[BKM+25] D. Bork, S. Klikovitz, J. Michael, L. Netz, B. Rumpe:
Inclusive Model-Driven Engineering for Accessible Software.
In: 2025 ACM/IEEE 28th International Conference on Model Driven Engineering Languages and Systems (MODELS),
pp. 253-259, DOI 10.1109/MODELS67397.2025.00030, IEEE, Oct. 2025.

Inclusive Model-Driven Engineering for Accessible Software

Dominik Bork*, Stefan Klikovits†, Judith Michael‡, Lukas Netz§, Bernhard Rumpe§
*Business Informatics Group, TU Wien, Vienna, Austria, Email: dominik.bork@tuwien.ac.at
† Department of Business Informatics, JKU Linz, Linz, Austria, Email: stefan.klikovits@jku.at
‡Programming and Software Engineering, University of Regensburg, Regensburg, Germany, Email: judith.michael@ur.de
§Software Engineering, RWTH Aachen University, Aachen, Germany, Email: netz@se-rwth.de, rumpe@se-rwth.de

***Abstract*—While model-driven engineering (MDE) claims to be a good development approach for cross-cutting concerns, this raises the question of why not every application created with MDE is accessible. Moreover, why are the MDE development processes and the tools themselves not more inclusive? In this vision paper, we sketch an ideal picture of how inclusivity in MDE — considered throughout tools, methods, artifacts, and processes — would intrinsically lead to more accessible software systems. We review the state-of-the-art, discuss current challenges, and present a possible future of inclusive MDE.**

***Index Terms*—Model-driven engineering, Software engineering, Inclusion, Accessibility**

I. Introduction

*"**Accessibility** is about making sure that barriers that may prevent people with disabilities from taking part are removed. **Inclusion** is about going a step further and ensuring that people with disabilities are included as valuable members in all aspects of society. This includes things like listening to views and opinions and allowing people with disabilities to contribute to planning, decisions and their futures."* [1]

Similarly to the overall societal and academic ambitions toward becoming more inclusive, there is a growing need for achieving inclusion in software engineering [2], especially in model-driven engineering (MDE) where often domain-specific solutions (languages, methods, and tools) are being used that lack—compared to general software engineering with powerful players and platforms—the economic possibilities and human resources to realize inclusivity in the development processes and accessibility in the resulting software.

As society strives for a more *'inclusive world'*, with this vision paper, we aim to shed light on the diverse needs of MDE developers [3]–[5] wishing to be involved in the software development and the users for whom we are developing software for [6]–[8]. Consequently, we outline a vision in which the future MDE processes, including all used languages, methods, and tools, are inclusive, and, through such an extended involvement of diverse stakeholders, the resulting software intrinsically becomes more accessible [9], [10].

One important aspect is the inclusion of heterogeneous software user needs. We can find ongoing discussions and research in various areas, e.g., AI and diversity [11], bias in data science [12], and diversity in development teams [13], [14]. Traditionally, accessibility is often considered late in the development cycle, leading to late or costly adaptations of systems or exclusionary designs. A survey of existing MDE languages, techniques, and tools that support the development of accessible software is proposed in [15]. Notably, the authors conclude that *"... many of the studies focus on making the final software product accessible, without considering the accessibility of previously designed models ..."*.

Inclusion is relevant not only for the technologies we create as software engineers, but also the processes that lead to the development of these technologies need to be inclusive to support developers with heterogeneous needs [16], [17]. Having more diversity in development teams leads to more diversity in technical solutions and more equitable technology [17], [18] as development teams adopt accessibility practices in software development. In the literature, we can find approaches that discuss accessibility in software engineering processes [6], [19], how to reduce barriers for people with disabilities to become employed in software engineering [20], or the need to still increase the accessibility of the tools used [21].

In this vision paper, we focus on MDE as a development method, and analyze how it can become more inclusive toward eventually facilitating the engineering of accessible software. Liebel et al. suggest how to consider human factors in MDE in general [22], e.g., the use of models and the design of modeling languages. However, their work has no focus on inclusion or accessibility. Our work zooms further in: We consider the different steps in the MDE process, the different roles (e.g., modeler, software developer, language engineer, software user), and relevant artefacts and tools.

II. The vision

In our vision, software engineers treat inclusion and accessibility as first-class citizens in MDE, leading towards more inclusive software engineering processes and more accessible systems. This covers two main areas: creating inclusive environments for developers within MDE processes and engineering software that is accessible for end users.

Developer Inclusivity. Embedding inclusivity into MDE practices and tools supports the participation of developers with diverse backgrounds and disabilities in the MDE process, e.g., by providing accessible modeling tools and platforms. Such approaches promote diversity, equity, involvement, and belonging within the software engineering community.

979-8-3315-4910-7/25/$31.00 ©2025 IEEE
DOI 10.1109/MODELS67397.2025.00030

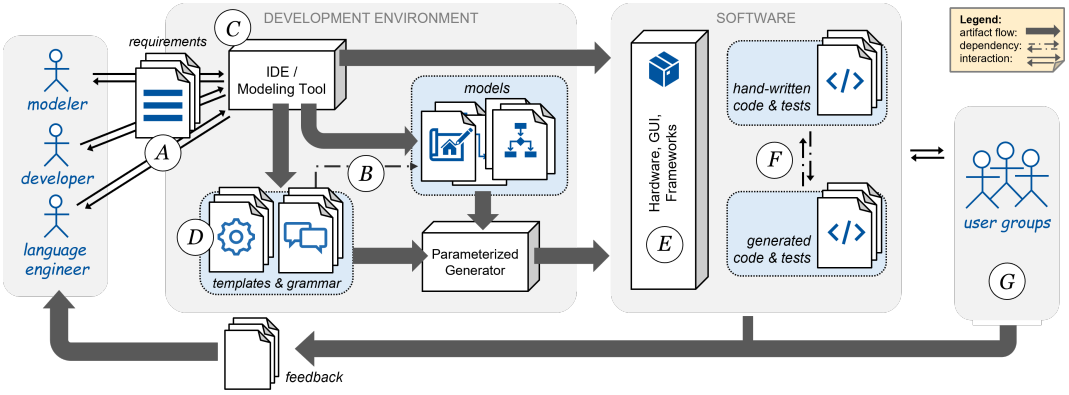

Fig. 1: Inclusive Model-driven Software Engineering

Software Accessibility. By integrating accessibility concerns into core modeling artifacts, e.g., requirements models, user interface models, and behavioral specifications, engineers can systematically address diverse user needs. This shift enables the generation of accessible software by construction and fosters a more inclusive design mindset.

Together, these approaches promote equity for end-users and within the software engineering community. For realizing our vision, we highlight, based on the state-of-the-art, the challenges and potential future in the subsequent sections.

III. MDE Developer Inclusivity

A core principle of MDE is the acknowledgement that domain experts and practitioners should be more actively involved [23] and obtain the capacity to design, modify, and update the software according to their needs. Yet, despite decades of development in the field of MDE, the current state of MDE software could be described as "inadequate". Mainstream IDEs (e.g., Eclipse, VSCode, PyCharm) increasingly provide support for visually impaired developers in the form of high-contrast modes, zoom, font size flexibility, partial support for screen readers, or shortcuts/global command execution. Although such features represent incremental progress, they often do not support the full workflow of visually or motor-impaired developers across the entire MDE process (cf. Fig. 1), and still lack many accessibility features [24]. VSCode recently added a VoiceSupport-plugin, which enables speech-based command and menu interaction [25] for keyboard- and mouse-less interaction. Nonetheless, most IDEs and modeling platforms still lack a systematic approach to inclusive and accessible development processes [26].

A. MDE Editor and Artefacts Accessibility

MDE editor and artefact (e.g., models, templates, grammars, code) inclusivity (see Fig. 1) is an important prerequisite for technical inclusion that needs to be considered in a twofold manner: 1. *artefact-external* (e.g., adapting MDE tools to offer inclusive artefact interactions), and 2. *artefact-internal*, where inclusivity is built into the artefact itself using, e.g., syntactic/semantic constructs. These approaches are complementary and can be viewed as two sides of the same coin.

As an example, consider graphical modeling languages where positioning typically carries implicit meaning, and elements that are logically and/or semantically related are placed nearby. On the one hand, model editors could be enhanced to calculate and articulate (relative) element distances (e.g., for visually impaired people). On the other hand, the model itself could be enhanced to contain descriptive knowledge using explicit properties (e.g., as logical groupings, related elements, etc.), instead of mere x/y coordinates. This duality—supporting perceptual interfaces while enriching internal model semantics—is crucial toward realizing inclusive MDE practices and, eventually, accessible software.

B1. State-of-the-Art, Challenges, and Vision

Clearly, many of the techniques and requirements for end-user software (see Sec. IV) equally apply to developer tools. For instance, color contrast analysis, keyboard operability, and multimodal outputs benefit both software users and those developing the software itself. Nonetheless, it is evident that, given its frequent use of visual and/or graph-based modeling languages, inclusivity and accessibility in the MDE domain require dedicated attention. We review related work that could be applied to foster a more inclusive MDE landscape, structured along the WCAG's core principles for web content, i.e., *perceivable*, *operable*, *understandable*, and *robust* [27].

1) Perceivable: Inclusive MDE tools must enable all users to perceive the model information (Ⓒ in Fig. 1), regardless of sensory abilities. Visual modeling languages, however, use shapes, colors, and spatial layouts to convey relationships, creating barriers for users with visual impairments, including color blindness. In the past, several alternative perception techniques have been explored, in addition to screen readers: Sonification/audification translates visual information to auditory cues. Existing works on the audification of graphs [28], diagrams [29], [30], algebra [31], [32], and molecules [33] could be applied to models to enable users to "hear" their models. Tactile output devices [34] offer yet another dimension of model perception, such as UML diagrams [35]. Hybrid DSLs [36] offer live switching between graphical and textual model representations.This has been used for UML

diagrams [37] and brainstorming models ('mind maps') [38] for visually impaired people.

Challenges & Vision: Current MDE tools and techniques target able-bodied developers. We develop our software with intuitive graphical representations, icons, and relations that can be distinguished by color, shape, and positioning. Similarly, textual editors often primarily rely on text highlighting of (sometimes even unintuitive) keywords. This requires a) more research into accessible forms of model perception, including truly comprehensible tactile model representations, and audification/sonification of various forms of data, and b) research into hybrid, configurable languages that treat all syntax versions as first-class citizens on an industrial scale.

2) Operable: The second WCAG core principle caters to the interaction with models and the use of modeling tools [39]. This means that alternatives to the traditional mouse-and-keyboard interaction paradigm must be provided. An evident approach is the support of touch interfaces [40] and styluses, which allow the dragging and pointing [41], and *drawing* of diagrams [42]. Alternatively, modern AI techniques allow the parsing of actual drawings into diagrams [43].

Orthogonally, natural language processing (NLP) and AI (in the form of, e.g., LLMs) were explored to enable novel specification capabilities. Recently, these approaches were extended to support the generation of UML diagrams [44]–[46], Ecore models [47] as a basis for DSLs from natural language, and web applications from natural languages [48].

Challenges & Vision: The primary challenge is the support for alternative forms of interaction and devices, including touch, tactile, keyboard, natural language, and speech commands for a broad set of popular and domain-specific languages. This includes enhancing editors/IDEs to translate various forms of model editing/interaction commands, as well as designing models that are built for accessible editors.

3) Understandable: Third, the understandability of models, modeling languages, and tools must be tackled. As of now, modeling typically requires expert knowledge and/or formal training [49]. Automated explanations might help novice developers to comprehend the intent behind complex model structures, while automatically generated summaries can help onboard new colleagues. Mussbacher et al. [5], e.g., investigated opportunities in intelligent modeling assistance.

Early approaches such as OntoVerbal [50] investigated the verbalization of ontologies in Protegé, while other techniques helped users without ontology-expertise navigate medical ontologies [51]. Alternatively, [52] showed the description of UML models in natural language.

Orthogonally, we know that most modeling and programming languages require some understanding of English, which may also pose difficulties to non-English speakers [53].

Challenges & Vision: The understandability of models and tools can be split into various challenges. From a linguistic viewpoint, the internationalization of DSLs and models must be extended beyond the mere translation of keywords, but support true polyglot interactions. This includes various aspects, including model syntax, editing commands, and training materials, that all need to support a truly native-language interaction. As for comprehension, we argue for the inclusion of descriptions straight into the models themselves, alongside automatically generated explanations and conversational/chatbot-like model interaction possibilities. AI and LLMs, for instance, appear to be ideal candidates to include such functionality, as recently introduced by tools such as NotebookLM [54].

4) Robust: Robustness denotes the ambition to increase compatibility, such that models can be shared across tool vendors. While MDE as a domain has in the past gathered around popular formats such as XMI and Ecore, the community's current move toward web-based platforms results in a landscape of heterogeneous and complementary tools for which interoperability and reuse become crucial.

Challenges & Vision: As widespread standards enable tool vendors to build upon, it is also evident that this standardization must not hinder innovation. Tool builders often require custom extensions, which should be developed in a traceable, shareable, and extensible way. Balancing the diverging ambitions of standardization and innovation is the main challenge to robustness. We argue that we have to learn from the web development community, where consortia are formed to compromise on opposing views to establish a common foundation upon which creativity and innovation can unfold.

B2. Inclusive Methods & Processes

Concerning inclusivity, we must not forget the processes in which the developers engage in MDE. To realize a truly inclusive MDE, not only the used tools but also the development methods need to become inclusive to remove barriers and enable participation of developers with heterogeneous needs [16], [17]. Notably, this vision goes beyond gender balance but acknowledges, among others, social, racial, ethnic, and underrepresented groups. Furthermore, we need to strive for an actively positive environment for neurodivergent modelers. Research showed that such an increase in developer diversity leads to greater diversity in technical solutions, increased creativity, and more equitable technology [17], [18].

Challenges & Vision: Active participation of developers with heterogeneous backgrounds raises the challenge of following a streamlined development process. Naturally, the more heterogeneous a group of humans is, the more challenging it is to find norms for structuring communication and collaboration [55]. In collaborative software engineering, there needs to be a means of open communication, critical feedback, peer review, and deadline-driven delivery. Such an environment, often characterized also by stress and pressure to deliver the software in adequate quality at a pre-defined time, establishes challenges for e.g., introverted developers or neurodiverse developers coping with ADHD, autism, and dyslexia.

IV. MDE of Accessible Software

Software engineering research proposes ways to integrate accessibility in the software development process, as this is a reliable way to obtain accessible software [6]. However, it is important to see the creation of accessible systems not as a

limitation on creativity and design of software [56], but crucial within the entire MDE process (cf. A, B, D-G in Fig. 1). In the following, we describe related works regarding the accessibility support in the MDE process. We further discuss remaining challenges toward realizing our vision.

(A) *Requirements.* To consider accessibility already in the requirements engineering phase, there exist different guidelines, e.g., WCAG [27], [57], ISO 9241-171 [58], Accessibility development documentation for Android [59] and a guide for iOS [60], as well as the Material Design guideline for accessibility [59]. We can find approaches that consider these specifications in the requirements engineering phase, e.g., [61]–[64], however, they apply the mappings from the guidelines to requirements in a manual way.

Challenges & Vision: An automated mechanism to help map concrete requirements to accessibility guidelines is needed. In the future, we envision the provision of reusable models and domain-specific languages (DSLs) that describe relevant aspects from accessibility guidelines to foster and automate their incorporation in MDE processes.

(B) *Models and Grammar.* Existing approaches use different modeling languages and techniques to model various aspects related to accessibility coming from these guidelines, e.g., the user interface [65]–[67], interaction with the software [68]–[70], navigation in the software [71]–[73], capabilities of users [74]–[76], accessibility requirements [77], [78], context information [75], [79], or adaptation rules to consider accessibility needs [80], [81]. However, it is hard to learn from existing approaches and transfer them to other MDE projects as the descriptions are often on a high abstraction level and the developed models are not made available as reusable artifacts.

Challenges & Vision: While accessibility is strongly related to user interfaces, there is a notable lack of research on explicitly modeling accessibility requirements and their traceable mapping to concrete system functionality, as well as modeling context information and user capabilities. Moreover, accessibility in general is hardly achievable, as different user groups have different and even conflicting requirements. Thus, we need to develop modeling and generative methods that could handle sets of requirements variants for different user groups. Furthermore, research should investigate if existing modeling languages are sufficient to cover all aspects relevant for considering different kinds of accessibility needs, or if language extensions or even newly developed domain-specific languages are needed to describe accessibility requirements and software adaptation options. In addition, this work has to be documented in a reusable way for other developers.

(D) *Transformation Rules and Templates.* There are only a few approaches that use adapted templates to accommodate individual accessibility needs [73]. Some approaches [66], [80], [82] implement transformation rules to adapt the corresponding models, resulting in a custom-accessible software.

Challenges & Vision: Enabling reusability in MDE remains a challenge because many approaches rely on high-level descriptions without providing replication packages, making it difficult for others to reproduce or build upon existing work. Future work should place greater emphasis on developing generic transformation rules and templates that are more reusable and adaptable, enabling their integration into arbitrary model-driven engineering projects.

(E) *Hardware, GUI & Frameworks.* Other works focus specifically on user interfaces, e.g., Abrahão et al. [83] summarize work on model-based user interface adaptation. Bouraoui and Gharbi [84] propose a method for generating accessible UIs. Gerasimov et al. [85] propose a GUI component library to better handle accessibility needs. Few studies focus on the impact of device usage in MDE for GUI generation [75] and mainly focus on mobile devices.

Challenges & Vision: Runtime environment components are often tightly coupled to specific platforms and frameworks, limiting their flexibility to address diverse accessibility requirements in a generic manner. Future research should focus on improving the adaptability of selected components, for example, by using component libraries or generic APIs, to better accommodate individual accessibility needs. Model-driven approaches should also be leveraged to develop accessible software for a wider variety of devices.

(F) *Code and Functional Tests.* Software that is developed in generative software engineering environments often contains both generated and handwritten code. MDE approaches can be used not only to generate software, but also to generate the corresponding test environments and tests [86], [87], however, these are often mostly functional tests that are not optimized to ensure accessibility or inclusion guidelines. In addition, there are several frameworks that can be used to evaluate the accessibility of web applications [88].

Challenges & Vision: There is a need for methodologies and MDE approaches that focus on semantic tests [89] for specific user needs. This ensures compliance with requirements while remaining accessible for targeted user groups. Upcoming MDE research should include accessibility and inclusion-focused tests within the generated test frameworks.

(G) *Evaluation with users.* The W3C WAI highlights the importance of including users in evaluating accessibility [90] besides checking the conformance to accessibility standards. One has to evaluate the software with the respective user groups, e.g., with users with visual impairments [74], blind users [75], [91], or users with cognitive disabilities [92] to avoid creating superficial or marginally useful solutions [93]. This involvement can range from brief consultations in informal settings to large-scale usability studies in formal settings.

Challenges & Vision: Getting representatives from target user groups to evaluate the created software is not easy, and such an evaluation is not part of many publications presenting research on accessible software systems [94]. While this applies to software created with MDE just as much as to those developed conventionally, MDE research should work on developing methods to include relevant user groups and their feedback throughout the MDE process.

V. The future of inclusive MDE

Next to web technologies and AI, MDE is one of the core techniques to enable inclusive software design and push for accessible software. Given the widespread use of model-driven design in industry, our community can create a lasting impact that tears down current barriers.

To reach this goal, the MDE community must realize that the status quo offers much room for improvement. To address the challenges stressed at the outset, we need a holistic perspective that accounts for all parts of the MDE process through inclusive design from the inception. This means that our *tools* need to be created with accessibility-by-design, such that developers, regardless of their abilities, can fully participate. For physically impaired users, this means the creation of keyboard-first modeling processes, while in parallel enabling voice-driven commands and speech interfaces. Graphical editors have to be accessible and should enable touch, draw, and also game controller-like input devices, similar to tactile outputs. We further envision the design of WCAG-compliant modeling editors that support standardized interfaces for accessibility tools and assistive technology, which could be driven through inclusion into standards such as LSP/GLSP. Additionally, past studies have shown a (disproportionately) high fraction of neurodiverse developers in tech and software engineering [95], which includes, i.e., colleagues with ADHD, autism, and dyslexia. Our tools, methods, and processes must be designed in such a way that enables developers with these and similar characteristics a distraction-minimizing development, that enables them to fully engage during the entire MDE process.

Models and modeling languages should be extended and designed in such a way that switching between graphical and textual syntax (i.e., blended modeling) is possible, while at the same time natively enabling semantically accessible annotations and language extensions such as element grouping in spatial layouts and descriptive metadata. Languages with graphical concrete syntax should focus on perceptive redundancy (element identification through shape, color, and text), while keywords and element names should be chosen with speech capabilities in mind.

To create more accessible software, the MDE community can rely on many helpful techniques: We can either generate software that is adaptable for users to tailor it to their specific needs, or we can generate different software variants that address different accessibility needs. Other options are self-adaptive systems that configure themselves according to particular user profiles that specific user groups carry. E.g., Savidis et al. [96] stress the need for new interaction metaphors, manipulating diverse collections of interaction objects, automatic interface adaptation, and ambient interactions. One possible solution can be provided through GenAI, serving as a mediator to facilitate individually customized user interactions. Users with diverse individual needs can collaboratively develop the same software through inclusive AI agents, which transform varying input modalities into unified software models [48], [97]. Similarly, software can be generated directly to support standards such as MCP [98], thereby natively enabling agent-based, inclusive interactions with the generated software. Users could simply 'tell' the software what to do and ask questions using their preferred input modality.

When moving to the model representation, models no longer need to be represented solely in predefined syntaxes, such as textual artifacts or simple box-and-line diagrams. In the future, inclusive MDE should generate customized representations involving icons, storyboards, or video sequences [99], [100] to inclusively, semantically transparent, and comprehensibly illustrate models in the most appropriate formalism for specific user groups and purposes. Here, the adoption of AR/VR techniques for MDE can realize inclusion [101], [102].

From a research community point of view, we must establish an environment that promotes inclusion. Dedicated tracks and workshops at MDE conferences can be used as platforms for experience reports and exchange of ideas, while at the same time raising awareness. Moreover, our courses and teaching materials should teach accessibility and inclusive practices as a core principle. Furthermore, modeling tool tracks could include accessibility evaluations in their tool assessment

VI. Conclusion

In this paper, we outlined our vision of inclusive model-driven engineering as a foundation for developing accessible software. Following the stages of the MDE process, we examined the current state of inclusion and accessibility, identifying key limitations and gaps in existing tools and practices. Building on this analysis, we proposed a future direction for inclusive MDE–one that embeds accessibility as a first-class concern and supports the collaboration of both abled and disabled engineers in the realisation of accessible software systems. This paper aims to spark innovation and creativity, fostering a more open, inclusive, and equal MDE landscape.

References

[1] Sign Solutions, "What's the difference between accessibility and inclusion?" https://www.signsolutions.uk.com/whats-the-difference-between-accessibility-and-inclusion/, 2022.

[2] P. Teixeira, C. Eusébio, and L. Teixeira, "Understanding the integration of accessibility requirements in the development process of information systems: a systematic literature review," *Requirements Engineering*, vol. 29, no. 2, pp. 143–176, 2024.

[3] P. Calais and L. Franzini, "Test-Driven Development Benefits beyond Design Quality: Flow State and Developer Experience," in *45th Int. Conf. on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER '23. IEEE Press, 2023, p. 106–111.

[4] A. Bouraffa and W. Maalej, "Two Decades of Empirical Research on Developers' Information Needs: A Preliminary Analysis," in *Int. Conf. on Software Engineering Workshops*, 2020, p. 71–77.

[5] G. Mussbacher *et al.*, "Opportunities in intelligent modeling assistance," *Software & Systems Modeling*, vol. 19, no. 5, 2020.

[6] D. M. B. Paiva, A. P. Freire, and R. P. de Mattos Fortes, "Accessibility and Software Engineering Processes: A Systematic Literature Review," *Journal of Systems and Software*, vol. 171, p. 110819, 2021.

[7] M. Brhel, H. Meth, A. Maedche, and K. Werder, "Exploring principles of user-centered agile software development: A literature review," *Information and Software Technology*, vol. 61, pp. 163–181, 2015.

[8] L. L. Constantine and L. A. D. Lockwood, "Usage-centered software engineering: an agile approach to integrating users, user interfaces, and usability into software engineering practice," in *25th Int. Conf. on Software Engineering*, ser. ICSE '03. IEEE, 2003, p. 746–747.

[9] D. E. Damian, K. Blincoe, D. Ford, A. Serebrenik, and Z. Masood, Eds., *Equity, Diversity, and Inclusion in Software Engineering: Best Practices and Insights*. Apress, 2024.

[10] G. Rodríguez-Pérez, R. Nadri, and M. Nagappan, "Perceived diversity in software engineering: a systematic literature review," *Empir. Softw. Eng.*, vol. 26, no. 5, p. 102, 2021.

[11] P. Daugherty, H. Wilson, and R. Chowdhury, "Using Artificial Intelligence to Promote Diversity," *MIT Sloan Management Review*, 2018.

[12] M. O. Prates, P. H. Avelar, and L. C. Lamb, "Assessing gender bias in machine translation: a case study with google translate," *Neural Computing and Applications*, vol. 32, pp. 6363–6381, 2020.

[13] G. A. A. Prana, D. Ford, A. Rastogi, D. Lo, R. Purandare, and N. Nagappan, "Including Everyone, Everywhere: Understanding Opportunities and Challenges of Geographic Gender-Inclusion in OSS," *IEEE Trans. Software Eng.*, vol. 48, no. 9, pp. 3394–3409, 2022.

[14] W. Hussain, H. Perera, J. Whittle, A. Nurwidyantoro, R. Hoda, R. A. Shams, and G. C. Oliver, "Human Values in Software Engineering: Contrasting Case Studies of Practice," *IEEE Trans. Software Eng.*, vol. 48, no. 5, pp. 1818–1833, 2022.

[15] K. Ordoñez, J. Hilera, and S. Cueva, "Model-driven development of accessible software: a systematic literature review," *Universal Access in the Information Society*, vol. 21, no. 1, pp. 295–324, 2022.

[16] E. Dagan, A. Sarma, A. Chang, S. D'Angelo, J. Dicker, and E. Murphy-Hill, "Building and Sustaining Ethnically, Racially, and Gender Diverse Software Engineering Teams: A Study at Google," in *31st ACM Joint Europ. Software Engineering Conf. and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2023. ACM, 2023.

[17] I. Cardoso-Pereira, G. Gomes, D. M. Ribeiro, A. d. Souza, D. Lucena, and G. Pinto, "Supporting the Careers of Developers With Disabilities: Lessons From Zup Innovation," *IEEE Software*, vol. 40, no. 5, 2023.

[18] S. M. Hyrynsalmi, S. Baltes, C. Brown, R. Prikladnicki, G. Rodriguez-Perez, A. Serebrenik, J. Simmonds, B. Trinkenreich, Y. Wang, and G. Liebel, "Making Software Development More Diverse and Inclusive: Key Themes, Challenges, and Future Directions," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 5, 2025.

[19] N. Rajh, K. Miesenberger, and R. Koutny, "Accessibility in the Software Engineering (SE) Process and in Integrated Development Environments (IDEs): A Systematic Literature Review," in *Computers Helping People with Special Needs, ICCHP'24*. Springer, 2024.

[20] T. Rocha, N. Davila, R. Vaccari, N. Menezes, M. Mota, E. Monteiro, C. R. B. de Souza, and G. Pinto, "Affirmative Hackathon for Software Developers with Disabilities: An Industry Initiative," in *Cooperative and Human Aspects of Software Engineering*, 2025, pp. 39–50.

[21] T. A. da Rocha, C. de Souza, L. Teran, and M. Mota, "Effective Inclusion of People with Disabilities in Software Development Teams," in *18th ACM/IEEE Int. Symp. on Empirical Software Engineering and Measurement*, ser. ESEM '24. ACM, 2024, p. 447–453.

[22] G. Liebel *et al.*, "Human factors in model-driven engineering: future research goals and initiatives for mde," *Software & Systems Modeling*, vol. 23, no. 4, pp. 801–819, 2024.

[23] G. Fischer, S. Lindstaedt, J. Ostwald, M. Stolze, T. Sumner, and B. Zimmermann, "From domain modeling to collaborative domain construction," in *1st Conf. on Designing interactive systems: processes, practices, methods, & techniques*, 1995, pp. 75–85.

[24] K. Albusays *et al.*, "Interviews and Observation of Blind Software Developers at Work to Understand Code Navigation Challenges," in *19th Int. Conf. on Computers and Accessibility*. ACM, 2017.

[25] Microsoft Inc., "VS Code Speech ," https://code.visualstudio.com/docs/configure/accessibility/voice, 2025.

[26] A. Sarioglu, H. Metin, and D. Bork, "Accessibility in conceptual modeling - A systematic literature review, a keyboard-only UML modeling tool, and a research roadmap," *Data Knowl. Eng.*, vol. 158, 2025.

[27] W3C, "Web content accessibility guidelines (wcag) 2.2," https://www.w3.org/TR/WCAG22/, 2023.

[28] A. Brown, R. Stevens, and S. Pettifer, "Audio representation of graphs: A quick look," in *Int. Conf. on Auditory Displays*. Citeseer, 2006.

[29] A. R. Kennel, "Audiograf: A diagram-reader for the blind," in *2nd annual ACM Conf. on Assistive technologies*, 1996, pp. 51–56.

[30] T. Murillo-Morales and K. Miesenberger, "AUDiaL: A Natural Language Interface to Make Statistical Charts Accessible to Blind Persons," in *17th Int. Conf. Computers Helping People with Special Needs(ICCHP), P. I*, ser. LNCS, vol. 12376. Springer, 2020.

[31] R. Stevens, S. Brewster, P. Wright, A. Edwards, and G. Kramer, "Design and Evaluation of an Auditory Glance at Algebra for Blind Readers," in *2nd Int. Conf. on Auditory Display*. Addison-Wesley, 1994.

[32] R. Stevens, A. Edwards, G. Allen, J. Wilkinson, and P. Wright, "Strategy and prosody in listening to algebra," in *Adjunct Proceedings of HCI'95: People and Computers*. British Computer Society, 1995.

[33] A. Brown, S. Pettifer, and R. Stevens, "Evaluation of a non-visual molecule browser," *ACM SIGACCESS Accessibility And Computing*, no. 77-78, pp. 40–47, 2003.

[34] J. C. Bliss, M. H. Katcher, C. H. Rogers, and R. P. Shepard, "Optical-to-Tactile Image Conversion for the Blind," *IEEE Transactions on Man-Machine Systems*, vol. 11, no. 1, pp. 58–65, 1970.

[35] C. Loitsch and G. Weber, "Viable haptic UML for blind people," in *Int. Conf. on Computers for Handicapped Persons*, 2012, pp. 509–516.

[36] I. Predoaia, D. Kolovos, and A. Garcia-Dominguez, "Hybrid Graphical-Textual DSL Editors: Vision, Requirements and Challenges," in *ACM/IEEE 27th Int. Conf. on Model Driven Engineering Languages and Systems*, ser. MODELS Companion '24. ACM, 2024.

[37] A. King, P. Blenkhorn, D. Crombie, S. Dijkstra, G. Evans, and J. Wood, "Presenting UML software engineering diagrams to blind people," in *9th Int. Conf. Computers Helping People with Special Needs (ICCHP'04). Proc. 9*. Springer, 2004, pp. 522–529.

[38] S. Pölzer, D. Schnelle-Walka, D. Pöll, P. Heumader, and K. Miesenberger, "Making brainstorming meetings accessible for blind users," in *Assistive Technology: From Research to Practice*, 2013, pp. 653–658.

[39] D. Bork and G. D. Carlo, "An extended taxonomy of advanced information visualization and interaction in conceptual modeling," *Data Knowl. Eng.*, vol. 147, p. 102209, 2023.

[40] F. Wildhaber, N. Salloum, M. Gygli, and A. Kennel, "Self-Directed Creation and Editing of UML Class Diagrams on Mobile Devices for Visually Impaired People," in *IEEE 10th Int. Model-Driven Requirements Engineering (MoDRE)*, 2020, pp. 49–57.

[41] J. Stark, M. Burwitz, R. Braun, and W. Esswein, "Cognitive efficient modelling using tablets," in *Enterprise Modelling and Information Systems Architectures (EMISA 2013)*. GI, 2013, pp. 57–70.

[42] B. Tenbergen, R. Buck, and L. Lazarro, "Sketch UML: A Tablet PC-based e-Learning Tool for UML Syntax using a Minimalistic Interface," 2008.

[43] A. Conrardy and J. Cabot, "From image to UML: first results of image based UML diagram generation using LLMS," in *STAF 2024 Workshops - LLM4MDE*. CEUR, 2024.

[44] S. Yang and H. Sahraoui, "Towards automatically extracting UML class diagrams from natural language specifications," in *Model Driven Engineering Languages and Systems: Companion Proc.*, 2022.

[45] A. Ferrari, S. Abualhaija, and C. Arora, "Model Generation with LLMs: From Requirements to UML Sequence Diagrams," in *IEEE 32nd Int. Requirements Engineering Conf. Workshops (REW)*, 2024.

[46] D. K. Deeptimahanti and M. A. Babar, "An automated tool for generating UML models from natural language requirements," in *IEEE/ACM Int. Conf. on Automated Software Engineering*, 2009.

[47] A. Alaoui Mdaghri, M. Ouederni, and L. Chaari, "MDE in the Era of Generative AI," in *17th Int. Conf. Verification and Evaluation of Computer and Communication Systems*. Springer, 2025.

[48] L. Netz, J. Michael, and B. Rumpe, "From Natural Language to Web Applications: Using Large Language Models for Model-Driven Software Engineering," in *Modellierung 2024*, ser. LNI. GI, 2024.

[49] S. Friedenthal, "SysML: Lessons from early applications and future directions," *Insight*, vol. 12, no. 4, pp. 10–12, 2009.

[50] S. F. Liang, R. Stevens, D. Scott, and A. L. Rector, "OntoVerbal: a Protégé plugin for verbalising ontology classes," in *ICBO*, 2012.

[51] S. F. Liang, D. Scott, R. Stevens, and A. Rector, "Unlocking medical ontologies for non-ontology experts," in *BioNLP 2011 workshop*, 2011.

[52] F. Meziane, N. Athanasakis, and S. Ananiadou, "Generating natural language specifications from UML class diagrams," *Requirements Engineering*, vol. 13, pp. 1–18, 2008.

[53] B. A. Becker, "Parlez-vous Java? Bonjour La Monde != Hello World: Barriers to Programming Language Acquisition for Non-Native English Speakers," in *Annual WS of the Psychology of Prog. IG*, 2019.

[54] Alphabet Inc., "NotebookLM," https://notebooklm.google/, 2025.

[55] G. Márquez, M. Pacheco, H. Astudillo, C. Taramasco, and E. Calvo, "Inclusion of individuals with autism spectrum disorder in Software Engineering," *Inf. Softw. Technol.*, vol. 170, p. 107434, 2024.

[56] B. Regan, "Accessibility and design: a failure of the imagination," in *Int. Cross-Disciplinary Workshop on Web Accessibility (W4A)*, ser. W4A '04. ACM, 2004, p. 29–37.

[57] W3C, "Guidance on applying wcag 2.2 to mobile (wcag2mobile)," https://w3c.github.io/matf/, 2025.

[58] I. O. for Standardization, *Ergonomics of Human System Interaction - Part 171: Guidance on software accessibility*, iso 9241-171:2008 ed. International Organization for Standardization, 2008.

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

[60] Apple Inc., "Accessibility," https://www.apple.com/accessibility/, 2025.

[61] F. Dias, L. Duarte, and R. Fortes, "Accessmdd: An mdd approach for generating accessible mobile applications," in *39th ACM Int. Conf.on Design of Communication*, ser. SIGDOC '21. ACM, 2021, p. 85–95.

[62] C. Rieger, D. Lucrédio, R. P. M. Fortes, H. Kuchen, F. Dias, and L. Duarte, *A Model-Driven Approach to Cross-Platform Development of Accessible Business Apps*. ACM, 2020, p. 984–993.

[63] E. Krainz and K. Miesenberger, "Accapto, a generic design and development toolkit for accessible mobile apps," in *Harnessing the Power of Technology to Improve Lives*. IOS Press, 2017.

[64] A. Martín, G. Rossi, A. Cechich, and S. Gordillo, "Engineering accessible web applications. an aspect-oriented approach," *World Wide Web*, vol. 13, no. 4, pp. 419–440, 2010.

[65] L. Zouhaier, Y. B. Hlaoui, and L. J. B. Ayed, "A MDA-based Approach for Enabling Accessibility Adaptation of User Interface for Disabled People," in *16th Int. Conf. on Enterprise Information Systems (ICEIS), V.2*. SciTePress, 2014, pp. 120–127.

[66] R. Miñón, L. Moreno, and J. Abascal, "A graphical tool to create user interface models for ubiquitous interaction satisfying accessibility requirements," *Univers. Access Inf. Soc.*, vol. 12, no. 4, 2013.

[67] P. Göhner, S. Kunz, S. Jeschke, H. Vieritz, and O. Pfeiffer, "Integrated accessibility models of user interfaces for it and automation systems," in *21st Int. Conf.on Computer Applications in Industry and Engineering, CAINE'08*. ISCA, 2008, pp. 280–285.

[68] R. Miñón, F. Paternò, M. Arrue, and J. Abascal, "Integrating adaptation rules for people with special needs in model-based UI development process," *Universal Access in the Information Society*, 2016.

[69] M. González-García, L. Moreno, P. Martínez, R. Miñon, and J. Abascal, "A model-based graphical editor to design accessible media players," *Journal of Universal Computer Science*, vol. 19, no. 18, 2013.

[70] K. Van Hees and J. Engelen, "Equivalent representations of multimodal user interfaces: Runtime reification of abstract user interface descriptions," *Universal Access in the Information Society*, vol. 12, 2013.

[71] E. Krainz, J. Feiner, and M. Fruhmann, "Accelerated development for accessible apps – model driven development of transportation apps for visually impaired people," in *Human-Centered and Error-Resilient Systems Development*. Springer, 2016, pp. 374–381.

[72] T. J. Bittar, L. L. Lobato, R. P. M. Fortes, and D. F. Neto, "Accessible organizational elements in wikis with model-driven development," in *28th Int. Conf. on Design of Communication*. ACM, 2010.

[73] W. M. Watanabe, D. F. Neto, T. J. Bittar, and R. P. M. Fortes, "WCAG Conformance Approach Based on Model-Driven Development and WebML," in *28th ACM Int. Conf. on Design of Com.* ACM, 2010.

[74] Y. Bendaly Hlaoui, L. Zouhaier, and L. Ben Ayed, "Model driven approach for adapting user interfaces to the context of accessibility: case of visually impaired users," *Journal on Multimodal User Interfaces*, vol. 13, no. 4, pp. 293–320, 2019.

[75] A. Khan and S. Khusro, "Blind-friendly user interfaces – a pilot study on improving the accessibility of touchscreen interfaces," *Multimedia Tools and Applications*, vol. 78, no. 13, pp. 17 495–17 519, 2019.

[76] F. Yazdi, H. Vieritz *et al.*, "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*. Springer, 2011.

[77] W. T. Andrade, R. G. d. Branco, M. I. Cagnin, D. M. B. Paiva, and H. Nakanishi, "Incorporating accessibility elements to the software engineering process," *Advances in Human-Computer Interaction*, 2018.

[78] H. Vieritz, O. Pfeiffer, and S. Jeschke, "BELEARNING: Designing accessible eLearning applications," in *37th Annual Front. in Education Conf.*, 2007.

[79] S. Jeschke and H. Vieritz, "Accessibility and model-based web application development for elearning environments," in *Innovations in E-learning, Instruction Technology, Assessment, and Engineering Education*. Springer Netherlands, 2007, pp. 439–444.

[80] B. Gamecho, R. Miñón, A. Aizpurua, I. Cearreta, M. Arrue, N. Garay-Vitoria, and J. Abascal, "Automatic generation of tailored accessible user interfaces for ubiquitous services," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 5, pp. 612–623, 2015.

[81] M. González-García, L. Moreno, and P. Martínez, "Adaptation rules for accessible media player interface," in *XV Int. Conf.on Human Computer Interaction*, ser. Interacción '14. ACM, 2014.

[82] H. Vieritz, F. Yazdi, D. Schilberg, P. Göhner, and S. Jeschke, "User-centered design of accessible web and automation systems," in *Information Quality in e-Health*. Springer, 2011, pp. 367–378.

[83] S. Abrahão, E. Insfran, A. Sluÿters, and J. Vanderdonckt, "Model-based intelligent user interface adaptation: challenges and future directions," *Software & Systems Modeling*, vol. 20, no. 5, pp. 1335–1349, 2021.

[84] A. Bouraoui and I. Gharbi, "Model driven engineering of accessible and multi-platform graphical user interfaces by parameterized model transformations," *Science of Computer Programming*, vol. 172, 2019.

[85] A. Gerasimov, N. Jansen, J. Michael, B. Rumpe, and S. Will, "A model-driven approach to design, generation, and deployment of gui component libraries," in *18th ACM SIGPLAN Int. Conf. on Software Language Engineering*, 2025, pp. 57–70.

[86] B. Rumpe, *Agile Modeling with UML: Code Generation, Testing, Refactoring*. Springer International, May 2017.

[87] I. Drave, T. Greifenberg, S. Hillemacher, S. Kriebel, M. Markthaler, B. Rumpe, and A. Wortmann, "Model-Based Testing of Software-Based System Functions," in *Conf. on Software Engineering and Advanced Applications (SEAA)*, 2018, pp. 146–153.

[88] A. Alsaeedi, "Comparing Web Accessibility Evaluation Tools and Evaluating the Accessibility of Webpages: Proposed Frameworks," *Information*, vol. 11, no. 1, 2020.

[89] M. Bajammal and A. Mesbah, "Semantic web accessibility testing via hierarchical visual analysis," in *IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 1610–1621.

[90] W3C WAI, "Involving Users in Evaluating Web Accessibility," https://www.w3.org/WAI/test-evaluate/involving-users/, 2024.

[91] J. Abascal, A. Aizpurua, I. Cearreta, B. Gamecho, N. Garay-Vitoria, and R. Miñón, "Automatically Generating Tailored Accessible User Interfaces for Ubiquitous Services," in *13th Int. ACM SIGACCESS Conf. on Computers and Accessibility (ASSETS)*. ACM, 2011.

[92] L. Moreno, R. Alarcon, and P. Martínez, "Designing and Evaluating a User Interface for People with Cognitive Disabilities," in *21st Int. Conf. on Human Computer Interaction*. ACM, 2021.

[93] A. Lundgard, C. Lee, and A. Satyanarayan, "Sociotechnical considerations for accessible visualization design," in *IEEE Visualization Conf. (VIS)*, 2019, pp. 16–20.

[94] R. Angelini, K. Spiel, and M. de Meulder, "Speculating deaf tech: Reimagining technologies centering deaf people," in *2025 CHI Conference on Human Factors in Computing Systems*, 2025, pp. 66:1–66:18.

[95] A. El-Deeb, "Neurodiversity: What Can the Software Industry Gain from It?" *SIGSOFT Softw. Eng. Notes*, vol. 48, no. 4, p. 19, Oct. 2023.

[96] A. Savidis and C. Stephanidis, "Inclusive development: Software engineering requirements for universally accessible interactions," *Interact. Comput.*, vol. 18, no. 1, pp. 71–116, 2006.

[97] N. Baumann, J. S. Diaz, J. Michael, L. Netz, H. Nqiri, J. Reimer, and B. Rumpe, "Combining Retrieval-Augmented Generation and Few-Shot Learning for Model Synthesis of Uncommon DSLs," in *Modellierung 2024 - Workshopband*. GI, March 2024.

[98] Anthropic, "Model Context Protocol," https://modelcontextprotocol.io/, 2024, accessed: 2025-07-09.

[99] A. Gavric, D. Bork, and H. A. Proper, "How Does UML Look and Sound? Using AI to Interpret UML Diagrams Through Multimodal Evidence," in *Advances in Conceptual Modeling*, 2024, pp. 187–197.

[100] A. Gavric, D. Bork, and H. Proper, "Turning process models into videos," in *27th International Conference on Business Informatics (CBI)*, 2025. [Online]. Available: https://model-engineering.info/publications/papers/CBI25-Video-Generation.pdf

[101] R. Campos-López, E. Guerra, J. de Lara, A. Colantoni, and A. Garmendia, "Model-Driven Engineering for Augmented Reality," *J. Object Technol.*, vol. 22, no. 2, pp. 1–15, 2023.

[102] R. Reuter, F. Hauser, D. Muckelbauer, T. Stark, E. Antoni, J. Mottok, and C. Wolff, "Using Augmented Reality in Software Engineering Education? First insights to a comparative study of 2D and AR UML modeling," in *Hawaii Int. Conf. on System Sciences, HICSS 2019*, 2019.