Addressing Visual Impairments with Model-Driven Engineering: A Systematic Literature Review

JUDITH MICHAEL, Software Engineering, RWTH Aachen University, Germany and University of Regensburg, Germany
LUKAS NETZ and BERNHARD RUMPE, Software Engineering, RWTH Aachen University, Germany
INGO MUELLER, Monash Health, Australia
JOHN GRUNDY and SHAVINDRA WICKRAMATHILAKA, Monash University, Australia
HOURIEH KHALAJZADEH, Deakin University, Australia

Software applications often pose barriers for users with accessibility needs, e.g., visual impairments. Model-driven engineering (MDE), with its systematic nature of code derivation, offers systematic methods to integrate accessibility concerns into software development while reducing manual effort. This paper presents a systematic literature review on how MDE addresses accessibility for vision impairments. From 447 initially identified papers, 30 primary studies met the inclusion criteria. About two-thirds reference the Web Content Accessibility Guidelines (WCAG), yet their project-specific adaptions and end-user validations hinder wider adoption in MDE. The analyzed studies model user interface structures, interaction and navigation, user capabilities, requirements, and context information. However, only few specify concrete modeling techniques on how to incorporate accessibility needs or demonstrate fully functional systems. Insufficient details on MDE methods, i.e., transformation rules or code templates, hinder the reuse, generalizability, and reproducibility. Furthermore, limited involvement of affected users and limited developer expertise in accessibility contribute to weak empirical validation. Overall, the findings indicate that current MDE research insufficiently supports vision-related accessibility. Our paper concludes with a research agenda outlining how support for vision impairments can be more effectively embedded in MDE processes.

CCS Concepts: • **Software and its engineering** → **Software design engineering**; **Model-driven software engineering**; • **Human-centered computing** → **Accessibility design and evaluation methods**.

Additional Key Words and Phrases: Model-driven Engineering, Accessibility, Vision Impairments

ACM Reference Format:
Judith Michael, Lukas Netz, Bernhard Rumpe, Ingo Mueller, John Grundy, Shavindra Wickramathilaka, and Hourieh Khalajzadeh. 2026. Addressing Visual Impairments with Model-Driven Engineering: A Systematic Literature Review. 1, 1 (March 2026), 46 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' addresses: Judith Michael, judith.michael@ur.de, Software Engineering, RWTH Aachen University, Aachen, Germany and University of Regensburg, Regensburg, Germany; Lukas Netz, netz@se-rwth.de; Bernhard Rumpe, rumpe@se-rwth.de, Software Engineering, RWTH Aachen University, Aachen, Germany; Ingo Mueller, ingo.mueller@monashhealth.org, Monash Health, Melbourne, Australia; John Grundy, john.grundy@monash.edu; Shavindra Wickramathilaka, shavindra.wickramathilaka@monash.edu, Monash University, Melbourne, Australia; Hourieh Khalajzadeh, Deakin University, Melbourne, Australia, hkhalajzadeh@deakin.edu.au.

1 INTRODUCTION

Developers often overlook that modern software applications are not only technical solutions but are embedded in a larger socio-technical systems including diverse end users with heterogeneous needs. In this article, we focus on a particular aspect of accessibility needs, namely visual impairments. To give an example: When selecting the menu for lunch or reading a bus schedule, users with low vision or blind users have specific requirements for the technical applications used, e.g., showing larger text sizes and having more contrast for people with low vision or enabling the reading of texts with screen readers for blind users. This requires developers of such systems to consider these needs within the engineering of socio-technical systems; not only due to their strive for inclusion but also to fulfill legal requirements for providing accessible systems, i.e., the European accessibility act [34].

Ensuring the usability and accessibility of Graphical User Interfaces (GUIs) is particularly important for people with visual impairments [55]. GUIs are the main medium through which users interact with software applications, and their accessibility has a direct impact on the usability and comprehensibility of digital content. For people with visual impairments, inclusive GUIs facilitate equal access to information, promote independence, and social inclusion. Descriptions on how to cover needs for visual impairments are prominently featured in accessibility guidelines such as Web Content Accessibility Guidelines (WCAG) [98]. Accessibility as a non-functional requirement, however, is challenging to realize in a software system because its effects are widespread across the entire system [82].

Model-Driven Engineering (MDE) claims to have various advantages useful for addressing accessibility in applications [93]. This includes better reusability and portability, increasing development speed, increasing software quality due to uniform representations in the code, and being better maintainable when it comes to technological changes. Cross-cutting aspects, including UX and accessibility, can be changed in just one place – the models. When generating code, this change is then reflected consistently in many different places in the code. Many MDE methods have been adopted in practice [2, 19], e.g., for design automation [46], code generation [21, 57, 92], test automation [10, 80], automated verification and validation [23, 53, 72], or simulation [32, 49]. MDE enhances the software engineering process by increasing automation through the use of General Purpose Languages (GPLs) or Domain-Specific Languages (DSLs), as well as automated code generation, transformations, or interpretation. On top of MDE [26], low-code development platforms have moved MDE from research to practice, providing methods for supporting additional stakeholder groups [6, 25].

Even though MDE seems to be a promising approach for providing more accessible applications, how is it used to improve applications for visually impaired users? Even though standards and guidelines for developers have existed for many years, e.g., WCAG [98], User Agent Accessibility Guidelines [96], or *EN 301 549* [33], their application in practice seems to be still limited. Widely adopted GUI frameworks provide suggestions on how to tackle it, but there exists no automated support to cover accessibility. Existing techniques lack enough support for agile software development including iterations to improve end user reported issues like accessibility [90]. In this paper, we report the results of a Systematic Literature Review (SLR) to identify the current state-of-the-art of addressing accessibility needs with MDE *based on the concrete example of vision impairments.* The main contributions of this work include:

- a detailed examination of 30 highly relevant studies shedding light on the publication venue, timing, output, and nature of reported approaches,
- an analysis of which visual impairments are addressed with model-driven approaches;
- analysis of how MDE approaches support the design and implementation of systems that address visual-impairment needs;

- analysis of how the developed approaches have been evaluated;
- summarization of the approaches' reported strengths, limitations, gaps and challenges;
- the main limitations of the reported studies; and
- a research roadmap covering different aspects that we recommend be investigated in the future.

The article is structured as follows: Section 2 provides the relevant background and related literature reviews. Section 3 describes our research method. Section 4 presents the main results of the analysis based on the posed research questions and highlights the main findings. Section 5 discusses identified limitations, states what an effective MDE contribution could entail, and presents a research roadmap. Section 6 discusses the main threats to validity, and Section 7 concludes.

2 BACKGROUND AND RELATED WORK

We introduce the concepts of software accessibility and model-driven engineering, set the scope of this study to visual impairments, and discuss related literature reviews.

2.1 Software Accessibility and Visual Impairments

Software accessibility refers to the design of software systems that ensures barrier-free use by all, including people with disabilities. The concept of accessibility has been explored in a software engineering context as early as the mid 1990s (See Laux et al. [61] for an example). These efforts have evolved into several technical standards including the *WCAG* [98], *User Agent Accessibility Guidelines (UAAG)* [96], *Authoring Tool Accessibility Guidelines (ATAG)* [95], or *EN 301 549* [33] to name a few. Furthermore, software accessibility has gained significance in the IT industry over the last decade, particularly as it has been legislated in several jurisdictions as a mandatory requirement for public sector Web sites and mobile applications such as the US (21st Century Communications and Video Accessibility Act (CVAA) [86]), EU (Directive 2016/2102 [35]), and Australia (Web Accessibility National Transition Strategy [8]).

Although accessibility encompasses "[...] a wide range of disabilities, including visual, auditory, physical, speech, cognitive, language, learning, and neurological disabilities." [98], this research focused on *visual impairments* only. We have chosen this narrow scope because we are not interested in studying the coverage (completeness of implementation) of accessibility standards such as WCAG, but rather aim to drill down on a) the maturity of available tools and methods for the implementation of a concrete real-world accessibility need, as well as b) the benefits to the end users. Moreover, we decided to focus on visual impairments because they do not only affect people with disabilities but are experienced by a larger segment of the general population such as the elderly.

Common visual impairments as reported by the Australian Institute of Health and Welfare (AIHW) [9] are:

- astigmatism (imperfection in the curvature of the eye)
- blindness (complete and partial)
- cataract (cloudy area in the lens)
- color blindness
- hyperopia (long-sightedness)
- macular degeneration (eye disease that affects central vision)
- myopia (short-sightedness)
- presbyopia (loss of focusing ability with age).

All listed impairments except color-blindness and (complete and partial) blindness are typically referred to in the SE literature under the umbrella term of visual, or vision, impairments. The accessibility needs related to visual impairments are complex. Firstly, the above list is by no means exhaustive. Secondly, the severity of visual impairments differs from person to person and, thirdly, often changes with time or situation. It is therefore not trivial to design software systems that address visual impairments. Standards like WCAG provide guidance, however considerable effort on the part of the developers is still required to implement accessible software systems. A set of easy-to-use tools and techniques may simplify the situation and remove barriers for developers to implement accessible software systems. Model-driven engineering approaches have the potential to do exactly this.

2.2 Model-driven Engineering

MDE is a software development methodology that enables the generation of software systems based on the creation and transformation of models. Information about a given problem (domain) is typically provided with a visual or textual *domain model*, "which describes the various entities, their attributes, roles, and relationships, plus the constraints and interactions that describe and grant the integrity of the model elements [...]" [19]. Hence, MDE is centered on modeling the concepts of the respective problem domain, instead of the details of the programmatic implementation of a solution.

The other core MDE concept — *transformations* — are well-defined mappings between source and target models or code. Transformations enable the step-wise refinement of an abstract model (the domain model) to a concrete executable model, or in other words, source code. Executable models typically come in one of two forms: a) code generation with an existing programming language (also known as model-to-text transformation), or b) virtual machine-based interpretation of a low-level representation of a model. [1]

MDE claims to offer several key advantages [93] including

- *Openness.* As domain models represent concepts of a problem domain, even domain experts with limited to no programming skills can use MDE tools to create software systems.
- *Separation of concerns.* Different models can be used to capture different aspects of a problem domain separately to break down complexity.
- *Automation.* Once models and transformations are defined, software can be generated with limited to no need for manual intervention.

The topic MDE and accessibility for *visual impairments* is less researched, as existing work focuses on accessibility in general, e.g., MDE and accessibility for seniors [102], accessible web pages [75], automated test kits for accessibility in Android apps [41], the generation of accessible UIs [66], or present a roadmap for MDE and accessibility in general [16].

2.3 Related Literature Reviews

The concept of accessibility has attracted interest in SE since the mid 1990s. The roots of MDE can be traced back even further to the emergence of computer aided software engineering (CASE) tools in the 1980s. Consequently, a plethora of primary studies on either topic has been published over the years. The objective of this research is to identify those studies that address a combination of both, or more specifically, the design and implementation of software systems that address the needs of people with visual impairments using MDE methods. This article reports on the systematic and rigorous identification and assessment of relevant primary studies.

[1] A detailed discussion of MDE principles is beyond the scope of this study. Please refer to Brambilla et al. [19] for a comprehensive introduction.

As a first step, we identified the need for our study by reviewing existing secondary studies that already address our area of interest. At the time of writing, only one highly relevant study was available. Ordoñez et al. [74] offer a well-presented review of 37 primary studies based on the following research questions:

(1) ”What are the motivations for using model-driven development for producing accessible software?
(2) Which method is proposed for the model-driven development of accessible software?
(3) Which tools and languages provide technological support for the integration of accessibility in the context of model-driven development?
(4) What are the advantages offered by the model-driven development of accessible software?
(5) What are the disadvantages in the context of model-driven development of accessible software?” [74]

While Ordoñez et al. focus on the intersection of MDE and accessibility, they present a generic mapping of reported approaches, rather than a detailed analysis of the methods and tools used to model and generate accessible software systems. Despite being a valuable contribution, it is hard to assess the applicability of the reviewed approaches to address concrete real-world accessibility needs. In contrast our SLR will go more into detail and provide:

- An in-depth evaluation of the capabilities of approaches to capture accessibility (visual impairment) needs,
- A detailed analysis of the evaluations of reported approaches and benefits to end users, and
- A detailed assessment of the maturity and reproducibility of reported approaches.

In relation to our SLR but without the same focus, we have identified seven partially relevant secondary studies. Rokis and Kirikova [79] as well as Bucaioni et al. [20] review challenges of *Low Code* software development that includes some MDE approaches but they do not consider accessibility requirements. The following studies are relevant regarding accessibility and low vision but have no focus on MDE: Di Gregorio et al. [29] review practical aspects of the creation of accessible mobile apps. Paiva et al. [76] provide a review of the coverage of accessibility needs in software engineering processes. Teixeira et al. [84] analyze how accessibility requirements are considered in the development process of information systems. De Oliveira and Filgueiras [27] review development tools for creating mobile applications accessible to visually impaired users. De Souza et al. [28] analyze the use of intelligent environments for visually impaired people.

3 RESEARCH METHOD

Our SLR followed the widely accepted guidelines for the preparation of systematic SE literature reviews by Kitchenham et al. [56], as well as the Wohlin [103] guidelines for a systematic snowballing strategy. An overview of the design of our SLR process is depicted in Figure 1. The remainder of this section details the key elements of this process.

3.1 Research Questions

The key research questions (RQ) addressed by this study are:

RQ1. Which trends exist in terms of publication date, venue, impact and nature of reported approaches? – This RQ explored the visibility and impact of the selected studies in the MDE community. For this reason, we analyzed publication frequency, types and venues, looked into author affiliations and citation counts, and assessed the main contribution type of each of the selected studies.

RQ2. What visual impairments are addressed with model-driven approaches? – With this RQ, we aim to understand the depth and breadth with which human aspects of visual impairment needs were addressed in the selected studies, in the context of varying user needs and their complexities for different visual impairments.

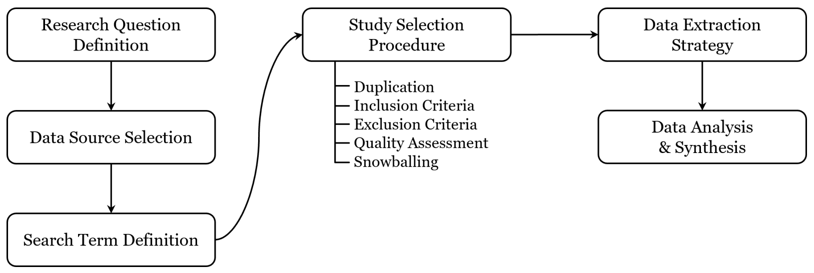


Fig. 1. Overview of our research process design and its key components, namely *Research Questions* (cf. Section 3.1), *Data Sources* (cf.Section 3.2), *Search Term* (cf. Section 3.3), *Study Selection* (cf. Section 3.4), *Data Extraction* (cf. Section 3.6) and *Data Analysis* (cf. Section 3.7)

RQ3. How do model-driven approaches support the design and implementation of systems that address visual-impairment needs? – We examined the technical details of the selected studies by analyzing the reported approaches in depth and we present our findings using the stages of a software development life cycle.

RQ4. How are these approaches evaluated? – This RQ reviewed the evaluation methods documented in the selected studies, with a specific focus on user acceptance testing with persons with visual impairments, or in other words, how well visual impairment needs were addressed by the selected studies.

RQ5. What strengths, limitations, gaps and challenges are reported? – With this RQ, we identified the limitations of the selected studies as well as open challenges and future work in the domain of MDE for the design and implementation of visual impairment needs.

3.2 Data Sources

We decided to search for literature in well-established scientific online databases. We chose to use:

- ACM Digital Library, https://dl.acm.org/
- IEEE Xplore, https://ieeexplore.ieee.org/
- ScienceDirect, https://www.sciencedirect.com/
- Scopus, https://www.scopus.com/.

These selected databases provide access to rigorously peer-reviewed publications — an important quality aspect reflected in our exclusion criteria. ACM and IEEE were included because they are primary venues for the dissemination of SE / MDE literature. Scopus and ScienceDirect were added to widen the search to relevant studies in other disciplines. Note, our original design included Springer (https://link.springer.com/) as well. However, a large number of irrelevant hits and duplicates combined with the export limit of 1000 records made the use of Springer ineffective. Instead, we opted for a backward and forward snowballing procedure to ensure the detection of further studies that did not produce hits in the selected databases (cf. Sec. 3.4.5).

3.3 Search Term

The search was performed based on below search string that contained several generic elements to ensure both aspects of our study — visual impairments and MDE— were present in all candidate studies:

(accessibility OR blind* OR "low vision" OR "vision impair*" OR "visual* impair*")
AND (MDE OR "model-driven")

Note that the omission of the conditions listed in Sec 2.1 was due to the fact that they are typically referred to in the SE literature under the umbrella term of 'visual impairment' or related synonyms. The keyword 'accessibility' was added to ensure the detection of candidate studies that do not mention visual impairments in the searched parts.

Logical operators ensured that the two different aspects of our study (accessibility and MDE) were both present (*AND*), and at least one alternative spelling of a keyword or synonym was reflected (*OR*) in all search hits. Parentheses *()* allowed logical groupings, apostrophes *""* marked atomic terms, while wildcards * were used to account for alternative spellings.

The search query was tested and refined with a series of pilot runs. The result of each run and subsequent refinements were discussed in several team meetings with all authors present. The final database-specific search parameters are presented in Table 15 in Appendix B. The search was carried out by one author and candidate studies were documented in a spreadsheet.

3.4 Study Selection Procedure

The selection procedure encompassed multiple screenings to remove duplicates and to apply inclusion and exclusion criteria. Initial screenings (duplication removal, exclusion criteria) were performed by one author who documented all rejected studies including the reason for their rejection in a spreadsheet. Five authors contributed in screening the inclusion criteria. The candidate studies were evenly distributed among them. Five percent of rejected studies (10 studies) were randomly assigned to the other authors for cross-checking if the criteria were applied correctly. There were no mistakes found.

3.4.1 Deduplication. The removal of all duplicates of candidate studies after the online search.

3.4.2 Inclusion Criteria. Note, our SLR did not place restrictions on the publication date to ensure all relevant previous and current contributions were detected. Likewise, we kept our study open to all scientific venues, including journals, conferences and workshops. Candidate studies were included if they satisfied all of the following conditions:

- Study reported a model-driven or related model-based approach
- Study focused on accessibility needs
- Study presented a model-driven method to address a visual impairment need.

Model-based approaches were considered *related* if they cover or contribute to model-driven techniques or methods such as the modeling of accessibility needs (e.g., with a domain-specific language and/or visual notation).

The inclusion criteria were applied by checking study titles, abstracts and author keywords for (a) direct keyword matches, and (b) semantic relevance. Semantic relevance was defined to exist if a study related to the key aspects of this SLR — visual impairments/accessibility and MDE. If no match was detected, the study was dropped.

3.4.3 Exclusion Criteria. The following types of studies were excluded

- Non-primary studies
- Non-English language studies
- Non-peer reviewed or non-verifiably peer-reviewed studies
- Studies superseded by newer versions
- Short papers of 4 or less pages incl. references.

Note that short papers were excluded because they do not provide sufficient detail due to space limitations.

3.4.4 Quality Assessment. The quality of each of the studies remaining after deduplication, and applying inclusion and exclusion criteria was evaluated by five authors based on below questions and the scoring system presented in Table 1. The scoring system was intentionally kept simple and was applied consistently across all Quality Assessments (QAs). Each QA used a three-level scale - Yes, Partial, or No - with criteria informed by patterns observed during the analysis of candidate studies. A score of Yes indicated that the study fully met the criterion with clear and specific evidence; Partial reflected cases where the criterion was addressed only in a generic or incomplete manner; and No was assigned when the study did not address the criterion at all. The results of our quality assessment are presented in appendix B in Table 17. Although none of the studies reached the maximum score (indicating very high quality), we decided not to remove studies based on their score due to the small number of identified candidate studies and to reduce publication bias. The QAs are:

QA1 Was the study clearly motivated by a real-world visual impairment need?
QA2 Was the presented approach clearly defined?
QA3 Was sufficient detail provided to enable reproduction and replication?
QA4 Was an accessibility evaluation with visually impaired users presented?

Table 1. Quality Assessment (QA) scoring system

Criterion	**Score**		
	Y (yes) – 1.0	**P (partial) – 0.5**	**N (no) – 0.0**
QA1	concrete visual impairment needs stated	generic motivation, visual impairments mentioned	motivation without visual impairments
QA2	solution concept clearly and formally defined	solution concept clearly but informally defined	solution concept is unclear or no concept given
QA3	detail sufficient and clear, reproduction possible	detail somewhat sufficient and clear, reproduction partially possible with assumptions	detail either insufficient or unclear, reproduction not possible
QA4	acceptance testing with visually impaired users reported	acceptance testing with partially / fully sighted users reported	weaker forms of evaluation or no evaluation presented

The quality assessment of the candidate studies was distributed evenly among five authors. We met several times to discuss individual findings, identify discrepancies, and cross-check our reviews. For the cross-checks, each of the five data-extracting authors was randomly assigned three studies that were originally reviewed by another team member. The results of these second reviews were discussed and agreed in two meetings.

3.4.5 Snowballing Procedure. All selected studies underwent forward and backward snowballing as per Wohlin [103]:

- **backward snowballing** "means using the reference list [of a candidate study] to identify new study to include"
- **forward snowballing** "refers to identifying new studies based on those studies citing the study being examined".

Forward snowballing was carried out with Google Scholar [45] which provides access to forward references of a study via the *'Cited by'* link. Google Scholar was chosen because it surpasses our selected databases in providing the most up-to-date list of 'Cited by' references based on information retrieved from a wide range of online databases and other online sources.

Backwards snowballing was carried out by collecting all references cited by our selected studies. After removing duplicates, we further analyzed them together with the candidate studies from the forward snowballing.

The set of snowballed candidate studies was divided up evenly among five authors. Identified studies were discussed and agreed upon in meetings with the five data-extracting authors. A preliminary manual selection of candidate studies, based on their titles, abstracts, and keywords and similar to the original database search, was conducted to reduce the number of snowballed studies requiring full text evaluation. Snowballed studies were treated like all other candidate studies and underwent the same selection procedure (deduplication, applying exclusion and inclusion criteria and the quality assessment). Snowballing was continued until no further candidates were found.

3.5 Search Results and Study Selection

Our initial literature search was performed in February 2023 and repeated in January 2024 and we performed an additional forward snowballing in October 2025. The search results per database and the total number of 207 identified candidate studies are shown in Table 2. The number of candidates with Scopus was dis-proportionally higher than with other databases. The likely reason for this is that Scopus provides access to research literature from a diverse range of disciplines where some of our search keywords may have a different meaning. However, the extra number of candidates did not pose a problem for our SLR, except for requiring additional time during the selection process. An additional 240 candidates were identified during two iterations of snowballing (iteration I: 136, iteration II: 104). Table 2 provides an overview of our study selection results. Columns 2–4 show the number of removed candidate studies for each step of the selection process, while column 5 shows the number of selected studies. After completing study selection, a total of 30 primary studies were selected for our SLR (cf. Table 14 in Appendix A).

Table 2. Overview of search / snowballing candidates and study selection results

Total number of candidates		**Number of removed candidate studies**			**Number of selected studies**
		Deduplication	**Inclusion Criteria**	**Exclusion Criteria**	
Database search					
ACM DL	38				
IEEE Xplore	32				
ScienceDirect	15				
Scopus	122				
	207	56	102	30	**19**
Snowballing, iteration I	136	61	41	23	**11**
Snowballing, iteration II	104	82	15	7	**0**

Total number of selected studies: 30

3.6 Data Extraction Process

A comprehensive set of data points was defined for extracting the necessary information to answer our RQs (cf. Sec. 3.1). These data points were grouped into the five categories: *Metadata*, *Accessibility*, *Approach and results*, *Evaluation* and, *Limitations and open challenges* as shown in Table 16 in Appendix B. Data extraction was conducted by five authors and collected in a shared spreadsheet. The extraction process started with three joint sessions to collectively establish a shared understanding and vocabulary, share data extraction best practices, and resolve open questions and anomalies. Joint sessions lasted 60 minutes each and were attended by the five authors involved in data extraction. During these sessions, each author was also randomly assigned three studies to cross-check for inconsistencies in data interpretation and extraction, as well as to detect potential mistakes. The cross-check results were then discussed collectively in the last of the three session. Although this procedure did not cover all candidate studies, it improved the overall quality of the data-extraction process by enabling authors to revisit and correct all their extractions when issues were identified.

3.7 Data Analysis and Synthesis

The extracted data (see all data points in Table 16, Appendix B). were analyzed and synthesized using quantitative and qualitative methods. For example, M1-M6 in the metadata category such as place and year of publication, DP 1.1-1.3 about visual impairments, target audience and used standards, or DP 2.5 about types of software systems were quantitatively summed up. In contrast, qualitative analyzes were conducted on the data collected using data point DP 2.1 for understanding how the MDE approaches work in detail, DP 2.2 on understanding if DSLs were used, DP 2.3 on modeled application aspects, DP 2.4 on the level of MDE process automation, DP 3.1 on the used evaluation method and details about it, DP 3.2 on the achieved level of accessibility, DP 4.1 about pros, cons, limitations and open challenges in the primary studies, DP 4.2 on addressed research questions, and DP 4.3 on open gaps and challenges from accessibility and MDE perspectives. For DP 2.1, we added a substructure for analyzing different software development phases (see structure in Table 16). This way it was easier to extract data about the published MDE processes in a structured way. These qualitative analyses were in a first step split up among the 5 authors performing the data extraction, each study had a first and second coder, and the results were then discussed in the whole group of authors. All selected studies were first analyzed individually to identify their key features. A horizontal analysis across all selected studies was then carried out to identify patterns and trends, and to group and classify individual studies. Data analysis and synthesis of each data point were carried out in groups of at least two authors. Results and findings were discussed and agreed on in more than 40 team meetings.

4 RESULTS AND OBSERVATIONS

We report on analysis results and observations following our five guiding research questions. A deeper discussion, interpretation, and summary of limitations of the results can be found in Section 5. Some complementary information can be found in our web repository [68].

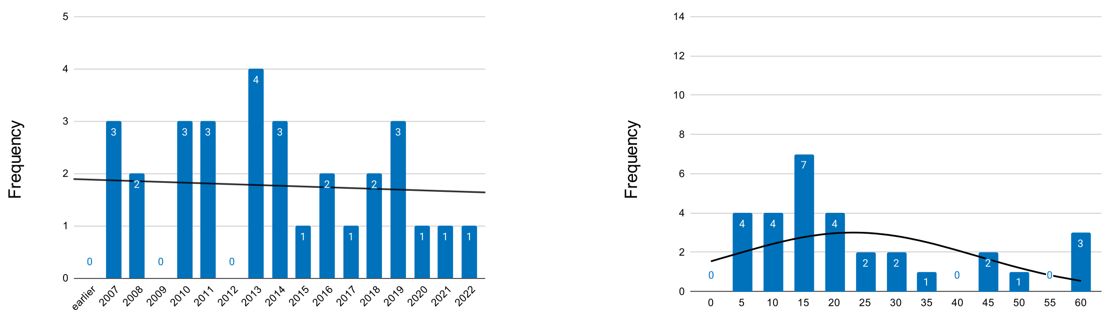
4.1 RQ1 – What trends exist in terms of timing, output, impact and nature of reported approaches?

To explore the visibility and impact of the selected studies in the MDE community, we analyzed publication frequency, types and venues, looked into author affiliations and citation counts, and assessed the problem and contribution types of each of the primary studies (based on data extraction points M1-M6 in Table 16, Appendix B).

4.1.1 Timing and output. As shown in Figure 2a, our 30 selected studies started appearing from 2007 onward with an overall low frequency that is slowly trending downward after a more active period between 2007 to 2014. On this basis, three observations can be made:

(1) *Delayed interest.* The MDE community started covering the topic of visual impairments significantly later than the SE domain in general (2007 vs mid 1990s) (cf. Section 2.1).
(2) *Low output.* MDE approaches that address visual impairments are a niche topic with a peak of only 4 publications in 2013.
(3) *No clear trend.* An increasing accumulation of knowledge (publications) over the years cannot be observed. On the contrary, the momentum has dropped off since 2019.

Research into MDE approaches that address visual impairment needs started late, trends downward, and has yielded a low output to date.



(a) Number of publications per year

(b) Number of citations (histogram + normal distribution)

Fig. 2. Comparison of publications and citations over the years

4.1.2 Impact in the MDE community. While publication venues show a large spread (cf. Table 18 in Appendix C), surprisingly none of the 30 selected studies was published at a high-ranked MDE venue such as MODELS, ASE, SLE, GPCE or the SoSyM journal. Instead a trend towards Human-Computer Interaction (HCI)-related journals and conferences can be observed, e.g., the Journal on Universal Access in the Information Society (3 studies), ACM International Conference on Design of Communication (3 studies), Conference on Innovations in E-learning, Instruction Technology, Assessment and Engineering Education (2 studies), and the International Conference on Universal Access in Human-Computer Interactions (2 studies).

Our 30 selected studies were created by a small community of authors who frequently collaborate with each other. Table 19 in Appendix C maps all 23 author teams against their published studies and depicts which of the studies were co-authored by multiple teams. The table also shows that only nine author teams have published more than one study on the topic of this SLR – often in collaboration with other teams. The dissemination of information beyond this community of authors is to date limited, as the low citation counts indicate. We used Google Scholar to determine the number of citations per selected study. Figure 2b depicts a histogram of the citation frequency. As can be seen, the largest portion of studies (7 studies) was cited between 11-15 times. Only a small number of studies (7) were cited more than 30 times as of 9 January 2026. This may be due to the specific focus of our selected studies, but can also be interpreted as a trend towards low impact in the MDE community.

The impact in the MDE community to date is low. None of the selected studies appeared in a high ranked MDE venue. Generally. low citation counts indicate that most of the studies did not create interest beyond a small community of collaborating authors.

4.1.3 Problem Approach, Contribution and Study Type. With respect to the nature of the 30 selected studies, we analyzed information about their problem approach, the type of contribution and study type. Regarding the overall problem approach, 29 out of 30 selected studies describe research under lab conditions, which typically entails a lack of motivation with a concrete real-world need, and/or a lack of evaluation with vision impaired end users. This reflects the often abstract and conceptual nature of the majority of the selected studies. Only study S6 presents and evaluates work under realistic real-world conditions.

A majority of 20 studies present a new method or technique, that is in just under 50 per cent of cases accompanied by a prototype demonstration or a tool description (cf. Table 3). In contrast, only 5 out of 30 studies provide an evaluation with end users – vision impaired or not. Hence a focus on technical aspects can be noticed.

Table 3. Overview of contribution types

Contribution type	Number of studies
Method / technique	20
Tool	3
Prototype	9
Evaluation	5
Replication study	0

Table 4. Overview of Study types

Study type	Number of studies
Journal papers	12
Regular conference paper	8
Short conference paper	8
Extended conference paper	2

The focus on technical aspects must be seen, however, in light of a high number of short papers (cf. Table 4), which due to their page limitations do not provide sufficient space for reporting necessary technical detail. Consequently, the short as well as some of the regular papers may not meet the requirements of the MDE community. The lack of technical detail is also problematic because it restricts replication efforts. It is therefore no surprise that not a single replication study was found (cf. Table 3).

MDE approaches that address visual impairments are often abstract and conceptual in nature, do not balance technical and human aspects well, while also lacking technical detail required for assessment and replication.

RQ1 Summary: Research into MDE approaches for visual impairment began late, shows a downward trend, and has produced limited output with low impact in the MDE community. Many of these studies are conceptual, lack technical depth, and are rarely published in high-ranking venues. This highlights a need for more balanced, technically detailed, and widely disseminated research in the field.

4.2 RQ2 – Addressed visual impairments

We investigate what visual impairments are addressed with model-driven approaches. This includes investigating both ongoing challenges from the accessibility perspective and the MDE perspective.

4.2.1 Addressed Visual Impairments (see DP 1.1 in Table 16, Appendix B). As pointed out in Section 2.1, *visual impairments* is a collective term for several conditions with a varying degree of vision loss, and thus different resulting requirements and features in a software. So we wanted to know which visual impairments were addressed and in what context.

Studies addressing accessibility in general. 24 out of 30 primary studies address accessibility in general rather than specifically targeting visual impairments (see Table 5). None of these 24 studies provides a clear definition of the visual impairments covered. Most of them rely on WCAG or similar standards and guidelines and thus remain at a technical level, opting not to explore socio-technical aspects as a whole. Accordingly, visual impairments are primarily used as examples and not as subjects for detailed study. 17 studies reference visual impairments as part of their own approach most often to provide a representative example of a technical measure. The most frequently used terms used to refer to visual impairments in these 17 studies are still mostly generic, incl. *visual impairments* (14 times), *blindness* (12 times), *low vision* (7 times), *color blindness* (4 times) and *cataract* (1 time). 7 studies do not reference visual impairments as part of their own contributions: 3 studies discuss visual impairments only in a case study or other form of evaluation, whereas 4 studies only provide examples of visual impairments in the introduction or related work sections.

Table 5. Addressing visual impairments in our primary studies (DP 1.1)

Number of studies	Covering visual impairments	Study keys
6	approach specifically for visually impaired users	S1-S3, S6, S10, S17
17	generic approach but refers to visual impairments in the approach sections	S4, S5, S8, S11-S13, S18-S22, S24, S25, S27-S30
3	generic approach but use a case study/evaluation with visual impairments outside of their own approach sections	S9, S7, S1
4	generic approach and uses only examples outside of their own approach and case study/evaluation section	S14, S15, S23, S26

Studies addressing visual impairments. On the other hand, 6 out of 30 primary studies specifically address visual impairments as expressed through their titles, abstracts and presented novel contributions (see Table 5). Although all 6 studies directly address visual impairments, only 4 studies (S1-S3, S6) also evaluate visual impairment-related aspects of their approaches via a case study or any other form of evaluation. The most frequently used key words in these 5 studies are: 6x *blindness (partial/total) or legally blind*; 5x *visual impairments or vision disabilities*; 5x *reduced/low vision or vision loss*, 1x *color blindness*, 1x *no light reception*, 1x *blurred vision*. The abstract nature of these keywords indicates that - just like the other 24 studies - these 6 primary studies do not consider the true nature and complexities of visual impairments in detail. Finally only one study (S17) provides basic definitions for common visual impairments.

The large majority of studies provide generic approaches improving accessibility without studying human-centric aspects of visual impairments in detail. We argue in favor of considering accessibility and in the concrete context of this SLR, visual impairments, as first-class citizens and consider them beyond the technical domain when addressing it.

4.2.2 Target audience. Another aspect we explore is the target audience of the selected primary studies (see DP 1.2 in Table 16, Appendix B). The obvious target audience are, of course, software engineers as the presented approaches were designed to support the integration of accessibility / visual impairment-related needs into MDE approaches and tools. 22 out of 30 studies name *software engineers* or *developers* directly (S1-S5, S7-S11, S15-S18, S20, S22-S28), while some of these studies also consider other functional roles such as (software) *architect*, *analyst*, *modeler*, *designer*, *service designer*, and *accessibility specialist*. In contrast, 4 primary studies refer exclusively to *designers* and *service designers* (S13, S14, S19, S21), and 4 do not explicitly name their target group (S6, S12, S29, S30). The differentiated view towards involved functional roles can be interpreted to mean that certain specifics need to be taken into consideration to ensure the activities, skills and needs of people acting in these roles are met. Although three studies (S5, S7, S8) present evaluations of their MDE tools by software developers, no study explicitly sets out such special considerations. In contrast, studies that do not explicitly state their intended target audience may reflect a technology-centric perspective.

Regarding the evaluation, there is another obvious target audience of the studies included in this SLR. The quality of the artifacts created with an MDE tool - in our case software features that address the accessibility needs of users with visual impairments - can only be effectively validated by end users suffering from these conditions. However, only 6 out of 30 studies present an evaluation with end users (see Section 4.4 for more details about the evaluation). The numbers pick up when looking at studies that explicitly name an end user target group. 17 studies refer to users with visual impairments, while 9 more studies refer more generally to people with disabilities. The latter can be explained by the fact that our search resulted in the inclusion of studies that do not explicitly target the accessibility needs of users with visual impairments but still address visual impairment-related approaches, or present relevant examples.

Most primary studies consider different functional roles to represent anticipated users of their MDE-based approach, and identify beneficiaries of improved accessibility. However, they often overlook or briefly address the specific needs of these audiences, neglecting the human and social aspects of socio-technical systems. This shortcoming may hinder understanding of how effectively the tools address accessibility needs or whether they tackle real-world accessibility issues.

4.2.3 Accessibility Guidelines. The accessibility guidelines used in our examined studies (DP 1.3 in Table 16, Appendix B), and the number of studies using them are shown in Table 6. Most of the studies, namely 21, used WCAG. The second common accessibility guideline used in six of the examined studies is WAI-ARIA [97], the Accessible Rich Internet Applications Suite. ISO 9241-171:2008 [37] is the third popular set of guidelines, used by three of the examined studies. Accessibility development documentation for Android [63] and iOS [50] applications is used in two of the studies. There exist 8 further standards, that are used in one primary study each (see Table 6). Finally, 5 of the studies did not use any guidelines.

Table 6. Accessibility guidelines used in our studies (DP 1.3)

Number of studies	Accessibility guidelines	Study keys
20	Web Content Accessibility Guidelines (WCAG) (including WCAG 2.0 [94], WCAG 2.1 [99], WCAG 2.2 [98], and WCAG mobile draft [100])	S2-S5, S7-S11, S13-S15, S18-S20, S21, S23-S25, S27, S28
6	WAI / WAI-ARIA: Accessible Rich Internet Applications Suite (developed by W3C) [97]	S6, S9, S13, S26, S27, S29
3	ISO 9241-171: Ergonomics of human-system interaction — Part 171: Guidance on software accessibility (developed by the International Organization for Standardization (ISO)) [37]	S4, S15, S16
2	Accessibility development documentation for Android [63] and iOS [50]	S2, S9
1	Authoring Tool Accessibility Guidelines (ATAG) web standard (developed by W3C) [95]	S23
1	BBC mobile accessibility guidelines [11] (for bbc.co.uk content, developed for UK audiences, for use with technology commonly available in the UK)	S2
1	IMS guidelines for developing accessible learning applications [47] (developed by 1EdTech)	S28
1	ISO/IEC 12207 [39]	S8
1	Material Design guideline for accessibility [63] (developed by Google)	S7
1	SIDI guidelines for accessible mobile applications (developed by SIDI)	S2
1	User Agent Accessibility Guidelines (UAAG) (developed by W3C) [96]	S16
1	Universal design guidelines (developed by U.S. General Services Administration (GSA))[5]	S6
5	not specified	S1, S12, S17, S22, S30

Other guidelines that are mentioned in several studies but have not been used in any of them are: ISO 9126 [36], A11Y [24], ISO/IEC 40500:2012 [38], BITV, UNE 139803, IEEE Std.610.12, European Commission, and national government guidelines, General Accessibility Framework for Administrations (RGAA) [73], Accessibility for Ontarians with Disabilities Act (AODA) [62], Section 508 technical standards [4], initiatives in different countries related to Web accessibility, and UNE 139803.

WCAG guidelines are the most used and helpful for developers new to accessibility, but must be tailored to specific project requirements. As they do not focus on particular impairments, end-user design validations and product evaluations are essential to identify missing aspects.

RQ2 Summary: The analysis shows that only a small subset of the reviewed studies (6 out of 30) explicitly addresses specific visual impairments as a primary concern. Instead, the majority of approaches focus on accessibility in a general sense, often without distinguishing between different types of visual impairments (e.g., low vision, color vision deficiency, or blindness). In most cases, visual impairments are not modeled as explicit requirements but are implicitly subsumed under the phrase accessibility. Most studies consider functional roles and beneficiaries but provide limited discussion on specific user needs. WCAG guidelines are commonly used but require project-specific adjustments, end-user validations, and product evaluations. As a result, the reviewed studies provide limited insight into how particular visual impairments are systematically considered or supported through MDE.

4.3 RQ3 – Model-Driven Engineering Approaches Used

We analyzed how model-driven approaches for the design and implementation of the needs of visually impaired people have been used in our selected primary studies (RQ3). This includes a consideration of which types of applications are developed in the studies, which frameworks and languages the authors use to realize their approaches, and what target application aspects were modeled.

To better compare how the different approaches work (DP 2.1 in Table 16, Appendix B), we have extracted which process steps occur in the primary studies. The chosen process steps are based on typical MDE software engineering processes (see Figure 3), namely requirements analysis in the *analysis phase*, modeling (different types of models such as Platform Independent Models (PIMs), or Platform Specific Models (PSMs)) and model transformation in the *design phase*, code generation (Gen Source Code) and manual coding (Hand-written (HW) Source Code) that fit to a Runtime Environment (RTE) in the *implementation phase*, and testing the resulting application in the *test phase*. In addition, one could derive models via reverse engineering from code. There might be variations in MDE approaches, e.g., more modern approaches often do not explicitly differentiate between PIMs, or PSMs as platform specific information is often covered in the RTE, or approaches try to reach a higher degree of automation by automatically transforming natural language requirements to intermediate formats (i.e., models) and then code. Despite these slight variations, it is important that these main steps and their artifacts are in depth explained to understand an MDE approach. In addition, we have analyzed which of the approaches in the primary studies allow for runtime adaption of the generated system during the *operation phase*. In Table 7, we show the results of the qualitative analysis of each primary study regarding the steps in Figure 3. In the following subsections, we explain each of the phases and the main findings in detail.

4.3.1 Analysis Phase. We have investigated which approaches have included accessibility requirements within the analysis phase and how they have included them. 16 out of 30 studies included requirement analysis elements in their approaches. To give some examples, S3 has studied and selected 28 accessibility requirements that explicitly target mobile devices. They suggest actionable recommendations when developing native mobile apps and integrate them in a cross-platform framework for MDE of mobile business apps. S2 lists 25 common accessibility requirements in mobile applications which the authors compiled from other publications and WCAG 2.1, and describes how they help to avoid certain challenges. They integrate these requirements similar to the work in S3 in extending their DSL, in the IDE, as well as in changed transformation rules. S13 lists 5 native requirements for media players and 11 additional requirements to satisfy specific accessibility needs of users. They stay on a very abstract level, e.g., “Color”, or “Find”. They used these requirements when modeling the user interface. S18 collects reusable components such as accessibility

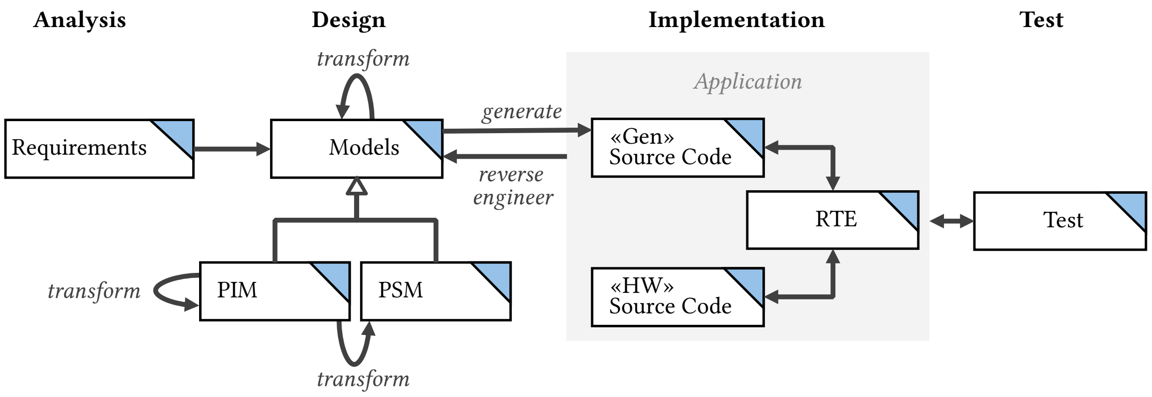


Fig. 3. Overview of a generic MDE approach. Steps in which, according to the studies considered, accessibility requirements had to be particularly taken into account by a developer are marked with a triangle in the top right-hand corner.

guidelines, design patterns, and norms in an accessibility-supportive repository. Requirements are modeled in a graph structure connecting requirements nodes by labeled arcs. However, their connection to the other models or the designed system is not clear and the approach is not replicable.

Only slightly more than half of the studies mention the use of accessibility requirements, with few explicitly addressing specific ones. Requirements are often listed but not explicitly modeled, integrated into tools, or manually used to support UI development, limiting reusability for other researchers.

4.3.2 Design Phase. We have further investigated which modeling languages and frameworks were used in the design phase of the systems (see Table 7), categorized the studies if they are using an approach that distinguishes between platform-independent and platform-specific models or not, and analyzed what aspects were modeled. In addition, we have analyzed the model-to-model transformation approaches described in the primary studies.

Clearly, modeling could also be a task in the analysis phase (then more related to the problem domain or domain of interest). When talking about models in MDE processes, theses models often belong more to the design (the solution space) than to the analysis of a system, even though models might have been developed during analysis and are then refined in the design phase. Thus, we rather see the models described in the primary studies as part of the design phase.

Used modeling languages and frameworks. To group the **different modeling approaches**, we have investigated which modeling languages and frameworks were used by the different approaches (DP 2.2 in Table 16, Appendix B). Most of the studies, namely 25, use GPLs or DSLs (see Table 7).

The modeling notation with the most mentions is UML, namely 10 times (see Table 8). 9 studies mention that their approach is based on the Cameleon Reference Framework [22], a reference for classifying user interfaces supporting multiple contexts. The Cameleon Reference Framework suggests four levels of abstraction: task and concepts, an abstract, concrete, and final user interface. These different levels of abstraction can be well handled by MDE approaches using model-to-model transformations. Out of these 9 studies, 5 studies also use the User Interface eXtensible Markup Language (UsiXML) [87]. UsiXML is structured in four levels defined by the Cameleon Reference Framework: Task and Concepts, as well as Abstract, Concrete, and Final User Interfaces.

Table 7. Categorization of the approaches based on the phases in the development process (DP 2.1). *Analysis phase*: which primary studies consider accessibility requirements; *Design phase*: which primary studies use a GPL or DSL, and specify PIMs, PSMs and model-to-model transformations; *Implementation phase*: which primary studies talk about code generation without giving further details, generate a full running application, generate an application core, or allow for hand-written additions, and which use a reverse-engineering approach to derive models from code; *Test phase*: which studies test the generate application automatically or manually; *Operation phase*: Which studies allow for runtime adaption.

Study key	**Analysis Phase**	**Design Phase**				**Implementation Phase**					**Test Phase**		**Application Operation**
	requirements	GPL DSL	PIM	PSM	M2M trans-formation	gen. but less details	full gen.	core gen.	hand-written additions	reverse eng.	automatic	manual	runtime adaption
S1		●	●			●							
S2	●	●				●							
S3	●	●				●							●
S4			●	●	●								
S5		●	●	●	●		●						
S6							●			●			●
S7		●						●	●		●	●	
S8	●	●	●					●			●	●	
S9		●				●							
S10		●						●	●				
S11	●	●	●			●						●	
S12		●						●					
S13	●	●	●		●								
S14		●	●										
S15			●										
S16	●	●			●		●						
S17		●	●	●	●			●					
S18	●	●	●										
S19	●	●	●		●	●							
S20	●	●	●	●	●								●
S21		●											
S22	●	●	●				●					●	
S23		●	●			●							
S24	●	●	●	●	●								
S25	●		●	●									
S26	●	●	●	●									
S27		●	●	●	●	●							
S28	●	●	●	●	●			●					
S29	●	●	●	●				●					
S30	●												

6 primary studies use the Eclipse Modeling Framework [83] and 3 out of these studies also use the Graphical Modelling Framework (GMF) [31] to define their models. 5 studies reference the Meta Object Facility (MOF) structure of instances, models, and meta-models as their underlying principle. 4 studies use UIML [85], a markup language for user interfaces developed by OASIS Open that provides a canonical XML representation of any user interface. 4 studies mention the EGOKI [40] system, which generates accessible user interfaces. 2 of these studies also use the SPA4USXML tool [70], a part of the EGOKI system which provides functionalities to generate task and abstract UI models (compliant with the USIXML syntax) from a service description, to relate interaction resources, e.g., texts, images, videos, audios, to abstract interaction elements in abstract UI models, and to generate a resource model. 2 studies rely on MD^2 and Xtext. MD^2 [48] is an approach for the model-driven cross-platform development of mobile applications for Android and iOS. Xtext [105] is a framework enabling the development of programming and domain-specific languages. One of these studies also mentions using the Xtend [104] language, a Java dialect and statically typed programming language that translates to Java source code. 5 studies developed its own modeling language, mainly for describing user interfaces and 4 primary studies lack to mention the used modeling languages or frameworks at all.

While one-third of the approaches use a GPL (UML) to define models, there is no predominant modeling language, framework, or tool used by the studies. The used languages mainly support the modeling of data or user interfaces.

Table 8. Used modeling languages and frameworks (DP 2.2)

Number of studies	**Modeling languages and frameworks**	**Study keys**
10	UML	S1, S5, S8, S11, S18, S22, S23, S26, S28, S29
9	Cameleon Reference Framework	S12-S14, S16, S19, S20, S22, S26, S27
6	Eclipse Modeling Framework (EMF)	S1, S5, S13, S16, S19, S23
5	User Interface eXtensible Markup Language (UsiXML)	S13, S14, S16, S19, S27
5	Meta Object Facility (MOF)	S5, S14, S19, S23, S28
4	UIML	S12, S16, S19, S21
4	EGOKI	S12, S14, S19, S21
3	Graphical Modelling Framework (GMF)	S13, S16, S19
2	SPA4USXML	S14, S19
2	MD^2 and Xtext	S2, S3
1	Xtend	S3
1	ISATINE Framework	S4
1	WebML	S24
1	UseML	S27
1	Advanced Adaptation Logic Description Language (*AAL_DL*)	S20
5	own DSL developed	S7, S9, S10, S17, S22
4	no details given	S6, S15, S25, S30

Modeled aspects. We further investigated **what aspects** of the described applications were **modeled** (DP 2.3 in Table 16, Appendix B, for answers see Table 9). In summary, 20 studies mentioned that the UI in general or specific screens are modeled and 7 studies referred to the modeling of specific UI components or elements. 12 studies modeled the interaction between users and the UI or individual actions a user could perform, or a function, or service the user could call. 8 studies model the appearance, or look and feel of the UI, and 7 studies mention the explicit modeling of workflows, navigation, or transition between pages. 6 studies mention user groups with their specifics and capabilities, e.g., concrete sensory abilities, vision impairments, physical and cognitive disabilities (S4, S12), or user capabilities for communication modalities (S21). 5 studies mentioned the modeling of requirements, or the input and the output of services or functions. 4 studies used models to define the data structure. 3 studies described the use of presentation models and 3 studies are modeling context information such as device usage, functions of images, or contextual recommendations. When looking into the studies in detail, we found it hard to extract more information, e.g., about how accessibility was modeled or providing concrete examples for UI models, including accessibility requirements to be used in generative processes. There are, however, approaches where one could find more details on modeling the UI (S5, S6), modeling user profiles (S5), or adaptation rules (S13) that can be used as inspiration for developers.

Table 9. Which aspects were modeled in the selected studies (DP 2.3)

Number of studies	Modeled aspects	Study keys
20	UI general/ screen	S6, S7, S10, S11, S13-S20, S22, S26-S28, S30
12	interaction/ action/ function/ service	S2, S7, S9-11, S16-S18, S21, S22, S25, S30
8	appearance/ look and feel	S1-S5, S11, S18 S22,
8	UI components/ elements	S1-S5, S7, S24, S25
7	workflows/ navigation/ transition	S4, S9, S10, S23-S25, S30
6	user groups, specifics, capabilities	S4, S6, S12, S21, S24, S30
5	input/ output	S4, S9, S17, S21, S25
5	requirements	S6, S8, S24, S25, S29
4	data structure	S8, S16, S24, S25
3	presentation	S4, S24, S30
3	context information (device usage, functions of images, recommendations)	S6, S12, S30
2	adaptation rules	S12, S13
1	information architecture (data, wiki content)	S23

In our analysis, it was a challenge to identify which models are used in the generated applications and which ones are only used for representation in a publication. This was especially hard, as no code repositories were provided. Moreover, one has to distinguish between metamodels and models - a distinction that not all publications explicitly make. The analysis has shown that there is no main framework or DSL used. No standard stands out, nor can it be seen that the trend is moving toward the development of DSLs. Surprisingly, none of the DSLs modeled accessibility aspects (i.e., requirements, user profiles and capabilities, or context information such as devices and their functions) explicitly, but rather focused on the basic structure of user interfaces. Modeling the UI is the main focus of the approaches. A surprisingly low number of studies model user groups and their needs, relevant context information, or data structures explicitly. Moreover, it was not possible to extract a suggestion for an accessibility model from the descriptions in these studies as suggestions for other developers, as the presentation was in most cases too abstract, just pictures without proper modeling formalism used, and in other cases too restricted to specific use cases. Thus, the reuse of the concrete modeling approaches is limited.

Model2Model Transformation. 10 primary studies mentioned model-to-model transformation in their approaches. 4 studies (S4, S19, S20, S24) add accessibility aspects to the transformation process starting from general models describing the User Interface (UI) to models incorporating accessibility requirements. E.g., S4 adapts their UI PIMs based on information from user disability profiles and the used UI components to adapted ones, e.g., to use different output devices. This transformation is performed with a set of basic adaptation rules. Only one study (S5) already started the transformation process with accessible UI models. The authors defined an accessible UI metamodel and an interaction platform metamodel to reuse this information.

To further characterize the approaches, 15 of the primary studies do not specify if their models are platform-dependent or independent. 10 studies use both platform-independent and platform-specific modeling approaches, to create their applications. However, only 7 out of them provide further details about the model-to-model transformation between PIM and PSM. 5 studies only describe a platform-independent modeling approach. The other 5 primary studies (S13, S16, S17, S27, S28) transform abstract UI models to PSMs but lack further details.

There are only five solutions describing how to incorporate accessibility in models on a more concrete level. While they provide some ideas worth looking into it, these solutions are not reusable for other MDE projects as the detailed accessibility requirements, modeling requirements and adaptation rules are provided by giving some examples only.

4.3.3 Implementation Phase. We have analyzed how the model-driven code generation processes is performed in the 30 primary studies, which approaches allow for hand-written additions of code and which types of applications are developed (see Table 7).

Code Generation. In an MDE process, we use defined models as input for code generators to synthesize the target code of applications. We have analyzed the 30 studies to understand details of the generation process (see DP 2.4 in Table 16, Appendix B). The results are presented in Table 10: 12 out of 30 studies (S4, S6, S13-S15, S18, S20, S21, S24, S25, S26, S30) mention MDE; however, the approaches described in the studies stop at the design phase and do not include details of the generative aspects of the MDE process. 8 studies (S1, S2, S3, S9, S11, S19, S23, S27) mentioned that they are generating an application without specifying if it is fully running without handwritten additions. Accessibility is, for example, integrated directly into transformation rules (S2, S3), e.g., by adding and interpreting a focus order. 7 studies are generating an application core, a scaffold, or parts of the application (S7, S8, S10, S12, S17, S28, S29). 4 studies are generating a full running application, such as an iPhone app for educational content (S5), an android email client (S6), a web interface for a ticket shelter (S22) and an accessible media player (S16).

Table 10. Model-Driven Approaches in the primary studies (DP 2.4)

Number of studies	MDE impact	Study keys
12	Mention MDE but explanations stop before the code generation	S4, S6, S13-S15, S18, S20, S21, S25, S24, S26, S30
8	Mention that they do code generation	S1, S2, S3, S9, S11, S19, S23, S27
7	Generate Part of Application	S7, S8, S10, S12, S17, S28, S29
4	Generate Full Application	S5, S6, S16, S22

In summary, many of the studies use MDE in order to generate accessible software. However, few demonstrate their approach with the generation of fully functioning software and nearly half of the approaches stop after the design phase. No study presents details about code generators.

Hand-written Code. MDE approaches in practice have shown that it is important to support handwritten additions to generated code, as there might exist requirements or specific business logic, which can not be generated [93]. Only two studies mentioned the possibility of adding hand-written additions or additional implementation to the generated code. S7 mentions that developers add native functionality for the mobile application which runs together with the generated app project. S10 mentions that developers have to add source code to the generated app scaffold which provides required app components, libraries and features such as screen reader support or active voice input. Moreover, only one study (S6) presents a reverse engineering approach, where existing apps on a smartphone are analyzed, and simplified UIs are generated.

In summary, we can only conclude limited support for the integration of hand-written implementations, as only two studies explicitly discuss the integration of hand-written artifacts. In addition, the application of reverse-engineering methods to make already developed applications more accessible for visually impaired users is rarely considered.

Types of applications developed. Despite being often generic and abstract in nature, the vast majority of our selected primary studies explicitly state the type of software environment they focus on (DP 2.5 in Table 16, Appendix B). As shown in Table 11, most frequently (22 times) mentioned are web applications (S4, S5, S8, S11-S16, S18-S30). 9 studies (S1-S7, S9, S10) focused on mobile applications. Desktop applications were mentioned by two studies (S4, S5), and other two studies focus on industrial automation systems (S18, S22). One study (S16) is very specific and discusses accessibility and MDE in the context of Web/HTML5 media players. One study (S17) takes a fully generic approach, without committing to a specific application environment. Instead, it provides examples across several contexts, including ATMs, desktop, mobile, and web. S17 is not the only study to span multiple environments: studies S4 and S5 also cover web, mobile, and desktop systems, while studies S18 and S22 focus on web applications alongside industrial automation systems.

Table 11. Distribution of the application environments covered by our selected primary studies (DP 2.5)

Number of studies	**Software type**	**Study keys**
22	Web Application	S4, S5, S8, S11-S16, S18-S30
9	Mobile Application	S1-S7, S9, S10
2	Desktop	S4, S5
2	Industrial Automation System	S18, S22
1	Media Player	S16
1	Generic Approach	S17

The focus on web applications is a natural choice and likely linked to the WCAG web accessibility standard most frequently cited by the primary studies. This makes WCAG's accessibility guidelines directly relevant and at an abstract level straightforward to implement. The limited amount of attention to mobile applications is surprising and may spell a future work topic. Further, considering more than one application environment may be explained by the code generation possibilities offered by MDE. However, the examples in all four studies are typically not presented in a detail that allows the reader to understand the full MDE 'pipeline' from abstract model to executable code. Hence focusing on a single application environment could reduce the amount of information to present and thus increase the clarity and reproducibility of these studies.

4.3.4 Test Phase. Testing is a key aspect of software engineering and application development (cf. Figure 3 and Table 7). The inclusion of accessibility guidelines should be validated and tested. 4 studies (S7, S8, S18, S22) incorporated testing aspects into their approaches. Among them, 2 studies (S7, S8) mentioned using both manual and automated testing while the other 2 focused solely on manual user testing. (S7) uses an accessibility scanner to find accessibility problems in apps in an automated way together with manual checks of the user interface done by the authors and (S8) use an integrated tool for automatic accessibility assessment together with user acceptance tests in the evaluation.

The low number of approaches showed, that automated testing is not sufficiently considered for MDE approaches for visual impairments.

4.3.5 Application Operation. Next to developing a software that is accessible, we had a look at the operation of such an accessible system in our primary studies. Accessibility can be provided through several means and can be provided through integrated adaptability within the runtime environment. Among the reviewed studies (see Table 7), only 3 studies integrated run-time adaptation elements. S6 describes an approach to model adaptive and thus personalized user interfaces that match the needs of the user. Similar to S11 who describe a model-driven adaptive UI approach to meet the needs of the user on an individual level, S3 argues for the need to enable users to adapt an interface according to their needs and show an example; however, no information is provided if this adaptation possibility is modeled or how this is integrated into code generation or model interpretation.

Only three studies address runtime adaptation, proposing personalized UIs and user-driven interface adjustments, though integration into code generation or model interpretation is rarely detailed.

RQ3 Summary: Model-driven approaches for designing and implementing visual impairment needs show a variety of methods, though explicit modeling of accessibility requirements is not consistently applied. Many studies focus on generating accessible software, primarily targeting web applications due to the relevance of WCAG guidelines. While models often emphasize UI structure, some approaches explore user needs and context-specific aspects. The integration of hand-written artifacts is occasionally addressed, and code generation potential for various application environments is recognized, though detailed examples remain limited. Automated testing, runtime adaption, and reverse-engineering are considered in a few cases, highlighting areas with potential for further development.

4.4 RQ4 – Evaluation

We have analyzed, how the presented approaches are evaluated (see DP 3.1 in Table 16, Appendix B). Out of 30 studies, 10 did not include an evaluation, 5 studies utilized example-based evaluations (S2, S3, S11, S21, S28), 9 studies validated their approach through proof-of-concept (S1, S2, S5, S9, S14, S16-S19), and 6 conducted user studies for validation (S4, S6, S7, S8, S12, S22).

Among these primary studies, 8 studies provided specific participant numbers in their user studies and proof-of-concept validations: S18 involved 3 software developers in their proof-of-concept. S5 included 4 software developers and 5 students. S4 conducted a study with 42 participants with impairments. S7 involved 42 software developers in their study. S22 executed two studies; the first with 4 software developers and the second with a blind participant. S8 conducted two case studies with students; one with 8 students assessing methodology and application development, and another one with 14 students evaluating the MDE tool. S6 involved 41 blind users using Android smartphones in their study.

For replicating these studies, access to evaluation data, the code of the developed applications, and the code of the approach leading to these applications is essential. However, none of the studies provided this information. This also hinders to provide more details for DP 3.2 (achieved level of accessibility), as anecdotal descriptions of examples do not provide enough information about if and which of the vision impairment requirements were met.

RQ4 Summary: Evaluation methods varied among the reviewed studies, however, one third lacked evaluations. Besides example-based assessments and proof-of-concept validations, only 6 performed user studies. Participants involved developers, students, and visually impaired users, whereas only 3 studies evaluated their approaches with visually impaired users. However, none of the studies provided access to evaluation data or source code for replication.

4.5 RQ5 – Reported Strengths and Limitations

Our last research question investigates what limitations, gaps, and challenges were reported within the 30 studies.

4.5.1 Reported limitations in the state-of-practice. We analyzed what limitations in the state-of-practice motivated the authors of our selected primary studies (DP 4.1 in Table 16, Appendix B), what research questions they addressed (DP 4.2), and what gaps and open challenges were identified in the MDE domain (DP 4.3). To answer RQ5, we have analyzed the abstracts and introductions of each study. If clarity was not fully established in this way, the remaining sections of a study were consulted as well.

Table 12 depicts clusters of the main motivational aspects of the reviewed studies. Lack of awareness and attention by developers as well as insufficient tools and methods are the most frequently used aspects to motivate research work. Less frequently used were argumentation lines about low accessibility in existing UI and the potential of adaptable UI to make them more accessible at an individual user level. There are a number of other technical aspects that have been used to motivate reviewed studies as shown in Table 12. Only one of 30 studies (S8) took a human-centric approach and motivated the presented research with real-world impact for disadvantaged people.

Table 12. Main motivational aspects mentioned in the primary studies

Number of studies	Motivation	Study keys
7	Insufficient tools and methods for implementing accessibility	S14, S15, S18, S20, S24, S25, S27
6	Lack of awareness and attention to accessibility needs by developers	S3, S4, S9, S10, S21, S29
4	Existing barriers to accessibility in UI	S6, S13, S17, S28
4	Adaptable UI have the potential to improve individual accessibility	S1, S11, S22, S30
2	Need for multi-platform support for addressing accessibility needs	S5, S26
2	Low accessibility of mobile apps	S2, S7
2	Extension of Egoki system will simplify the creation of accessible apps	S12, S19
1	Wikis lack built-in accessibility support	S23
1	Lack of accessible Web content	S16
1	Accessibility is essential for user inclusion and equality	S8

The predominant focus on technology to the detriment of human aspects becomes more obvious when looking at the number of studies that directly refer to user needs when motivating their work. Only 2 of 30 studies refer to visual impairments – the user needs in focus of this SLR. Van Hees and Engelen (S17) refer to blind users, while Khan and Khusro (S6) use visually impaired and blind users to motivate their work. Neglecting user needs in favor of technology is as worrisome as it is a hindrance when designing solutions aimed at improving human-centric aspects, or more specifically the accessibility, of software systems.

Most studies are driven by technological limitations, such as developers' lack of awareness or inadequate tools. Human-centric motivations—like real-world impact or addressing specific user needs—are rare. Only one study directly aimed to help disadvantaged users, and just two considered the needs of visually impaired users.

4.5.2 Reported Strengths of Promising Approaches and Tools (DP 4.1 in Table 16, Appendix B). We observed a high frequency of early integration and evaluation of accessibility features, due to the use of MDE methods. Studies S7, S8, S26, S27 and S18 mention that the benefits of MDE can be transferred to the development methodologies for accessible applications and yield similar results, such as an "increase in productivity" (S2), "improved development time" and higher flexibility during the development. Secondly, several studies reported a simplification and of accessibility standard compliance by leveraging MDE methods (S2, S7, S17, S18, S24). By creating high-level abstractions of software, these methods provide a comprehensive way to address accessibility requirements while reducing complexity. The separation of problem domains allows the developer to focus on the developed application itself while reducing the required knowledge for the accessibility domain. Finally, a strength that emerged from the studies was enhancing user efficiency and satisfaction, by using MDE (S23). By easing the development of accessibility functionality, the threshold to invest the required resources to do so is lowered thus resulting in a broader implementation of accessibility within the applications.

The reviewed studies highlight several strengths of MDE for accessibility: early integration of accessibility features, improved productivity and flexibility, easier compliance with standards through high-level abstractions, and enhanced user efficiency.

4.5.3 Reported Limitations (DP 4.1 in Table 16, Appendix B). In general, the number of reported shortcomings and limitations is surprisingly low. Based on the information provided, we have identified eight categories, shown in Table 13.

Table 13. Limitations mentioned in studies.

Number of studies	Mentioned limitation	Study keys
6	Limited evaluation	S1, S8, S11, S12, S18, S25
4	Limited scope	S18, S25, S29, S30
2	Limited user experience	S4, S27
3	Modeling difficulties	S14, S23, S24
1	Too complex/ Difficult to use	S17
4	Limited level of maturity	S8, S10, S13, S27
2	Limited level of adaptability	S11, S25
2	Practical limitations in real-world projects	S2, S16

Six primary studies (S1, S8, S11, S12, S18, S25) reported evaluation limitations. S18 discusses three shortcomings. Firstly, they point out that early-stage testing is difficult because most WCAG criteria are content-related, and thus the testing of early accessibility integration cannot be very comprehensive. Secondly, “only a subset of accessibility criteria can be tested” in an automated manner. Manual audit is still required, which impedes potential gains of an MDE approach. Thirdly, their evaluation is focused on “assistive technologies (AT) for visual impairments” only. With this, the authors indicate limitations regarding the applicability of their findings to non-visual impairment related accessibility needs. S11 highlights that in both their case studies, only one adaptation rule was used and tested. The support of multiple adaptation rules is earmarked for future work including more evaluations, especially to test potentially conflicting rules. Moreover, more evaluation is required to test incorporated transformations at different abstraction levels. S8 suggest that an automated evaluation has shortcomings by stating that the evaluation by specialists or a test with users, would probably reveal more problems and conclude “that tests with real users are essential for the identification of other accessibility problems“ These evaluation limitations are of a more serious nature because they limit not only the ability to make statements about the accessibility of created apps but also statements about the reproducibility of presented approaches.

Four studies (S18, S25, S29, S30) mention scope limitations or incomplete coverage of accessibility requirements. S18 highlight that their work was focused on assistive technologies (AT) for visual impairments only, implying the generalizability of their results must be carefully tested. S29 and S30 argue in a similar fashion by stating that basic concepts may be applicable to other accessibility needs than visual impairments but given limitations with supporting interactions and limited multimedia coverage.

S4 and S7 report limited user experiences of their approaches. S4 states that "while users with high experience with computing platforms were effective with the UI, users with low experience were not as effective." S7 explains that their"design is still limited to simple workflows and user interface elements, but can be extended in further versions".

Modeling difficulties were reported by three studies (S14, S23, S24) incl. the semantics of UI models are hard to generate and need further developer input to provide good results (S14), difficulties to build meta-models that generalize problems (S23), and WebML models that restrict the representation of rich interactions on web pages (S24).

Four studies (S8, S10, S13, S27) report a limited level of maturity and a need for future extensions. S7 explains that the designs that can be generated are "limited to simple workflows and user interface elements". S10 reports a need for the "improvement of [their] concept concerning the utility and usability of the model generation and transformation." S13 reports missing tooling (graphical editor) to better support developers. S27 remarks that tool support is only available for Java and PHP programming languages and only considers WCAG 2.0. All the above shortcomings are of either technical or conceptual nature and can be remedied with a reasonable amount of effort.

S2, S16 draw attention to the practical limitations of their research approaches in real-world projects. The essence of their remarks boils down to not just focusing on standard compliance and technological solutions but also considering operational procedures and other such factors. Again these issues require further research and user-centred design and evaluation.

Few of the 30 reviewed studies report limitations. The most common are limited evaluation and scope, with some noting issues like immature tools, modeling challenges, and restricted user experience. Key concerns include the lack of real-user testing, narrow focus on visual impairments, and low applicability to real-world projects.

4.5.4 Reported Gaps and Challenges. The research gaps stated in the primary studies (DP 4.1 and DP 4.2 in Table 16, Appendix B) revolve around methodologies for the new implementation and optimization of existing accessibility and the improvement of maintenance in MDE. Only three studies target MDE approaches directly (S23, S24, S28) while the remaining studies have a stronger emphasis on general improvements of accessibility.

(1) Introducing accessibility: In general, studies focus on the simplification of the development of accessible desktop applications and mobile apps, by presenting new frameworks, languages, or methodologies. Studies S4, S14, and S17 focus on the incorporation of accessibility features into the UI. S26 presents integrated models to introduce accessibility. S9 and S27 focus on the entire architecture of the targeted application in order to introduce accessibility features into the targeted applications, whereas S8 presents a methodology for the integration.

(2) Improving accessibility coverage: Ten studies (S2, S3, S5, S6, S7, S10, S11, S13, S15, S28) focus on expanding the accessibility coverage of existing approaches and frameworks. Studies in this group focus on increasing the number of covered WCAG guidelines of a given approach or reducing the required resources or expertise to meet those guidelines. The studies S11, S13, and S15 discuss adaptation rules to do so. User interfaces are present on a large variety of devices,

S3 and S5 focus on cross-platform coverage of accessibility guidelines, and S2, S7 and S10 focus especially on mobile applications. The studies S28 and S30 focus on the improvement of guideline coverage for web applications.
(3) Technical / maintenance challenges: Studies S27, S20 and S21 discuss the benefits of concepts and methodologies for the model-based development of accessible applications. S18 investigates an approach to evaluate accessibility in the very early stages of application development and discusses several requirements that should be met during a development process for an application that adheres to accessibility guidelines. In general, these studies claim, that existing techniques do not provide enough support for easy and agile software development of accessible applications. The approach focuses on simplifying the inclusion of accessibility aspects into the targeted software. The analyzed studies present the effects of model-driven or at least model-based development on the implementation of accessible applications, however, the studies differ in the specific guidelines that were chosen and the amount of guidelines that were covered.

4.5.5 Open gaps and challenges from accessibility and MDE perspectives (DP 4.3 in Table 16, Appendix B). We aimed to investigate what specific gaps and open challenges were addressed from an MDE perspective, however, our results showed that the gaps and challenges were rather coming from the accessibility perspective. Out of 30 studies, 27 focused on accessibility as the main research perspective and only used MDE as a method to support this. As a result, these publications do not mention MDE challenges and gaps explicitly. E.g., S8 mentions that web accessibility is not a priority in development projects, that software engineers lack technical knowledge and tools supporting accessibility quickly and simply, and that insufficient resources are allocated for software development projects. This leads the authors to suggest the use of MDE to support the execution of repetitive tasks, reduce the complexity, and improve productivity. S21 sees gaps rather in ubiquitous system design than in MDE, i.e., that the interfaces downloaded to a user's mobile device are usually the same for all users and often inaccessible for many of them.

Only 3 studies focus on improving MDE approaches to derive accessible applications. S3 explicitly integrates accessibility concerns into the model-driven process of accessible mobile apps and discusses issues that can be automatically handled in the transformations. S24 aims to improve the WebML modeling language to become capable of mapping WCAG guidelines to requirements and describes how MDE transformations are derived. S25 has the aim to develop modeling techniques for handling the non-functional, generic, and crosscutting characteristics of accessibility concerns.

The large majority of studies focused on open gaps and challenges from the accessibility perspective, not from the MDE perspective. This focus can be one of the reasons why none of the primary studies were published at main MDE venues.

RQ5 Summary: The studies on MDE for accessibility highlight strengths such as early integration of accessibility features, improved development productivity, and simplified compliance with accessibility standards through high-level abstractions. Limitations include restricted evaluation due to limited automated testing and content-specific criteria, narrow application scope focused mainly on visual impairments, and technical challenges like complex modeling and insufficient tool support. Identified research gaps involve expanding accessibility standard coverage, improving development methodologies, and addressing maintenance challenges for broader applicability.

5 DISCUSSION AND THE ROAD AHEAD

Our analysis of the 30 primary studies reveals several shortcomings and areas where further research is needed. While the reviewed studies offer valuable insights and innovative methodologies, limitations such as underspecified modeling constructs, incomplete transformation descriptions, and missing tooling artifacts restrict reproducibility and broader applicability to related application contexts.

Despite these constraints, the studies present a wealth of interesting insights and significant potential for advancing the field. By addressing the identified issues (such as providing concrete models, well-documented transformation rules, and reusable tool support), future research can contribute to solutions that are robust in the sense of behaving consistently across different UI technologies and assistive settings, transparent by making modeling decisions, assumptions, and processing steps explicit, and generalizable by being applicable beyond a single prototype or narrowly defined use case. This chapter highlights these opportunities, discusses the limitations, and proposes directions for future work to drive progress in this domain.

5.1 Identified Limitations

Limitation 1: Measuring Achieved Accessibility. Measuring how much of WCAG each study actually implements is extremely difficult because most studies neither specify which conformance level they aim for nor report how many guidelines they meet. Instead, they offer isolated examples—often without any explanation of why those particular checkpoints were chosen—and follow up with thorough, user-based evaluations to validate their work. In our review, we looked at whether studies covered guidelines broadly or focused deeply on a few ("horizontal versus vertical coverage"), whether they sampled problems by complexity or simply picked examples at random, and how mature their implementations appeared to be. We found that many studies fall into a "vertical prototype" approach, demonstrating one or two recommendations with no rationale for their selection. A few claim full compliance based on automated checkers, e.g., S7 uses a free web accessibility checker [3], but such tools miss numerous real-world issues, while others fit neither category and leave their actual contributions to accessibility vague and undefined. The core challenge lies in the very nature of WCAG itself: it abstracts diverse human needs into technical checkpoints, obscuring the nuances of real-user experience. Automated tools can only catch code-level violations, and many guidelines—such as writing meaningful alternative or descriptive text—demand subjective, manual assessment. Ultimately, ticking off criteria does not guarantee that users can accomplish tasks.

Limitation 2: Generalizability and Reproducibility of Approaches. All analysed studies present their approaches and methodologies; however, not all approaches can be reproduced without further information or access to sources from the authors. In 14 of the studies (S4, S5, S9, S11, S15, S17, S18, S20, S21, S23, S24, S26, S27, 28), the technical details remain unclear. Further, 4 studies (S18, S20, S21, S30) do not state a systematic or detailed discussion. Page and word limits for study submissions are a likely reason for authors to omit this in-depth information in favour of a more detailed description of their findings. The fact that these data are missing makes it almost impossible to reproduce the results or to build on the findings of the work. In addition, although the studies present working methodologies and solutions, they often lack clear reasoning as to why specific steps were taken, making it harder to generalise the presented approach and apply it to similar use cases.

Limitation 3: Limited Evaluation with Key Target User Groups. Only a few of the inspected studies provided an evaluation of their approaches with real target end users (S1, S8, S11, S12, S18, S25) and most lacked a precisely defined user groups, which made a clear evaluation with a specific target group very difficult. It is often unclear if the presented approach provides a benefit beyond those presented in theory. In addition, a stronger focus on both sides of the stakeholders (developers and end users) is often lacking: The studies often either discuss the benefits for the user (e.g. enhanced user experience) or the benefits for the developer (e.g. higher efficiency during development). A simultaneous consideration is missing.

Limitation 4: Lack of Details and Reusability of the Presented Approaches. A significant number of studies (S4, S10, S15, S18, S20, S21, S28, S27) address MDE and vision impairments at a conceptual, abstract, or generic level. Only 4 studies (S2, S5, S7, S18) state their objectives with clearly formulated research questions. This means that many of the presented studies lack the necessary details to comprehend the implementation in such detail that it can be reproduced. Such details are, e.g., which modeling formalisms are used, which modeling constructs are relevant for improving accessibility, what parts of applications are generated, together with providing access to concrete code artifacts (models, transformation rules, code templates, code generators, the used RTE and reusable RTE components, or generated applications). Authors should provide reusable artifacts to strengthen research that can build on each other and provide impact in practice. Therefore, we would like to encourage the authors to provide access to repositories and specifications to help advance this field of research. Publishing in core MDE venues and submitting longer publication formats such as journal articles would also help mitigate this limitation, as these venues require more information about the used modeling methods, tooling, and artifacts, provide feedback for improvements in the review rounds, and have the needed space for in-depth descriptions.

Limitation 5: Accessibility as a Prerequisite Competence. The presented approaches focus on the inclusion of accessibility guidelines in MDE approaches. A key aspect of MDE approaches is their potential for automation. Although the presented approaches include many automation aspects, all of them still rely highly on the developer to provide accessibility expertise. It is challenging to provide full automation in an MDE approach, however, there is a high potential to increase the automation of accessibility related tasks in the development process and thus reduce the time and effort spent by developers on building up expertise in accessibility needs and requirements.

Limitation 6: Coverage of Accessibility Needs and Guidelines. The degree to which accessibility needs are covered by a given approach is hard to measure, and thus, reliable coverage is hard to reach: Very few studies address the specific guidelines that are covered (S1-S3, S6, S10, S17). In addition, the technical perspective often dominates: specific guidelines are often covered as a proof of concept rather than to address a complete scenario for a specific visual impairment. Thus, the effectiveness of the approach is hard to evaluate. However, it should be noted that complete coverage of all guidelines may neither be required nor attainable within a specified use case, as the guidelines can conflict each other. A social perspective, rather than a technical one, offers advantages in this context by directing attention to the guidelines most relevant for a specific visual impairment use case and target users and developing adaptable systems.

Limitation 7: Lack of connection to the MDE community. Our analysis has shown that publications about MDE and vision impairments are scarce, have slightly decreased since 2019, and are largely absent from main MDE venues. This can have several reasons: (1) While the topic of vision impairments and MDE fits topic-wise very well to MDE venues (c.f. [16]), a substantial portion of MDE research has focused on cyber-physical systems, digital twins of physical assets, and systems-of-systems. This indirectly contributes to so-called 'softer' topics receiving less attention from MDE researchers. However, this is also directly related to (2): Is there enough funding for research on accessibility? Research on vision impairments or more general accessibility and MDE requires funding over a longer period of time to develop methods, tooling for a working prototype, and evaluate it with key target user groups. It might not be easy to get industry funding for this topic, and some countries even restrict research on accessibility and inclusion. The absence of sustained research programs limits the visibility and continuity in core venues. (3) Accessibility research in general is more commonly published in human-computer-interaction, user experience, or assistive technology venues [64]. Thus, researchers working on accessibility may use models implicitly rather than as first-class artifacts, and might lack familiarity with core MDE venues. This is related to (4) the application of MDE for vision impairments is considered applied research. While MDE venues have a strong focus on modeling foundations and the improvement of MDE methods (so the meta-level), most of them are open for work on the application of MDE. Thus, publications sent to MDE venues should make this focus explicit, and authors should make sure to send their work to, e.g., the right conference tracks (if there are foundational and applied ones). (5) Even though not all MDE venues might require a full evaluation with key target user groups (see limitation 4), a lack of such evaluations hinders getting publications placed at higher-ranked MDE venues. These evaluations are not easily obtainable in practice compared to simulations, formal analysis, or performance benchmarks; however, they are crucial to show the acceptance of such approaches.

5.2 What could an effective MDE contribution for visual impairments entail?

We believe it is possible to automate the implementation of (large parts of) systems coping with accessibility needs for users with visual impairments. The following list shows what an effective MDE contribution for visual impairments could entail and what we expected to find in the primary studies [2]:

(1) *The concrete type of impairment being addressed*, e.g., low vision, color blindness, with concrete research question(s) linked to it.
(2) *Requirements that are traceable to guidelines* and *models reflecting accessibility requirements* that are fitting to the research questions. This can include constraints on UI elements, e.g., minimum contrast ratios, scalable font sizes, focus order constraints, or non-visual alternatives, as well as non-visual interaction requirements, e.g., mandatory keyboard access, audio descriptions, or semantic labeling.
(3) *Domain model concepts* capturing variability points and context information for visually impaired users. This represents relevant data structures after code generation processes. E.g., main and alternative devices, screen reader flow, color contrast thresholds, or touch target sizes.
(4) *Models capturing relevant structural and behavioral user information* such as preferences, contextual variability, and capabilities, e.g., residual vision, color perception, reliance on screen readers, preference for tactile/audio feedback (cf. Digital Human Twins [81] or modeling humans as parts of socio-technical ecosystems [15]).
(5) *UI models and their extensions*, e.g., semantic annotations for names, roles, or navigation orders. This could include models for interaction and navigation with the interfaces.
(6) *Formal transformation rules*, e.g., defined using the Atlas Transformation Language (ATL), or *code templates*, e.g., defined as Apache FreeMarker templates, and *code generators* to map accessibility requirements captured in models to source code together with a description of the *generation pipeline and other artifacts for automation* (e.g., Gradle scripts, GitLab runners). Such generative approaches could cover the generation of one application that is adaptable during runtime for the needs of specific user profiles or the generation of different variants for particular needs (especially when they are conflicting), e.g., a version with simplified graphical layouts for low-vision users and an audio version for blind users in addition to an application for all other users.

[2]Please note that this list is not exhaustive and research might focus on particular aspects in this list, e.g., model-based or model-driven testing of accessible applications.

(7) *Architectural components and architecture models* of the application. Some components could be reusable throughout different applications (c.f. [69]) and serve as the RTE in generative approaches or could be added to generated applications over time and identified as being reusable for other applications (similar to building a software product line based on a reference architecture), e.g., a screen reader component, a text shortening or simplifying component, a color switch component.
(8) *Adaptation rules* that map user profiles to variants in the UI within systems that are adaptable during runtime, as well as descriptions of the *architectural components* needed to realize this adaptation.
(9) *Test cases* for checking if visual impairment requirements are covered in the generated application and reusable test frameworks. This could also include methods to automatically provide test protocols that can be used in user evaluations.
(10) *Empirical evaluation* of the results with realistic scenarios and key target users defined in the research goals. This could include qualitative and quantitative methods.
(11) *Generalizability strategies.* As the accessibility needs and severity of visual impairments differ from person to person, and often change with time and situation, systems need to be tailored to these individual needs. Thus, approaches shall show how applicable they are for various needs and describe generalizability strategies on how an approach can help users with similar needs and beyond.

Reuse. The developed *models* stated in points 2-9 of this list (or parts of the models, e.g., parts of the domain model describing the input and output modalities, triggered actions, disability profile ontologies, devices, and preference types as described in S4) are the *main reusable artifacts.* These models could be realized using one or more GPLs and DSLs that are tailored to the needed concepts (i.e., requirements, user preferences and abilities, restrictions, context information such as devices and their capabilities, semantic annotations, workflows). How these models are then treated in the development pipeline can largely vary depending on the type of impairments to be supported, the application domain, and the used technologies. Beyond the models, other *reusable artifacts* are typically code generators and architectural components. Also, transformation rules, adaptation rules (as stated in S11), or code templates can be reusable, together with concrete software components or test infrastructures. These artifacts are already connected to a concrete technology, i.e., programming language, or frontend framework. Thus, their reuse depends on the technology constraints of a given project. In addition, one can *use artifacts and patterns from the analysis and design phase as inspiration*, e.g., common accessibility problems in mobile applications that lead to requirements as stated in S2 and S3. These aspects are technology independent; however, as specific user needs and application domains differ, full reuse might not be possible. Other *non-reusable artifacts* are typically concrete implementation logic added in hand-written code or test cases (especially ones that are not generated from models) together with application-specific configurations, runtime data, and evaluation data. However, they might still serve as inspiration and provide ideas for best practices in the long run.

5.3 Research Roadmap

The field of vision impairment in software development continues to evolve, presenting numerous opportunities to address existing challenges and explore new frontiers. This subsection outlines key areas for future research and innovation (see Figure 4) with a focus on vision-related accessibility needs.

Analysis Phase.

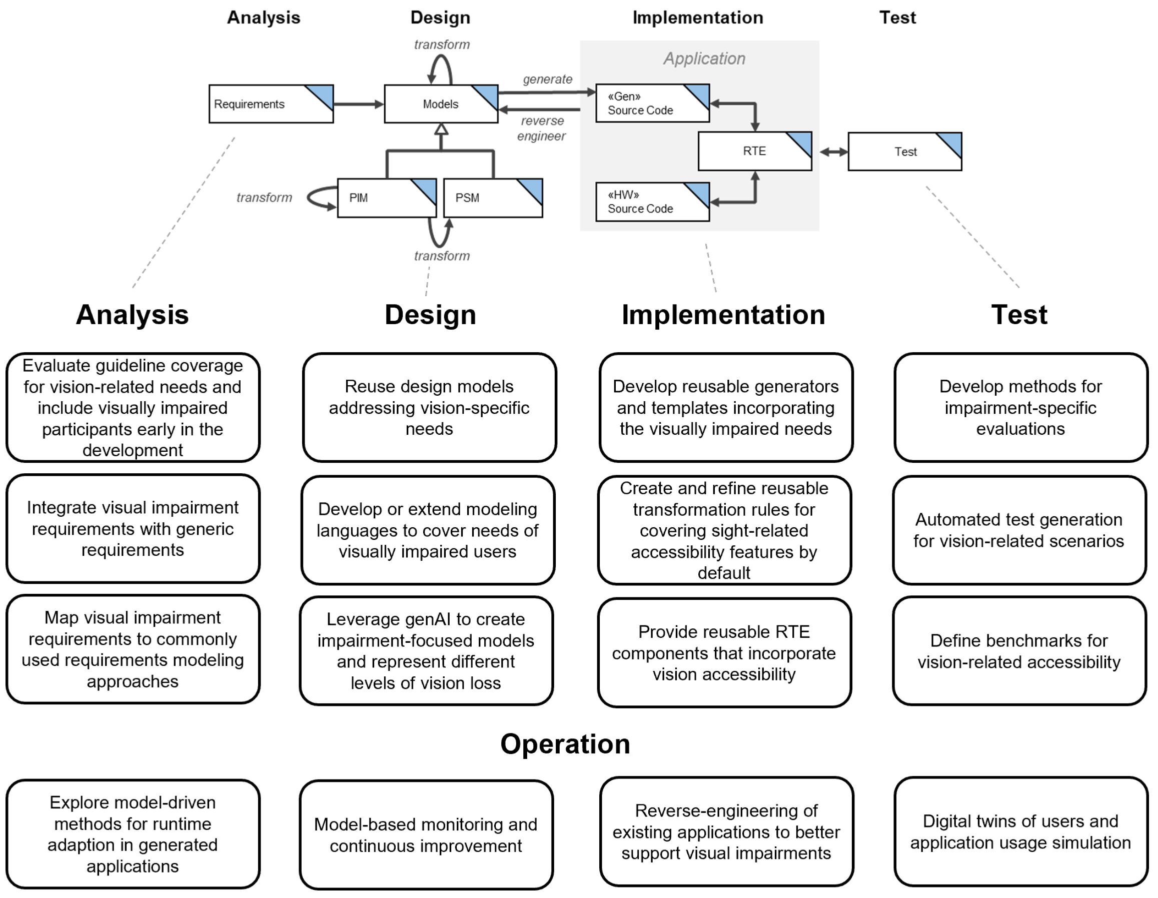

Analysis
Design
Implementation
Test
transform
Application
generate
Requirements
Models
«Gen» Source Code
reverse engineer
RTE
Test
transform
PIM
PSM
«HW» Source Code
transform
Analysis
Design
Implementation
Test
Evaluate guideline coverage for vision-related needs and include visually impaired participants early in the development
Reuse design models addressing vision-specific needs
Develop reusable generators and templates incorporating the visually impaired needs
Develop methods for impairment-specific evaluations
Integrate visual impairment requirements with generic requirements
Develop or extend modeling languages to cover needs of visually impaired users
Create and refine reusable transformation rules for covering sight-related accessibility features by default
Automated test generation for vision-related scenarios
Map visual impairment requirements to commonly used requirements modeling approaches
Leverage genAI to create impairment-focused models and represent different levels of vision loss
Provide reusable RTE components that incorporate vision accessibility
Define benchmarks for vision-related accessibility
Operation
Explore model-driven methods for runtime adaption in generated applications
Model-based monitoring and continuous improvement
Reverse-engineering of existing applications to better support visual impairments
Digital twins of users and application usage simulation

Fig. 4. Current gaps and potential areas for further research

- **Evaluate guideline coverage for vision-related needs and include visually impaired participants early in the development.** Investigate the coverage of specific guidelines and standards to ensure comprehensive support for accessibility requirements for different types of visual impairments (e.g., blindness, low vision, color blindness). By systematically examining the applicability and thoroughness of these guidelines, we can identify gaps and explore guidelines that are less often covered by research and reveal which impairments are underrepresented and need additional guidance. When gathering requirements, consider incorporating user studies with visually impaired participants as early as possible in the development process to refine domain-specific needs beyond standard guidelines.
- **Integrate visual impairment requirements with generic requirements.** Explore the integration of specific accessibility requirements for different types of visual impairments with generic requirements for software projects, aiming for seamless alignment. The aim is to identify potential conflicts between accessibility guidelines and generic requirements, such as requirements for graphical dashboards vs. needs for non-visual interaction,

early in the development process. Early analysis should make it explicit how features translate to voice, tactile or high-contrast modalities.

- **Map visual impairment requirements to commonly used requirements modeling approaches.** Provide reusable mappings from impairment-specific needs, e.g., text-to-speech compatibility, zoom support, haptic alteration, to requirements modeling approaches. This aims to bridge the gap between accessibility needs and traditional requirements engineering practices. These mappings can facilitate the understanding and systematic incorporation of accessibility in mainstream requirement engineering tools and methods.

Design Phase incl. Modeling.

- **Reuse design models addressing vision-specific needs.** Investigate how to reuse requirements models that address the accessibility needs of users with vision impairments in the design phase to improve efficiency and consistency. Reusable models can act as templates (i.e., including constraints for screen reader compatibility, font scaling, contrast ratios or multimodal alternatives such as audio or tactile representations), reducing effort and ensuring that visually impaired users are systematically supported across multiple projects.
- **Develop or extend modeling languages to cover needs of visually impaired users.** Evaluate and enhance or develop modeling languages to better represent specific accessibility needs, i.e., annotating color-independent interactions, providing alternative modalities, tagging semantic descriptions of UI elements, or dependencies on screen readers, in software design. By adapting and extending existing modeling languages, developers can more effectively articulate and document accessibility requirements within their designs.
- **Leverage genAI to create impairment-focused models and represent different levels of vision loss.** Assess the potential of generative AI tools to define models that effectively address specific accessibility needs, i.e., by configuring prompts or datasets to focus on UI/UX best practices for vision impairments, such as voice navigation flows or haptic feedback models, or to propose alternative interaction flows. Leveraging AI can help automate and streamline the creation of tailored models, making it easier to incorporate accessibility and provide variants of solutions representing different levels of vision loss.

Implementation and Generation.

- **Develop reusable generators and templates incorporating the visually impaired needs.** Our developed generators and templates should specifically address accessibility needs as defaults, i.e., to produce UIs optimized for screen readers, voice control, and magnification tools, facilitating consistent implementation. These resources can save time and ensure that accessibility features are implemented uniformly across different projects and teams.
- **Create and refine reusable transformation rules for covering sight-related accessibility features by default.** Applying these rules should enforce, e.g., contrast, scalable texts, alternative text descriptions, color-independent interaction, and keyboard or voice navigation pathways by default, and ensure compatibility with assistive technologies such as screen readers or braille displays. Transformation rules can help automate the process of adapting generic components to meet accessibility standards, reducing manual effort and errors.
- **Provide reusable RTE components that incorporate vision accessibility.** Developers should design reusable runtime environment components, such as GUI components, that include settings for specific accessibility needs, e.g., high-contrast themes, text-to-speech toggles, or magnification. These components can simplify the

development process by providing pre-built, accessible elements that developers can easily integrate into their projects without redesigning accessibility features from scratch.

Testing Phase.

- **Develop methods for impairment-specific evaluations.** We have to develop methods to enable developers to effectively test the level of support and the coverage of accessibility for visually impaired end users. This includes creating frameworks and tools to assess how well development tools support accessibility requirements and how accessible the resulting software is for various user groups, e.g., by simulating low-vision conditions, or integrating real screen reader or haptic device testing.
- **Automated test generation for vision-related scenarios.** One shall investigate automated test generation techniques to evaluate accessibility features and ensure compliance with accessibility standards. This requires including usage scenarios for visually impaired users, e.g., where missing visual clues are backed by auditory feedback, and generating test cases that check for missing alternative texts, non-scalable fonts, poor contrast ratios, or inaccessible navigation flows. Automation can enhance the efficiency and reliability of accessibility testing, allowing developers to identify and address issues more quickly.
- **Define benchmarks for vision-related accessibility.** The requirements coverage of developed systems should be measured against vision-related accessibility benchmarks, e.g., how well the software functions without visual input or under low-contrast conditions, measuring the proportion of UI elements with semantic annotations, or the availability of auditory or tactile alternatives for visual feedback.

Operation of the Application.

- **Explore model-driven methods for runtime adaptation in generated applications.** We need to develop approaches that cover both code generation and model interpretation during runtime to tailor applications better to the heterogeneous needs of users. These adaptations could be based on models@runtime [12] describing user profiles and their preferences. One could also explore incremental model transformation within a continuous software engineering pipeline, so with constant delivery of tailored applications in cases where applications cannot be easily adapted to fit everybody during runtime.
- **Model-based monitoring and continuous improvement.** Using monitoring information during system operation (e.g., observe accessibility-relevant runtime properties, such as focus traversal behavior, error rates in non-visual navigation, or interaction patterns indicating usability barriers for visually impaired users), one could improve runtime models describing user profiles. This could then trigger adaptations for the particular user or be used to improve applications in the long run. These feedback loops enable the continuous improvement of accessibility support beyond design-time assumptions.
- **Reverse-engineering of existing applications to better support visual impairments.** In practice, applications are often not developed from scratch but exist for a long time. This calls for model-driven reverse-engineering approaches that support the addition of functionalities for visually impaired users within existing (brown-field) applications.
- **Digital twins of users and application usage simulation.** Model-driven approaches can be used to create and maintain digital twins of users that capture different types and degrees of visual impairments, i.e., blindness, low vision, and color vision deficiencies. Such user-centric digital twins could be employed during the operation of an application to simulate how generated or adapted user interfaces are perceived and interacted with under

varying needs. Simulation results may support runtime decision-making (e.g., selecting appropriate UI variants), adapting the application accordingly, and provide evidence for assessing accessibility without relying exclusively on costly user studies.

More recent approaches explore how genAI can improve the development processes in general. These methods can be applied in all of these phases to further explore how to support each step. Thus, genAI can be seen as cross-cutting methods used to explore how they might be able to improve the development of applications for visually impaired users.

6 THREATS TO VALIDITY

Although this study followed well-established best practices for a systematic literature review [56, 103], and great care was taken during its execution, we acknowledge that our research may have been exposed to the following threats.

By following Kitchenham et al. [56] and Wohlin [103] for the study design, we minimized threats to internal validity. To ensure all relevant studies were identified, we conducted several trial runs and refinements of our query design to make sure database-specific formats, operator precedence, etc were honored. We also incorporated a systematic snowballing procedure according to Wohlin [103] to counter the exclusion of SpringerLink (see Section 3.2 for the reasons of the exclusion). To ensure only studies of suitable quality were considered, we rigorously assessed all studies by (i) removing studies of four or less pages (because space limitations prevent the provision of necessary detail), (ii) filtering out non-peer-reviewed studies, and (iii) carrying out quality assessments during all iterations of the selection procedure. One author performed the database search to keep the search queries throughout the repeated search consistent, all other activities were assigned to and carried out by one randomly assigned author per study, followed by regular team discussions of the results and randomly assigned cross-checks. Threats during study execution were mitigated in a similar fashion: All authors met on a weekly basis during study selection, and data extraction, analysis and synthesis to discuss and verify each others results based on randomly assigned cross-checks. All collected information was stored in a single Cloud-based document. In this way, we minimized threats arising from the manual nature of the majority of these tasks.

To ensure that our selected studies are representative of the state-of-the-art in MDE approaches that address visual impairments, we paired two IT-specific online databases (ACM, IEEE) with two databases that cover a wider range of disciplines (ScienceDirect, Scopus). Further, we carried out forward and backward snowballing to identify studies that did not register hits during the database search. The exclusion of short studies and studies not written in English may affect the representativeness of our set of selected studies. However, we argue that the former are too short to provide sufficient detail, while the latter are rare. English is the SE lingua franca and thus the loss of relevant studies in other languages is unlikely. Finally, we decided not to consider gray literature. Although there is a risk of missing relevant studies, we felt that focusing on peer-reviewed scientific studies represents a good trade-off in favor of the quality of selected studies.

Another possible limitation are the keywords included in our search terms (cf Section 3.3). To mitigate this threat we included 'accessibility' to catch relevant studies that did not use any of the visual impairment keywords in our search term. On the other hand, we omitted MDE-related keywords such as 'MDA' or 'MDD' from our search term because their inclusion did not result in additional relevant hits during trial, and to keep the search term simple and easy to execute.

7 CONCLUSIONS AND FUTURE WORK

While accessibility needs for visually impaired persons are often overlooked in developing socio-technical systems, an MDE approach should provide the needed developer support to incorporate them systematically and consistently throughout the complete application. This study presents a systematic literature review of 30 primary studies on the application of model-driven engineering for visual impairments selected from an initial pool of 447 papers. We have analyzed existing trends regarding timing, output, impact, and nature of the reported approaches, and what visual impairments they address. We investigated the MDE approaches and their development steps and evaluations in detail, and gave an overview of the reported strengths, limitations, gaps, and challenges. Key findings are that most studies to date operate at a high abstraction level, mainly rely on WCAG, and rarely provide reproducible pipelines or working software artifacts. Only a small number of studies provide concrete modeling approaches for accessibility requirements, functioning implementations, and evaluations with visually impaired end users. This limits reuse, hinders independent verification of the results and constrains the impact for practical adoption.

Only a few approaches target MDE methods, languages, transformations, code generators, or further tooling as the object of innovation and rather use MDE as an instrument. This might explain why contributions are often rather conceptual, short on implementation details (e.g., no publicly available metamodels, transformations, or generators), and underrepresented at core MDE venues. Consequently, the analysis has shown limited research outputs and low visibility. This calls for technically grounded work that explicitly specifies models, transformations, and tool chains, and that treats accessibility as a first-class concern from requirements to design and software solution. The applied methods are mixed: While WCAG is used by most of the approaches, and models commonly capture the UI, interactions, and navigation, only five studies consider modeling accessibility requirements (without being specific enough to enable reproduction). Evaluations are sparse: 10 studies have no evaluation, and only 6 conducted user studies, whereas only 3 of them involved visually impaired participants. None provided additional evaluation data packages to support replication of the evaluation. Measuring the achieved accessibility of applications remains challenging, as only using automated checkers is not sufficient and conducting user studies is time-intensive.

Using our results, we have sketched some possible research topics in analysis, design, implementation, and testing of accessible applications using MDE. Promising research directions include (i) analysis and design techniques that make accessibility requirements explicit, traceable and verifiable at the model level, (ii) reusable model transformations, code templates, code generators and runtime components that include accessibility, and (iii) testing approaches increasing the level of automation and checking for compliance with accessibility requirements. Exploring the interplay with methods from AI and generative AI provides additional possible research directions. Advancing the field will require publishing more implementation details and replication packages.

In summary, the current state-of-the-art demonstrates potential for improving accessibility of software applications with MDE methods. Providing more reusable artifacts on model, template, and code level and transparent evaluations with relevant user groups, the MDE community can improve development methods for accessibility for visual impairments, delivering an important impact on society.

ACKNOWLEDGMENTS

This work is supported in part by ARC Laureate Fellowship FL190100035.

REFERENCES

[1] Julio Abascal, Amaia Aizpurua, Idoia Cearreta, Borja Gamecho, Nestor Garay-Vitoria, and Raúl Miñón. 2011. Automatically Generating Tailored Accessible User Interfaces for Ubiquitous Services. In *The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility* (Dundee, Scotland, UK) *(ASSETS '11)*. Association for Computing Machinery, New York, NY, USA, 187–194. https://doi.org/10.1145/2049536.2049570

[2] Roberto Acerbis, Aldo Bongio, Marco Brambilla, Massimo Tisi, Stefano Ceri, and Emanuele Tosetti. 2007. Developing eBusiness Solutions with a Model Driven Approach: The Case of Acer EMEA. In *Web Engineering*, Luciano Baresi, Piero Fraternali, and Geert-Jan Houben (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 539–544.

[3] AChecker open-source project. 2025. Free Web Accessibility Checker – Instant WCAG 2.2 Audit & Accessibility Score. https://achecker.ca/

[4] United States General Services Administration. 2018. Information and Communication Technology Revised 508 Standards and 255 Guidelines. https://www.section508.gov/develop/universal-design/

[5] United States General Services Administration. 2025. Universal Design and Accessibility. https://www.section508.gov/develop/universal-design/

[6] Iván Alfonso, Aaron Conrardy, Armen Sulejmani, Atefeh Nirumand, Fitash Ul Haq, Marcos Gomez-Vazquez, Jean-Sébastien Sottet, and Jordi Cabot. 2024. Building BESSER: An Open-Source Low-Code Platform. In *Enterprise, Business-Process and Information Systems Modeling*, Han van der Aa, Dominik Bork, Rainer Schmidt, and Arnon Sturm (Eds.). Springer Nature Switzerland, 203–212.

[7] Wesley Tessaro Andrade, Rodrigo Gonçalves de Branco, Maria Istela Cagnin, Débora Maria Barroso Paiva, and Hideyuki Nakanishi. 2018. Incorporating Accessibility Elements to the Software Engineering Process. *Advances in Human-Computer Interaction* 2018 (2018), 17 pages. https://doi.org/10.1155/2018/1389208

[8] Australian Government. 2010. Web Accessibility National Transition Strategy (WCAG 2.0). https://www.adcet.edu.au/resource/7356/web-accessibility-national-transition-strategy-wcag-2-0

[9] Australian Institute of Health and Welfare (AIHW). 2021. Eye health - How common is visual impairment? https://www.aihw.gov.au/reports/eye-health/eye-health/contents/new

[10] Paul Baker, Shiou Loh, and Frank Weil. 2005. Model-Driven Engineering in a Large Industrial Context — Motorola Case Study. In *Model Driven Engineering Languages and Systems*, Lionel Briand and Clay Williams (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 476–491.

[11] BBC. 2014. BBC Mobile Accessibility Guidelines. https://www.bbc.co.uk/accessibility/forproducts/guides/mobile/

[12] Nelly Bencomo, Sebastian Götz, and Hui Song. 2019. Models@run.time: a guided tour of the state of the art and research challenges. *Software and Systems Modeling* 18, 5 (2019), 3049–3082. https://doi.org/10.1007/s10270-018-00712-x

[13] Yousra Bendaly Hlaoui, Lamia Zouhaier, and Leila Ben Ayed. 2019. Model driven approach for adapting user interfaces to the context of accessibility: case of visually impaired users. *Journal on Multimodal User Interfaces* 13, 4 (2019), 293–320. Issue 4. https://doi.org/10.1007/s12193-018-0277-z

[14] Thiago Jabur Bittar, Luanna Lopes Lobato, Renata P. M. Fortes, and David Fernandes Neto. 2010. Accessible Organizational Elements in Wikis with Model-Driven Development. In *Proceedings of the 28th ACM International Conference on Design of Communication* (São Carlos, São Paulo, Brazil) *(SIGDOC '10)*. Association for Computing Machinery, New York, NY, USA, 49–56. https://doi.org/10.1145/1878450.1878459

[15] Federico Bonetti, Antonio Bucchiarone, Judith Michael, Antonio Cicchetti, Annapaola Marconi, and Bernhard Rumpe. 2024. Digital Twins of Socio-Technical Ecosystems to Drive Societal Change. In *MODELS Companion '24: Int. Conf. on Model Driven Engineering Languages and Systems*. ACM, 275–286. https://doi.org/10.1145/3652620.3686248

[16] Dominik Bork, Stefan Klikovits, Judith Michael, Lukas Netz, and Bernhard Rumpe. 2025. Inclusive Model-Driven Engineering for Accessible Software. (October 2025), 253–259. https://doi.org/10.1109/MODELS67397.2025.00030

[17] Amina Bouraoui and Imen Gharbi. 2019. Model driven engineering of accessible and multi-platform graphical user interfaces by parameterized model transformations. *Science of Computer Programming* 172 (2019), 63–101. https://doi.org/10.1016/j.scico.2018.11.002

[18] Amina Bouraoui, Mohamed Jemni, and Mohsen Laabidi. 2007. E-learning and handicap: new trends for accessibility with model driven approach. In *Innovations in E-learning, Instruction Technology, Assessment, and Engineering Education*, Magued Iskander (Ed.). Springer Netherlands, Dordrecht, 451–454. https://doi.org/10.1007/978-1-4020-6262-9_78

[19] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. 2017. *Model-Driven Software Engineering in Practice.* Springer International Publishing, Cham. https://doi.org/10.1007/978-3-031-02549-5

[20] Alessio Bucaioni, Antonio Cicchetti, and Federico Ciccozzi. 2022. Modelling in low-code development: a multi-vocal systematic review. *Software and Systems Modeling* 21 (2022), 1959–1981. Issue 5. https://doi.org/10.1007/s10270-021-00964-0

[21] Constantin Buschhaus, Arkadii Gerasimov, Jörg Christian Kirchhof, Judith Michael, Lukas Netz, Bernhard Rumpe, and Sebastian Stüber. 2024. Lessons learned from applying model-driven engineering in 5 domains: The success story of the MontiGem generator framework. *Journal Science of Computer Programming* 232 (Jan 2024), 103033. https://doi.org/10.1016/j.scico.2023.103033

[22] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt. 2003. A Unifying Reference Framework for multi-target user interfaces. *Interacting with Computers* 15, 3 (2003), 289–308. https://doi.org/10.1016/S0953-5438(03)00010-9

[23] Alessandro Colantoni, Antonio Garmendia, Luca Berardinelli, Manuel Wimmer, and Johannes Bräuer. 2021. Leveraging Model-Driven Technologies for JSON Artefacts: The Shipyard Case Study. In *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. 250–260. https://doi.org/10.1109/MODELS50736.2021.00033

[24] Community. 2025. The A11Y Project. https://github.com/a11yproject/a11yproject.com

[25] Manuela Dalibor, Malte Heithoff, Judith Michael, Lukas Netz, Jérôme Pfeiffer, Bernhard Rumpe, Simon Varga, and Andreas Wortmann. 2022. Generating Customized Low-Code Development Platforms for Digital Twins. *Journal of Computer Languages (COLA)* 70 (2022). https://doi.org/10.1016/j.cola.2022.101117

[26] Davide Di Ruscio, Dimitris Kolovos, Juan de Lara, Alfonso Pierantonio, Massimo Tisi, and Manuel Wimmer. 2022. Low-code development and model-driven engineering: Two sides of the same coin? *Software and Systems Modeling* (2022). https://doi.org/10.1007/s10270-021-00970-2

[27] Arthur Floriano Barbosa Andrade de Oliveira and Lucia Vilela Leite Filgueiras. 2018. Developer Assistance Tools for Creating Native Mobile Applications Accessible to Visually Impaired People: A Systematic Review. In *17th Brazilian Symposium on Human Factors in Computing Systems (IHC 2018)*. ACM, New York, NY, USA, Article 16. https://doi.org/10.1145/3274192.3274208

[28] Leandro Rossetti de Souza, Rosemary Francisco, João Elison da Rosa Tavares, and Jorge Luis Victória Barbosa. 2024. Intelligent environments and assistive technologies for assisting visually impaired people: a systematic literature review. *Universal Access in the Information Society* (2024). https://doi.org/10.1007/s10209-024-01117-y

[29] Marianna Di Gregorio, Dario Di Nucci, Fabio Palomba, and Giuliana Vitiello. 2022. The making of accessible Android applications: an empirical study on the state of the practice. *Empirical Software Engineering* 27, 145 (2022), 37 pages. Issue 6. https://doi.org/10.1007/s10664-022-10182-x

[30] Felipe Dias, Lianna Duarte, and Renata Fortes. 2021. AccessMDD: An MDD Approach for Generating Accessible Mobile Applications. In *39th ACM International Conference on Design of Communication* (Virtual Event, USA) *(SIGDOC '21)*. ACM, 85–95. https://doi.org/10.1145/3472714.3473904

[31] Eclipse. 2006. Graphical Modeling Framework. http://www.eclipse.org/modeling/gmp

[32] Romina Eramo, Martina Nolletti, Luigi Pomante, Laura Pasquale, and Dario Pascucci. 2024. Model–driven engineering for simulation models interoperability: A case study in space industry. *Software: Practice and Experience* 54, 6 (2024), 1010–1033. https://doi.org/10.1002/spe.3309

[33] ETSI. 2021. EN 301 549 V3 the harmonized European Standard for ICT Accessibility. https://www.etsi.org/human-factors-accessibility/en-301-549-v3-the-harmonized-european-standard-for-ict-accessibility Version 3.2.1 at https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.02.01_60/en_301549v030201p.pdf.

[34] European Commission. 2025. European accessibility act. https://commission.europa.eu/strategy-and-policy/policies/justice-and-fundamental-rights/disability/union-equality-strategy-rights-persons-disabilities-2021-2030/european-accessibility-act_en

[35] European Parliament and Council. 2016. Directive (EU) 2016/2102 of the European Parliament and of the Council of 26 October 2016 on the accessibility of the websites and mobile applications of public sector bodies. https://eur-lex.europa.eu/eli/dir/2016/2102/oj

[36] International Organization for Standardization. 2004. *Software engineering — Product quality; Part 4: Quality in use metrics* (iso/iec tr 9126-4:2004 (withdrawn) ed.). International Organization for Standardization, Geneva, Switzerland. https://www.iso.org/standard/39752.html

[37] International Organization for Standardization. 2008. *Ergonomics of Human System Interaction - Part 171: Guidance on software accessibility* (iso 9241-171:2008 ed.). International Organization for Standardization, Geneva, Switzerland. https://www.iso.org/standard/39080.html

[38] International Organization for Standardization. 2012. *Information technology — W3C Web Content Accessibility Guidelines (WCAG) 2.0* (iso/iec 40500:2012 ed.). International Organization for Standardization, Geneva, Switzerland. https://www.iso.org/standard/58625.html

[39] International Organization for Standardization. 2017. *Systems and software engineering — Software life cycle processes* (iso/iec/ieee 12207:2017 ed.). International Organization for Standardization, Geneva, Switzerland. https://www.iso.org/standard/63712.html

[40] Borja Gamecho, Raúl Miñón, Amaia Aizpurua, Idoia Cearreta, Myriam Arrue, Nestor Garay-Vitoria, and Julio Abascal. 2015. Automatic Generation of Tailored Accessible User Interfaces for Ubiquitous Services. *IEEE Transactions on Human-Machine Systems* 45, 5 (2015), 612–623. https://doi.org/10.1109/THMS.2014.2384452

[41] Anderson Canale Garcia, Silvana Maria Affonso de Lara, Lianna Mara Castro Duarte, Renata Pontin de Mattos Fortes, and Kamila Rios Da Hora Rodrigues. 2023. Early accessibility testing – an automated kit for Android developers. In *29th Brazilian Symposium on Multimedia and the Web* (Ribeirão Preto, Brazil) *(WebMedia '23)*. ACM, New York, NY, USA, 11–15. https://doi.org/10.1145/3617023.3617028

[42] Peter Göhner, Simon Kunz, Sabina Jeschke, Helmut Vieritz, and Olivier Pfeiffer. 2008. Integrated Accessibility Models of User Interfaces for IT and Automation Systems. In *Proceedings of the ISCA 21st International Conference on Computer Applications in Industry and Engineering, CAINE 2008*, Frederick C. Harris Jr. (Ed.). ISCA, 280–285. https://dblp.org/rec/conf/caine/GohnerKJVP08.bib

[43] María González-García, Lourdes Moreno, and Paloma Martínez. 2014. Adaptation Rules for Accessible Media Player Interface. In *XV International Conference on Human Computer Interaction* (Puerto de la Cruz, Tenerife, Spain) *(Interacción '14)*. ACM, Article 5, 8 pages. https://doi.org/10.1145/2662253.2662258

[44] María González-García, Lourdes Moreno, Paloma Martínez, Raúl Miñon, and Julio Abascal. 2013. A Model-Based Graphical Editor to Design Accessible Media Players. *Journal of Universal Computer Science* 19, 18 (2013), 2656–2676. https://doi.org/10.3217/jucs-019-18-2656

[45] Google. 2025. Google Scholar. https://scholar.google.com.au/

[46] Christian Granrath, Christopher Kugler, Judith Michael, Bernhard Rumpe, and Louis Wachtmeister. 2025. Generating Logical Architectures from SysML Behavior Models. *INCOSE Systems Engineering* (2025). https://doi.org/10.1002/sys.70005

[47] 1EdTech Accessibility Project Group. 2004. 1EdTech Guidelines for Developing Accessible Learning Applications. https://www.imsglobal.org/accessibility/accessiblevers/index.html

[48] Henning Heitkötter, Tim A. Majchrzak, and Herbert Kuchen. 2013. Cross-Platform Model-Driven Development of Mobile Applications with Md2. In *28th Annual ACM Symposium on Applied Computing (SAC '13)*. ACM, 526–533. https://doi.org/10.1145/2480362.2480464

[49] John Hutchinson, Jon Whittle, and Mark Rouncefield. 2014. Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. *Science of Computer Programming* 89 (2014), 144–161. https://doi.org/10.1016/j.scico.2013.03.017 Special

issue on Success Stories in Model Driven Engineering.

[50] Apple Inc. 2025. Accessibility. https://www.apple.com/accessibility/

[51] Sabina Jeschke and Helmut Vieritz. 2007. Accessibility and Model-Based Web Application Development for eLearning Environments. In *Innovations in E-learning, Instruction Technology, Assessment, and Engineering Education*, Magued Iskander (Ed.). Springer Netherlands, Dordrecht, 439–444. https://doi.org/10.1007/978-1-4020-6262-9_76

[52] Sabina Jeschke, Helmut Vieritz, and Olivier Pfeiffer. 2008. Developing Accessible Applications with User-Centered Architecture. In *Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)*. 684–689. https://doi.org/10.1109/ICIS.2008.117

[53] Hendrik Kausch, Judith Michael, Mathias Pfeiffer, Deni Raco, Bernhard Rumpe, and Andreas Schweiger. 2021. Model-Based Development and Logical AI for Secure and Safe Avionics Systems: A Verification Framework for SysML Behavior Specifications. In *Aerospace Europe Conference 2021 (AEC 2021)*. Council of European Aerospace Societies (CEAS).

[54] Akif Khan and Shah Khusro. 2019. Blind-friendly user interfaces – a pilot study on improving the accessibility of touchscreen interfaces. *Multimedia Tools and Applications* 78, 13 (2019), 17495–17519. https://doi.org/10.1007/s11042-018-7094-y

[55] Kamran Khowaja, Dena Al-Thani, Aboubakr Aqle, and Bilikis Banire. 2019. Accessibility or Usability of the User Interfaces for Visually Impaired Users? A Comparative Study. In *Universal Access in Human-Computer Interaction. Theory, Methods and Tools*, Margherita Antona and Constantine Stephanidis (Eds.). Springer International Publishing, Cham, 268–283.

[56] Barbara Kitchenham, Lech Madeyski, and David Budgen. 2023. SEGRESS: Software Engineering Guidelines for REporting Secondary Studies. *IEEE Transactions on Software Engineering* 49, 3 (2023), 1273–1298. https://doi.org/10.1109/TSE.2022.3174092

[57] Heiko Koziolek, Andreas Burger, Marie Platenius-Mohr, Julius Rückert, Hadil Abukwaik, Raoul Jetley, and PP Abdulla. 2020. Rule-based Code Generation in Industrial Automation: Four Large-scale Case Studies applying the CAYENNE Method. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. 152–161.

[58] Elmar Krainz, Johannes Feiner, and Martin Fruhmann. 2016. Accelerated Development for Accessible Apps – Model Driven Development of Transportation Apps for Visually Impaired People. In *Human-Centered and Error-Resilient Systems Development*, Cristian Bogdan, Jan Gulliksen, Stefan Sauer, Peter Forbrig, Marco Winckler, Chris Johnson, Philippe Palanque, Regina Bernhaupt, and Filip Kis (Eds.). Springer International Publishing, Cham, 374–381. https://doi.org/10.1007/978-3-319-44902-9_25

[59] Elmar Krainz and Klaus Miesenberger. 2017. Accapto, a generic design and development toolkit for accessible mobile apps. In *Harnessing the Power of Technology to Improve Lives*. IOS Press, 660–664. https://doi.org/10.3233/978-1-61499-798-6-660

[60] Elmar Krainz, Klaus Miesenberger, and Johannes Feiner. 2018. Can We Improve App Accessibility with Advanced Development Methods?. In *Computers Helping People with Special Needs*, Klaus Miesenberger and Georgios Kouroupetroglou (Eds.). Springer International Publishing, Cham, 64–70. https://doi.org/10.1007/978-3-319-94277-3_12

[61] Lila F. Laux, Peter R. McNally, Michael G. Paciello, and Gregg C. Vanderheiden. 1996. Designing the World Wide Web for People with Disabilities: A User Centered Design Approach. In *Second Annual ACM Conference on Assistive Technologies (Assets '96)*. Association for Computing Machinery, New York, NY, USA, 94–101. https://doi.org/10.1145/228347.228363

[62] Legislative Assembly of Ontario. 2008-2025. Accessibility for Ontarians with Disabilities Act (AODA). https://www.aoda.ca/

[63] Google LLC. 2024. Accessibility & Material Design. https://developer.android.com/guide/topics/ui/accessibility/apps

[64] Kelly Mack, Emma McDonnell, Dhruv Jain, Lucy Lu Wang, Jon E. Froehlich, and Leah Findlater. 2021. What Do We Mean by "Accessibility Research"? A Literature Survey of Accessibility Papers in CHI and ASSETS from 1994 to 2019. In *2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. ACM, New York, USA, Article 371. https://doi.org/10.1145/3411764.3445412

[65] Adriana Martín, Gustavo Rossi, Alejandra Cechich, and Silvia Gordillo. 2010. Engineering Accessible Web Applications. An Aspect-Oriented Approach. *World Wide Web* 13, 4 (2010), 419–440. https://doi.org/10.1007/s11280-010-0091-3

[66] Susel María Matos Claro. 2025. Generation of User Interfaces for Different Context of Use from Conceptual Models. In *Intelligent Information Systems*, Luise Pufahl, Kristina Rosenthal, Sergio España, and Selmin Nurcan (Eds.). Springer Nature Switzerland, Cham, 305–312.

[67] Raúl Miñón, Lourdes Moreno, and Julio Abascal. 2013. A Graphical Tool to Create User Interface Models for Ubiquitous Interaction Satisfying Accessibility Requirements. *Univers. Access Inf. Soc.* 12, 4 (nov 2013), 427–439. https://doi.org/10.1007/s10209-012-0284-x

[68] Judith Michael, Lukas Netz, Bernhard Rumpe, Ingo Müller, John Grundy, Shavindra Wickramathilaka, and Hourieh Khalajzadeh. 2026. MDE and Visual Impairments: Complementary Repository. https://github.com/judithmichael/MDE-and-Visual-Impairments/

[69] Judith Michael and Volodymyr Shekhovtsov. 2024. A Model-Based Reference Architecture for Complex Assistive Systems and its Application. *Journal Software and Systems Modeling (SoSyM)* 23, 5 (October 2024), 1247–1274. https://doi.org/10.1007/s10270-024-01157-1

[70] Raúl Miñón, Lourdes Moreno, Paloma Martínez, and Julio Abascal. 2014. An approach to the integration of accessibility requirements into a user interface development method. *Science of Computer Programming* 86 (2014), 58–73. https://doi.org/10.1016/j.scico.2013.04.005 Special issue on Software Support for User Interface Description Languages (UIDL 2011).

[71] Raúl Miñón, Fabio Paternò, Myriam Arrue, and Julio Abascal. 2016. Integrating adaptation rules for people with special needs in model-based UI development process. *Universal Access in the Information Society* 15, 1 (2016), 153–168. https://doi.org/10.1007/s10209-015-0406-3

[72] Parastoo Mohagheghi, Wasif Gilani, Alin Stefanescu, Miguel A. Fernandez, Bjørn Nordmoen, and Mathias Fritzsche. 2013. Where does model-driven engineering help? Experiences from three industrial cases. *Software & Systems Modeling* 12, 3 (2013), 619–639. https://doi.org/10.1007/s10270-011-0219-7

[73] Government of France. 2016. General Accessibility Framework for Administrations (RGAA). https://disic.github.io/rgaa_referentiel_en/rgaa-companion-guide.html

[74] Karla Ordoñez, José Hilera, and Samanta Cueva. 2022. Model-Driven Development of Accessible Software: A Systematic Literature Review. *Universal Access in the Information Society* 21, 1 (2022), 295–324. https://doi.org/10.1007/s10209-020-00751-6

[75] Karla Ordoñez-Briceño, José R Hilera, Luis de Marcos, Salvador Otón-Tortosa, and Samanta Cueva-Carrión. 2024. UML profile to model accessible web pages. *IEEE Access* 12 (2024), 77181–77213.

[76] Débora Maria Barroso Paiva, André Pimenta Freire, and Renata Pontin de Mattos Fortes. 2021. Accessibility and Software Engineering Processes: A Systematic Literature Review. *Journal of Systems and Software* 171 (2021), 110819. https://doi.org/10.1016/j.jss.2020.110819

[77] Laiba Rana, Attique Ur Rehman, Sabeen Javaid, and Tahir Muhammad Ali. 2022. A Novel Model-Driven Approach for Visual Impaired People Assistance OPTIC ALLY. In *2022 Third International Conference on Latest trends in Electrical Engineering and Computing Technologies (INTELLECT)*. 1–8. https://doi.org/10.1109/INTELLECT55495.2022.9969400

[78] Christoph Rieger, Daniel Lucrédio, Renata Pontin M. Fortes, Herbert Kuchen, Felipe Dias, and Lianna Duarte. 2020. *A Model-Driven Approach to Cross-Platform Development of Accessible Business Apps*. Association for Computing Machinery, New York, NY, USA, 984–993. https://doi.org/10.1145/3341105.3375765

[79] Karlis Rokis and Marite Kirikova. 2022. Challenges of Low-Code/No-Code Software Development: A Literature Review. In *Perspectives in Business Informatics Research*, Ērika Nazaruka, Kurt Sandkuhl, and Ulf Seigerroth (Eds.). Springer International Publishing, Cham, 3–17. https://doi.org/10.1007/978-3-031-16947-2_1

[80] Matthew J. Rutherford and Alexander L. Wolf. 2003. A Case for Test-Code Generation in Model-Driven Systems. In *Generative Programming and Component Engineering*, Frank Pfenning and Yannis Smaragdakis (Eds.). Springer, 377–396.

[81] Bran Selic. 2025. The Digital Human Twin – A Human-Centric Extension of the Digital Twin Idiom. In *Int. Conf. on Engineering Digital Twins (EDTconf'25)*. IEEE.

[82] Ian Sommerville. 2018. *Software Engineering* . Pearson.

[83] David Steinberg, Frank Budinsky, Marcelo Paternostro, and Ed Merks. 2008. *EMF: Eclipse Modeling Framework 2.0* (2nd ed.). Addison-Wesley Professional. https://www.oreilly.com/library/view/emf-eclipse-modeling/9780321331885/

[84] Pedro Teixeira, Celeste Eusébio, and Leonor Teixeira. 2024. Understanding the integration of accessibility requirements in the development process of information systems: a systematic literature review. *Requirements Engineering* 29, 2 (2024), 143–176. https://doi.org/10.1007/s00766-023-00409-8

[85] UIML. 2020. OASIS User Interface Markup Language. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uiml

[86] US Consumer and Governmental Affairs. 2021. 21st Century Communications and Video Accessibility Act (CVAA). https://www.fcc.gov/consumers/guides/21st-century-communications-and-video-accessibility-act-cvaa

[87] UsiXML. 2011. User Interface Extensible Markup Language. http://www.usixml.org/

[88] Kris Van Hees and Jan Engelen. 2013. Equivalent representations of multimodal user interfaces: Runtime Reification of Abstract User Interface Descriptions. *Universal Access in the Information Society* 12 (2013), 339–368. Issue 4. https://doi.org/10.1007/s10209-012-0282-z

[89] Helmut Vieritz, Olivier Pfeiffer, and Sabina Jeschke. 2007. BELEARNING: Designing accessible eLearning applications. In *2007 37th Annual Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports*. S3D–1–S3D–6. https://doi.org/10.1109/FIE.2007.4417985

[90] Helmut Vieritz, Daniel Schilberg, and Sabina Jeschke. 2013. Early Accessibility Evaluation in Web Application Development. In *Universal Access in Human-Computer Interaction. User and Context Diversity*, Constantine Stephanidis and Margherita Antona (Eds.). Springer, 726–733. https://doi.org/10.1007/978-3-642-39191-0_78

[91] Helmut Vieritz, Farzan Yazdi, Daniel Schilberg, Peter Göhner, and Sabina Jeschke. 2011. User-Centered Design of Accessible Web and Automation Systems. In *Information Quality in e-Health*, Andreas Holzinger and Klaus-Martin Simonic (Eds.). Springer, Berlin, Heidelberg, 367–378. https://doi.org/10.1007/978-3-642-25364-5_26

[92] Birgit Vogel-Heuser, Daniel Schütz, Timo Frank, and Christoph Legat. 2014. Model-driven engineering of Manufacturing Automation Software Projects – A SysML-based approach. *Mechatronics* 24, 7 (2014), 883–897. https://doi.org/10.1016/j.mechatronics.2014.05.003 1. Model-Based Mechatronic System Design 2. Model Based Engineering.

[93] Markus Völter, Thomas Stahl, Jorn Bettin, Arno Haase, and Simon Helsen. 2013. *Model-Driven Software Development: Technology, Engineering, Management*. Wiley. https://www.wiley.com/en-sg/Model+Driven+Software+Development%3A+Technology%2C+Engineering%2C+Management-p-9781118725764

[94] W3C. 2008. Web Content Accessibility Guidelines (WCAG) 2.0. https://www.w3.org/TR/WCAG20/

[95] W3C. 2015. Authoring Tool Accessibility Guidelines (ATAG) 2.0. https://www.w3.org/TR/ATAG20/

[96] W3C. 2015. User Agent Accessibility Guidelines (UAAG) 2.0. https://www.w3.org/TR/UAAG20/

[97] W3C. 2023. Accessible Rich Internet Applications (WAI-ARIA) 1.2. https://www.w3.org/TR/wai-aria-1.2/

[98] W3C. 2023. Web Content Accessibility Guidelines (WCAG) 2.2. https://www.w3.org/TR/WCAG22/

[99] W3C. 2024. Web Content Accessibility Guidelines (WCAG) 2.1. https://www.w3.org/TR/WCAG21/

[100] W3C. 2025. Guidance on Applying WCAG 2.2 to Mobile (WCAG2Mobile). https://w3c.github.io/matf/

[101] Willian Massami Watanabe, David Fernandes Neto, Thiago Jabur Bittar, and Renata P. M. Fortes. 2010. WCAG Conformance Approach Based on Model-Driven Development and WebML. In *28th ACM International Conference on Design of Communication* (São Carlos, São Paulo, Brazil)

(SIGDOC '10). ACM, 167–174. https://doi.org/10.1145/1878450.1878479

[102] Shavindra Wickramathilaka, John Grundy, Kashumi Madampe, and Omar Haggag. 2025. Adaptive and accessible user interfaces for seniors through model-driven engineering. *Automated Software Engineering* 32, 2 (2025), 74. https://doi.org/10.1007/s10515-025-00547-z

[103] Claes Wohlin. 2014. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. Association for Computing Machinery, New York, NY, USA, Article 38. https://doi.org/10.1145/2601248.2601268

[104] Xtend. 2015. Introduction. https://eclipse.dev/Xtext/xtend/documentation/index.html

[105] Eclipse Xtext. 2023. Eclipse Xtext. https://projects.eclipse.org/projects/modeling.tmf.xtext

[106] Farzan Yazdi, Helmut Vieritz, Nasser Jazdi, Daniel Schilberg, Peter Göhner, and Sabina Jeschke. 2011. A Concept for User-Centered Development of Accessible User Interfaces for Industrial Automation Systems and Web Applications. In *Universal Access in Human-Computer Interaction. Applications and Services*, Constantine Stephanidis (Ed.). Springer, 301–310. https://doi.org/10.1007/978-3-642-21657-2_32

[107] Lamia Zouhaier, Yousra Bendaly Hlaoui, and Leila Jemni Ben Ayed. 2014. A MDA-based Approach for Enabling Accessibility Adaptation of User Interface for Disabled People. In *Proceedings of the 16th International Conference on Enterprise Information Systems - Volume 2: ICEIS*. SciTePress, 120–127. https://doi.org/10.5220/0004897901200127

A SELECTED PRIMARY STUDIES

Table 14. Overview of selected primary studies

ID	Authors	Title	Year
S1	Rana et al.	A Novel Model-Driven Approach for Visual Impaired People Assistance OPTIC ALLY [77]	2022
S2	Dias et al.	AccessMDD: An MDD Approach for Generating Accessible Mobile Applications [30]	2021
S3	Rieger et al.	A Model-Driven Approach to Cross-Platform Development of Accessible Business Apps [78]	2020
S4	Bendaly Hlaoui et al.	Model driven approach for adapting user interfaces to the context of accessibility: case of visually impaired users [13]	2019
S5	Bouraoui and Gharbi	Model driven engineering of accessible and multi-platform graphical user interfaces by parameterized model transformations [17]	2019
S6	Khan and Khruso	Blind-friendly user interfaces–a pilot study on improving the accessibility of touchscreen interfaces [54]	2019
S7	Krainz et al.	Can we improve app accessibility with advanced development methods? [60]	2018
S8	Andrade et al.	Incorporating accessibility elements to the software engineering process [7]	2018
S9	Krainz and Miesenberger	Accapto, a generic design and development toolkit for accessible mobile apps [59]	2017
S10	Krainz et al.	Accelerated development for accessible apps – model driven development of transportation apps for visually impaired people [58]	2016
S11	Minon et al.	Integrating adaptation rules for people with special needs in model-based UI development process [71]	2016
S12	Gamecho et al.	Automatic generation of tailored accessible user interfaces for ubiquitous services [40]	2015
S13	González-García et al.	Adaptation Rules for Accessible Media Player Interface [43]	2014
S14	Miñón et al.	An approach to the integration of accessibility requirements into a user interface development method [70]	2014
S15	Zouhaier et al.	A MDA-based Approach for Enabling Accessibility Adaptation of User Interface for Disabled People [107]	2014
S16	González-García et al.	A Model-Based Graphical Editor to Design Accessible Media Players [44]	2013
S17	van Hees and Engelen	Equivalent representations of multimodal user interfaces: Runtime Reification of Abstract User Interface Descriptions [88]	2013
S18	Vieritz et al.	Early accessibility evaluation in web application development [90]	2013
S19	Minon et al.	A graphical tool to create user interface models for ubiquitous interaction satisfying accessibility requirements [67]	2013
S20	Vieritz et al.	User-centered design of accessible web and automation systems [91]	2011
S21	Yazdi et al.	A concept for user-centered development of accessible user interfaces for industrial automation systems and web applications [106]	2011
S22	Abascal et al.	Automatically generating tailored accessible user interfaces for ubiquitous services [1]	2011
S23	Bittar et al.	Accessible Organizational Elements in Wikis with Model-Driven Development [14]	2010
S24	Watanabe et al.	WCAG Conformance Approach Based on Model-Driven Development and WebML [101]	2010
S25	Martin et al.	Engineering accessible Web applications. An aspect-oriented approach [65]	2010
S26	Göhner et al.	Integrated accessibility models of user interfaces for IT and automation systems [42]	2008

S27	Jeschke et al.	Developing Accessible Applications with User-Centered Architecture [52]	2008
S28	Bouraoui et al.	E-learning and handicap: New trends for accessibility with model driven approach [18]	2007
S29	Vieritz et al.	BeLearning: Accessibility in Virtual Knowledge Spaces [89]	2007
S30	Jeschke and Vieritz	Accessibility and model-based web application development for elearning environments [51]	2007

B RESEARCH METHODS

Additional detailed information on selected elements of our research process, as defined in Section 3.

Table 15. Database search parameters

Database	**Query**	**Search options**	**Scope / SUBJECT AREA**
ACM DL	(accessibility OR blind* OR "low vision" OR "vision impair*" OR "visual* impair*") AND (MDE OR "model-driven")	advanced search, full-text collection	abstract, author keywords title / -
IEEE Xplore	accessibility OR blind* OR "low vision" OR "vision impair*" OR "visual* impair*" AND MDE OR "model-driven"	advanced search	abstract, author keywords, document title / -
Science Direct	(accessibility OR blind OR "low vision" OR "vision impair" OR "visual impair") AND (MDE OR "model-driven")	advanced search, show all fields	'title, abstract or author-specified keywords' / COMPUTER SCIENCE
Scopus	(accessibility OR blind* OR {low vision} OR "vision impair*" OR "visual* impair*") AND (MDE OR "model-driven")	document search	'article title, abstract, keywords' / COMPUTER SCIENCE

Table 16. Overview of our data extraction points

Metadata	
M1	Publication venue and year
M2	Number of citations
M3	Paper type
M4	Problem type
M5	Contribution type
M6	Author names and affiliations
Accessibility	
DP 1.1	Addressed visual impairments
DP 1.2	Target audience in focus
DP 1.3	Used accessibility standards and guidelines
Approach and results	

DP 2.1	Technical details of the proposed approach
DP 2.2	Used MDE frameworks, techniques, models, etc
DP 2.3	Modeled application aspects
DP 2.4	Achieved level of MDE process automation
DP 2.5	Type of software system in focus
Evaluation	
DP 3.1	Used evaluation method
DP 3.2	Achieved level of accessibility
Limitations and open challenges	
DP 4.1	Reported pros, cons, limitations and open challenges
DP 4.2	Addressed research questions
DP 4.3	Addressed gaps and open challenges in the MDE domain

Table 17. Quality assessment results

ID	Authors	Title	Year	QA1	QA2	QA3	QA4	Score
S1	Rana et al.	A Novel Model-Driven Approach for Visual Impaired People Assistance OPTIC ALLY [77]	2022	Y	P	P	N	2/4
S2	Dias et al.	AccessMDD: An MDD Approach for Generating Accessible Mobile Applications [30]	2021	Y	Y	P	N	2.5/4
S3	Rieger et al.	A Model-Driven Approach to Cross-Platform Development of Accessible Business Apps [78]	2020	P	Y	P	N	2/4
S4	Bendaly Hlaoui et al.	Model driven approach for adapting user interfaces to the context of accessibility: case of visually impaired users [13]	2019	Y	Y	P	Y	3.5/4
S5	Bouraoui and Gharbi	Model driven engineering of accessible and multi-platform graphical user interfaces by parameterized model transformations [17]	2019	Y	Y	P	N	2.5/4
S6	Khan and Khruso	Blind-friendly user interfaces – a pilot study on improving the accessibility of touchscreen interfaces [54]	2019	Y	P	P	N	2/4
S7	Krainz et al.	Can we improve app accessibility with advanced development methods? [60]	2018	N	P	N	N	0.5/4
S8	Andrade et al.	Incorporating accessibility elements to the software engineering process [7]	2018	P	Y	N	Y	2.5/4
S9	Krainz and Miesenberger	Accapto, a generic design and development toolkit for accessible mobile apps [59]	2017	P	P	N	N	1/4
S10	Krainz et al.	Accelerated development for accessible apps – model driven development of transportation apps for visually impaired people [58]	2016	Y	P	N	N	1.5/4
S11	Minon et al.	Integrating adaptation rules for people with special needs in model-based UI development process [71]	2016	N	Y	P	N	1.5/4

S12	Gamecho et al.	Automatic generation of tailored accessible user interfaces for ubiquitous services [40]	2015	P	P	P	Y	2.5/4
S13	González-García et al.	Adaptation Rules for Accessible Media Player Interface [43]	2014	Y	P	P	N	2/4
S14	Miñón et al.	An approach to the integration of accessibility requirements into a user interface development method [70]	2014	N	Y	P	N	1.5/4
S15	Zouhaier et al.	A MDA-based Approach for Enabling Accessibility Adaptation of User Interface for Disabled People [107]	2014	P	P	P	N	1.5/4
S16	González-García et al.	A Model-Based Graphical Editor to Design Accessible Media Players [44]	2013	N	P	P	N	1/4
S17	van Hees and Engelen	Equivalent representations of multimodal user interfaces: Runtime Reification of Abstract User Interface Descriptions [88]	2013	Y	Y	P	N	2.5/4
S18	Vieritz et al.	Early accessibility evaluation in web application development [90]	2013	N	P	N	N	0.5/4
S19	Minon et al.	A graphical tool to create user interface models for ubiquitous interaction satisfying accessibility requirements [67]	2013	N	P	P	N	1/4
S20	Vieritz et al.	User-centered design of accessible web and automation systems [91]	2011	P	P	N	N	1/4
S21	Yazdi et al.	A concept for user-centered development of accessible user interfaces for industrial automation systems and web applications [106]	2011	N	P	N	N	0.5/4
S22	Abascal et al.	Automatically generating tailored accessible user interfaces for ubiquitous services [1]	2011	N	P	N	Y	1.5/4
S23	Bittar et al.	Accessible Organizational Elements in Wikis with Model-Driven Development [14]	2010	N	P	P	N	1/4
S24	Watanabe et al.	WCAG Conformance Approach Based on Model-Driven Development and WebML [101]	2010	N	P	N	N	0.5/4
S25	Martin et al.	Engineering accessible Web applications. An aspect-oriented approach [65]	2010	N	P	P	N	1/4
S26	Göhner et al.	Integrated accessibility models of user interfaces for IT and automation systems [42]	2008	Y	P	N	N	1.5/4
S27	Jeschke et al.	Developing Accessible Applications with User-Centered Architecture [52]	2008	P	P	P	N	1.5/4
S28	Bouraoui et al.	E-learning and handicap: New trends for accessibility with model driven approach [18]	2007	N	P	P	N	1/4
S29	Vieritz et al.	BeLearning: Accessibility in Virtual Knowledge Spaces [89]	2007	N	P	N	N	0.5/4
S30	Jeschke and Vieritz	Accessibility and model-based web application development for elearning environments [51]	2007	N	P	N	N	0.5/4

C RESULTS

Additional detailed information on selected results, as presented in Section 4.

Table 18. Overview of publication venues

Publication venue	**Number of studies**
Journals	
Journal on Universal Access in the Information Society	3
Journal on Science of Computer Programming	2
IEEE Transactions on Human-Machine Systems	1
Journal of Multimedia Tools and Applications	1
Journal of Universal Computer Science	1
Journal on Advances in Human-Computer Interaction	1
Journal on Internet and Web Information Systems	1
Journal on Multimodal User Interfaces	1
Journal on Studies in Health Technology and Informatics	1
Conferences	
ACM International Conference on Design of Communication (SIGDOC)	3
Innovations in E-learning, Instruction Technology, Assessment, and Engineering Education (ICTA)	2
International Conference on Universal Access in Human-Computer Interaction (UAHCI)	2
Annual ACM Symposium on Applied Computing (SAC)	1
ASEE/IEEE Frontiers in Education Conference	1
IEEE/ACIS International Conference on Computer and Information Science (ICIS)	1
International ACM SIGACCESS conference on Computers and accessibility (Assets)	1
International Conference on Computer Applications in Industry and Engineering (CAINE)	1
International Conference on Computers Helping People with Special Needs (ICCHP)	1
International Conference on Enterprise Information Systems (ICEIS)	1
International Conference on Human-Centred Software Engineering (HCSE), and International Working Conference on Human Error, Safety, and System Development (HESSDE)	1
International Conference on Human Computer Interaction (INTERACCIÓN)	1
International Conference on Latest Trends in Electrical Engineering and Computing Technologies (INTELLECT)	1
Symposium of the Austrian HCI and Usability Engineering Group	1
total number of studies	30

Table 19. Publications per author team (studies with authors from multiple teams in italic and bold font)

Author team affiliation	**Authored studies**	**Number of studies**
University of the Basque Country, Donostia, Spain	***S11,*** S12, ***S14, S16, S19,*** S22	6
University of Stuttgart, Germany	***S20, S21, S26, S27, S29***	5
ICMC-University of Sao Paulo, Sao Carlos, Brazil	S2, ***S3, S23,*** S24	4

Technical University Berlin, Germany	***S26, S27, S29,*** S30	4
Universidad Carlos III de Madrid, Spain	S13, ***S14, S16, S19***	4
FH Joanneum Kapfenberg, Austria	***S7, S9,*** S10	3
RWTH Aachen, Germany	S18, ***S20, S21***	3
Johannes Kepler University Linz, Austria	***S7, S9***	2
Laboratory LaTICE, University of Tunis, Tunisia	S4, S15	2
Ecole Supérieure des Sciences et Techniques de Tunis, Tunisia	S28	1
Federal University of Mato Grosso do Sul (UFMS), Brazil	S8	1
GULF University for Science and Technology, Mubarak Al-Abdullah, Kuwait	***S1***	1
ISTI-CNR, Pisa, Italy	***S11***	1
Katholieke Universiteit Leuven, Belgium	S17	1
UFPE Recife, Brazil	***S23***	1
UFSCar, Sao Carlos, Brazil	***S3***	1
Universidad Nacional de La Plata and Conicet, La Plata, Argentina	***S25***	1
Universidad Nacional del Comahue, Buenos Aires, Argentina	***S25***	1
University of Manouba, Tunesia	***S5***	1
University of Muenster, Germany	***S3***	1
University of Peshawar, Peshawar, Pakistan	S6	1
University of Sialkot, Pakistan	***S1***	1
University of Tunis El Manar, Tunesia	***S5***	1