WAKIL, K.; JAWAWI, D. N. Analyzing interaction flow modeling language in web development lifecycle. **International Journal of Advanced Computer Science and Applications**, v. 8, n. 1, p. 286–293, 2017. Publisher: SCIENCE & INFORMATION SAI ORGANIZATION LTD 19 BOLLING RD, BRADFORD, WEST. Citations on pages 64, 65, and 145.

WAKIL, K.; JAWAWI, D. N. A. Extensibility Interaction Flow Modeling Language Metamodels to Develop New Web Application Concerns. **Kurdistan Journal of Applied Research**, v. 2, n. 3, p. 172–177, Aug. 2017. ISSN 2411-7706. Number: 3. Available: <http://www.spu.edu.iq/kjar/index.php/kjar/article/view/91>. Citation on page 145.

WAKIL, K.; JAWAWI, D. N. A.; RACHMAT, H. Enhancing Interaction Flow Modeling Language Metamodels for Designing Features of Rich Internet Applications. **International Journal of Integrated Engineering**, v. 10, n. 6, Nov. 2018. ISSN 2600-7916. Number: 6. Available: <https://publisher.uthm.edu.my/ojs/index.php/ijie/article/view/2830>. Citations on pages 65, 68, and 145.

WAZLAWICK, R. **Metodologia de pesquisa para ciência da computação**. [S.l.]: Elsevier Brasil, 2015. Citation on page 32.

WEBAIM. **WebAIM: Web Accessibility In Mind**. 2021. Access 31/08/2021. Available: <https://webaim.org/>. Citations on pages 39 and 146.

WENDLER, S.; PHILIPPOW, I. Requirements for a Definition of Generative User Interface Patterns. In: KUROSU, M. (Ed.). **Human-Computer Interaction. Human-Centred Design Approaches, Methods, Tools, and Environments: 15th International Conference, HCI International 2013, Las Vegas, NV, USA, July 21-26, 2013, Proceedings, Part I**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 510–520. ISBN 978-3-642-39232-0. DOI: 10.1007/978-3-642-39232-0_55. Available: <http://dx.doi.org/10.1007/978-3-642-39232-0_55>. Citation on page 143.

WHITNEY, M. Teaching Accessible Design: Integrating Accessibility Principles and Practices into an Introductory Web Design Course. **Information Systems Education Journal**, v. 18, n. 1, p. 4–13, Feb. 2020. ISSN 1545-679X. Publisher: Information Systems and Computing Academic Professionals. Available: <https://eric.ed.gov/?id=EJ1246240>. Citation on page 145.

WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: ACM. **Proceedings of the 18th international conference on evaluation and assessment in software engineering**. [S.l.], 2014. p. 38. Citations on pages 51 and 52.

WORLD, D. **Disabilities: Definition, Types and Models of Disability**. 2021. Available: <https://www.disabled-world.com/disability/types/>. Citations on pages 36 and 145.

World Health Organization *et al.* **International Classification of Functioning, Disability and Health (ICF)**. World Health Organization, 2001. [Online; accessed 02. Set. 2021] - Last Update: 2018. Available: <https://www.who.int/standards/classifications/international-classification-of-functioning-disability-and-health>. Citations on pages 36 and 85.

____. **World report on disability.** World Health Organization, 2011. Available: <http://www.who.int/disabilities/world_report/2011/en/>. Citation on page 28.

____. **World report on ageing and health 2015**. World Health Organization, 2015. Available: <http://www.who.int//ageing/events/world-report-2015-launch/en/>. Citation on page 28.

WPT. **99% dos sites do Brasil apresentam barreiras de navegação para pessoas com deficiência - WPT**. 2019. [Online; accessed 12. Mar. 2021]. Available: <https://mwpt.com.br/99-dos-sites-do-brasil-apresentam-barreiras-de-navegacao-para-pessoas-com-deficiencia>. Citation on page 28.

XIONG, J.; FARENC, C.; WINCKLER, M. Analyzing tool support for inspecting accessibility guidelines during the development process of web sites. In: SPRINGER. **WISE Workshops**. [S.l.], 2007. p. 470–480. Citations on pages 28, 39, and 144.

YAN, S.; RAMACHANDRAN, P. The current status of accessibility in mobile apps. **ACM Transactions on Accessible Computing (TACCESS)**, v. 12, n. 1, p. 1–31, 2019. Publisher: ACM New York, NY, USA. Citation on page 145.

YIGITBAS, E.; FISCHER, H.; KERN, T.; PAELKE, V. Model-Based Development of Adaptive UIs for Multi-channel Self-service Systems. In: SAUER, S.; BOGDAN, C.; FORBRIG, P.; BERNHAUPT, R.; WINCKLER, M. (Ed.). **Human-Centered Software Engineering: 5th IFIP WG 13.2 International Conference, HCSE 2014, Paderborn, Germany, September 16-18, 2014. Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. p. 267–274. ISBN 978-3-662-44811-3. DOI: 10.1007/978-3-662-44811-3_18. Available: <http://dx.doi.org/10.1007/978-3-662-44811-3_18>. Citation on page 143.

YIGITBAS, E.; JOVANOVIKJ, I.; BIERMEIER, K.; SAUER, S.; ENGELS, G. Integrated model-driven development of self-adaptive user interfaces. **Software and Systems Modeling**, v. 19, n. 5, p. 1057–1081, Sep. 2020. ISSN 1619-1374. Available: <https://doi.org/10.1007/s10270-020-00777-7>. Citations on pages 66, 68, and 145.

YIGITBAS, E.; MOHRMANN, B.; SAUER, S. Model-driven UI Development integrating HCI Patterns. **Large-scale and Model-based Interactive Systems - LMIS@ EICS**, p. 42 – 46, 2015. Citations on pages 66 and 68.

____. Model-driven UI Development integrating HCI Patterns. **Large-scale and Model-based Interactive Systems - LMIS@ EICS**, p. 42 – 46, 2015. Citation on page 144.

YIGITBAS, E.; SAUER, S. Engineering Context-Adaptive UIs for Task-Continuous Cross-Channel Applications. In: BOGDAN, C.; GULLIKSEN, J.; SAUER, S.; FORBRIG, P.; WINCKLER, M.; JOHNSON, C.; PALANQUE, P.; BERNHAUPT, R.; KIS, F. (Ed.). **Human-Centered and Error-Resilient Systems Development**. Cham: Springer International Publishing, 2016. (Lecture Notes in Computer Science), p. 281–300. ISBN 978-3-319-44902-9. Citations on pages 66, 68, and 143.

YIGITBAS, E.; SAUER, S.; ENGELS, G. Adapt-ui: An ide supporting model-driven development of self-adaptive uis. In: **Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems**. New York, NY, USA: Association for Computing Machinery, 2017. (EICS '17), p. 99–104. ISBN 9781450350839. Available: <https://doi.org/10.1145/3102113.3102144>. Citation on page 145.

ZACHAREWICZ, G.; DACLIN, N.; DOUMEINGTS, G.; HAIDAR, H. Model Driven Interoperability for System Engineering. **Modelling**, v. 1, n. 2, p. 94–121, Dec. 2020. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute. Available: <https://www.mdpi.com/2673-3951/1/2/7>. Citation on page 145.

ZEFERINO, N. V.; VILAIN, P. A Model-driven Approach for Generating Interfaces from User Interaction Diagrams. In: **Proceedings of the 16th International Conference on Information Integration and Web-based Applications & Services**. New York, NY, USA: ACM, 2014. (iiWAS '14), p. 474–478. ISBN 978-1-4503-3001-5. Available: <http://doi.acm.org/10.1145/2684200.2684326>. Citations on pages 60, 68, and 143.

ZOU, J.; XU, L.; YANG, M.; ZHANG, X.; YANG, D. Towards comprehending the non-functional requirements through Developers' eyes: An exploration of Stack Overflow using topic analysis. **Information and Software Technology**, v. 84, p. 19 – 32, 2017. ISSN 0950-5849. Available: <http://www.sciencedirect.com/science/article/pii/S0950584916304268>. Citation on page 143.

ZOUHAIER, L.; HLAOUI, Y. B.; AYED, L. J. B. A model driven approach for improving the generation of accessible user interfaces. In: **2015 10th International Joint Conference on Software Technologies (ICSOFT)**. [S.l.: s.n.], 2015. v. 2, p. 1–6. Citation on page 143.

____. Methodology for the Development of Accessible User Interfaces Based on Meta-Model Transformations: The Case of Blind Users. In: SAEED, K.; HOMENDA, W.; CHAKI, R. (Ed.). **Computer Information Systems and Industrial Management**. Cham: Springer International Publishing, 2017. (Lecture Notes in Computer Science), p. 73–84. ISBN 978-3-319-59105-6. Citation on page 145.

APPENDIX

A

REFERENCES SET

Table 16 – Initial set of Works gathered in snowballing review

1	Bin, Zhi and Xiaohong (2012)
2	Dias, Fortes and Masiero (2012)
3	González-Huerta *et al.* (2012)
4	Lima *et al.* (2012)
5	Peissner *et al.* (2012)
6	Tsadimas, Nikolaidou and Anagnostopoulos (2012)
7	Agustin and Barco (2013)
8	Buarque, Castro and Alencar (2013)
9	Kaindl (2013)
10	Khatter and Kalia (2013)
11	Wendler and Philippow (2013)
12	Bačíková and Porubän (2014)
13	Bourimi and Tesoriero (2014)
14	Brambilla, Mauri and Umuhoza (2014b)
15	Brambilla and Fraternali (2014b)
16	Branco, Cagnin and Paiva (2014)
17	Costa, Neto and Oliveira (2014)
18	Garćıa-Borgoñón *et al.* (2014)
19	Grillo and Fortes (2014)
20	Jones and Jia (2014)
21	Macik, Cerny and Slavik (2014)
22	Miñón *et al.* (2014)
23	Molina *et al.* (2014)
24	Panach, Aquino and Pastor (2014)
25	Ramaswamy, Monsuez and Tapus (2014)
26	Raneburger, Meixner and Brambilla (2014)
27	Shirogane (2014)
28	Sottet and Vagner (2014)
29	Souza, Oliveira and Nascimento (2014)
30	Yigitbas *et al.* (2014)
31	Zeferino and Vilain (2014)
32	Acerbis *et al.* (2015)
33	Agirre, Sagardui and Etxeberria (2015)
34	Ameller *et al.* (2015)
35	Antonelli, Silva and Fortes (2015)
36	Francese *et al.* (2015)
37	Gaouar, Benamar and Bendimerad (2015)
38	Gibbs *et al.* (2015)
39	Martins and Souza (2015)
40	Li *et al.* (2015)
41	Panach *et al.* (2015)
42	Rodríguez, Acuña and Juristo (2015)
43	Roubi, Erramdani and Mbarki (2015)
44	Seffah (2015)
45	Silva (2015)
46	Zouhaier, Hlaoui and Ayed (2015)
47	Agh and Ramsin (2016)
48	Channonthawat and Limpiyakorn (2016)
49	Delgado *et al.* (2016)
50	Desruelle *et al.* (2016)
51	Eckhardt, Vogelsang and Fernández (2016)
52	Forbrig and Saurin (2016)
53	Laaz and Mbarki (2016a)
54	Laaz and Mbarki (2016b)
55	Lumertz, Ribeiro and Duarte (2016)
56	Miñón *et al.* (2016)
57	Ribeiro and Araújo (2016)
58	Oliveira *et al.* (2016)
59	Rhazali, Hadi and Mouloudi (2016a)
60	Roubi, Erramdani and Mbarki (2016b)
61	Thanh-Diane, Vanderdonckt and Seffah (2016)
62	Yigitbas and Sauer (2016)
63	Giachetti *et al.* (2017)
64	Zou *et al.* (2017)

Source: Elaborated by the author.

Table 17 – Works gathered from Backward and Forward of snowballing review

Backward and Forward set	
1999	
1 Salber, Dey and Abowd (1999)	
2000	
2 Ceri, Fraternali and Bongio (2000)	
3 Chung *et al.* (2000)	
4 Vilain, Schwabe and Souza (2000)	
2002	
5 Sprague and Schahczenski (2002)	
2003	
6 Calvary *et al.* (2003)	
7 Duran-Limon *et al.* (2003)	
8 Greenfield and Short (2003)	
9 Ludewig (2003)	
10 Mellor, Clark and Futagami (2003a)	
11 Mellor, Clark and Futagami (2003b)	
12 Miller and Mukerji (2003)	
13 Sendall and Kozaczynski (2003)	
14 Selic (2003)	
2004	
15 Favre (2004)	
16 Limbourg *et al.* (2004)	
2005	
17 Clerckx, Luyten and Coninx (2005)	
18 Favre (2005)	
19 Kramer (2005)	
20 Lachgar and Abdali (2005)	
2006	
21 Bézivin (2006)	
22 Favre (2006)	
23 Hailpern and Tarr (2006)	
24 Schmidt (2006b)	
2007	
25 Ahmed and Ashraf (2007)	
26 Juristo, Moreno and Sanchez-Segura (2007)	
27 Kramer (2007)	
28 Martín *et al.* (2007)	
29 Radeke *et al.* (2007)	
30 Xiong, Farenc and Winckler (2007)	
2008	
31 Acerbis *et al.* (2008)	
32 Freire, Russo and Fortes (2008)	
33 Hazzan and Kramer (2008)	
2009	
34 Chung and Leite (2009)	
35 Guerrero-Garcia *et al.* (2009)	
36 Helms *et al.* (2009)	
37 Lowe and Pressman (2009)	
38 Lucrédio (2009)	
39 Molina and Toval (2009)	
40 Paterno', Santoro and Spano (2009)	
41 Völter (2009)	
2010	
42 Bustos, Martín and Cechich (2010)	
43 Breiner *et al.* (2010)	
44 Engel (2010)	
45 Forward, Lethbridge and Badreddin (2010)	
46 Petrasch (2010)	
47 Seissler, Meixner and Breiner (2010)	
48 Gajos, Weld and Wobbrock (2010)	
49 Maia (2010)	
50 Martín *et al.* (2010)	
51 Martín, Mazalu and Cechich (2010)	
52 Moreno and These (2010)	
2011	
53 Embley and Thalheim (2011)	
54 Linaje *et al.* (2011)	
55 Martín, Cechich and Rossi (2011)	
56 Sauer (2011)	
2012	
57 Ferreira and Silva (2012)	
58 Torchiano *et al.* (2012)	
2013	
59 Ammar and Mahfoudhi (2013)	
60 Gonçalves *et al.* (2013)	
61 Heitkötter, Majchrzak and Kuchen (2013b)	
62 Horkoff and Yu (2013)	
2014	
63 Brambilla and Fraternali (2014a)	
64 Davies *et al.* (2014)	
65 Dias (2014)	
66 Fogli, Provenza and Bernareggi (2014)	
67 Oliveira *et al.* (2014)	
68 Sánchez-Gordón and Moreno (2014)	
69 Usman, Iqbal and Khan (2014)	
2015	
70 Ammar, Trabelsi and Mahfoudhi (2015)	
71 Brajnik and Harper (2015)	
72 Brambilla (2015)	
73 Hentati *et al.* (2015)	
74 Rhazali, Hadi and Mouloudi (2015)	
75 Yigitbas, Mohrmann and Sauer (2015b)	

Continues on the next page

Table 17 – Backward and Forward set (continuation from previous page)

2016

76 Ammar, Trabelsi and Mahfoudhi (2016)
77 Benouda *et al.* (2016)
78 Calvo, Seyedarabi and Savva (2016)
79 Chmielewski *et al.* (2016)
80 Duarte *et al.* (2016)
81 Hentati *et al.* (2016)
82 Krainz, Feiner and Fruhmann (2016)
83 Mitrovic, Bobed and Mena (2016)
84 Noureen, Amjad and Azam (2016)
85 Rhazali, Hadi and Mouloudi (2016b)
86 Roubi, Erramdani and Mbarki (2016a)
87 Skjerve, Giannoumis and Naseem (2016)
88 Steen-Hansen and Fagernes (2016)
89 Umuhoza and Brambilla (2016)

2017

90 Abrahão *et al.* (2017)
91 Alulema, Iribarne and Criado (2017)
92 Benouda *et al.* (2017)
93 Brambilla, Umuhoza and Acerbis (2017)
94 Hentati *et al.* (2017)
95 Krainz and Miesenberger (2017b)
96 Lachgar and Abdali (2017)
97 Pereira and Archambault (2017)
98 Silva, Gonçalves and Pereira (2017)
99 Wakil and Jawawi (2017a)
100 Wakil and Jawawi (2017b)
101 Yigitbas, Sauer and Engels (2017)
102 Zouhaier, Hlaoui and Ayed (2017)

2018

103 Andrade *et al.* (2018)
104 Ballantyne *et al.* (2018)
105 Carvalho *et al.* (2018b)
106 Cheon and Barua (2018)
107 Dingsøyr *et al.* (2018)
108 Hamdani *et al.* (2018)
109 Hong (2018)
110 Huang *et al.* (2018)
111 Krainz, Miesenberger and Feiner (2018b)
112 Oliveira, Moura and Leite (2018)
113 Queiroz *et al.* (2018)
114 Rieger and Kuchen (2018)
115 Shinohara, Wobbrock and Pratt (2018)
116 Silva, Eler and Fraser (2018)
117 Tufail *et al.* (2018)
118 Vaupel *et al.* (2018)
119 Wakil, Jawawi and Rachmat (2018)

2019

120 Anjos *et al.* (2019)
121 Bouraoui and Gharbi (2019)
122 Bruneliere *et al.* (2019)
123 Fatima *et al.* (2019)
124 Gaggi, Quadrio and Bujari (2019)
125 Greco (2019)
126 Kawas, Vonessen and Ko (2019)
127 Laaz and Mbarki (2019)
128 Moreno and Martinez (2019)
129 Nascimento *et al.* (2019)
130 Rasheed *et al.* (2019)
131 Rieger and Kuchen (2019a)
132 Sanchez-Gordon *et al.* (2019)
133 Silva (2019)
134 Silva *et al.* (2019)
135 Silva, Paiva and Silva (2019)
136 Yan and Ramachandran (2019)

2020

137 Acosta-Vargas *et al.* (2020)
138 Anjos *et al.* (2020)
139 BBC (2020)
140 Bucchiarone *et al.* (2020)
141 Cao and Liu (2020)
142 De', Pandey and Pal (2020)
143 World (2021)
144 Gamito and Silva (2020)
145 Houtenville and Boege (2020)
146 Khaddam and Vanderdonckt (2020)
147 Mateus *et al.* (2020)
148 Melouk, Rhazali and Youssef (2020)
149 Miranda and Araujo (2020)
150 Núñez *et al.* (2020)
151 Ordoñez, Hilera and Cueva (2020)
152 Patel *et al.* (2020)
153 Rieger *et al.* (2020)
154 Rosa and Valentim (2020)
155 Sebastián-Lombraña, Guerra and Lara (2020)
156 Snider, II and Trewin (2020)
157 Steinebach (2020)
158 Whitney (2020)
159 Yigitbas *et al.* (2020)
160 Zacharewicz *et al.* (2020)

2021

161 Level Access (2021)
162 A11Y (2021)

Continues on the next page

Table 17 – Backward and Forward set (continuation from previous page)

163 Ali *et al.* (2021)	177 Moreno *et al.* (2021)
164 Azenkot, Hanley and Baker (2021)	178 O'Neill (2021)
165 Bi *et al.* (2021)	179 Paiva, Freire and Fortes (2021)
166 Blanco and Lucrédio (2021)	180 Panach *et al.* (2021)
167 Chiou, Alotaibi and Halfond (2021)	181 PIMAR (2021)
168 Dias, Duarte and Fortes (2021)	182 Planas *et al.* (2021)
169 Gharaat *et al.* (2021)	183 SIDI (2021)
170 Hayat *et al.* (2021)	184 Shamsujjoha *et al.* (2021)
171 IBM (2021)	185 Tomandl *et al.* (2021)
172 Inc. (2021)	186 WebAIM (2021)
173 Kaufhold *et al.* (2021)	**2022**
174 Khan *et al.* (2021)	187 Almeida, Bernardo and Lacerda (2022)
175 Kemp (2021)	
176 Leite *et al.* (2021)	

Source: Elaborated by the author.


ICMC
USP
SÃO CARLOS