TU WIEN Informatics

Accessibility in Web Modeling Tools: Systematic Review and Conceptualization of a Keyboard-Only Prototype

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Software Engineering und Internet Computing

eingereicht von

Aylin Sarioğlu, BSc
Matrikelnummer 11776853

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Assistant Prof. Dipl.-Wirtsch.Inf.Univ. Dr.rer.pol. Dominik Bork
Mitwirkung: Univ.Lektor Dipl.-Ing. Dr.techn. Philip Langer

Wien, 21. August 2023

Aylin Sarioğlu	Dominik Bork

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Accessibility in Web Modeling Tools: Systematic Review and Conceptualization of a Keyboard-Only Prototype

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Software Engineering and Internet Computing

by

Aylin Sarioğlu, BSc
Registration Number 11776853

to the Faculty of Informatics

at the TU Wien

Advisor: Assistant Prof. Dipl.-Wirtsch.Inf.Univ. Dr.rer.pol. Dominik Bork
Assistance: Univ.Lektor Dipl.-Ing. Dr.techn. Philip Langer

Vienna, 21st August, 2023

Aylin Sarioğlu	Dominik Bork

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Aylin Sarioğlu, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 21. August 2023

Aylin Sarioğlu

Kurzfassung

Die Zahlen der Weltgesundheitsorganisation zeigen auf, dass die Anzahl der Personen mit Behinderungen kontinuierlich zunimmt. Heutzutage ist das Internet ein grundlegender Bestandteil unseres alltäglichen Lebens. Aus diesem Grund ist es essenziell, dass alle Benutzerinnen und Benutzer unabhängig von ihren Fähigkeiten gleichen Zugang zu verschiedenen Informationen und Dienstleistungen haben. Trotz wachsender Bemühungen, weltweit Inklusion zu fördern, gibt es vor allem im Bereich der Informationssysteme noch immer Werkzeuge und Technologien, die nicht barrierefrei sind. Dies führt für viele Menschen mit Behinderungen zu einem ungleichen Zugang zu Informationssystemen.

Folglich sollte Barrierefreiheit auch ein wesentlicher Bestandteil sein, insbesondere im Bereich der konzeptionellen Modellierung, da vor allem Web-Modellierungswerkzeuge an Beliebtheit gewonnen haben. Diese Werkzeuge sind in verschiedenen Bereichen unverzichtbar und ermöglichen Fachleuten die Zusammenarbeit sowie die Erstellung und Entwicklung komplexer webbasierter Systeme. In diesem Zusammenhang ist die Barrierefreiheit von Web-Modellierungswerkzeugen von entscheidender Bedeutung, da sie sicherstellt, dass Nutzerinnen und Nutzer mit Behinderungen vollständig am Modellierungsprozess teilnehmen können und gleiche Chancen haben, davon zu profitieren.

Diese Arbeit präsentiert bestehende Barrieren und Hindernisse im Bereich der konzeptionellen Modellierungsforschung, indem sie aktuelle Forschungslücken identifiziert und eine Vision für eine inklusivere konzeptionelle Modellierung präsentiert, basierend auf einer systematischen Literaturübersicht und der Evaluierung aktuellen Modellierungswerkzeugen.

Eine hervorstehende Lücke existiert vor allem in der Forschung und der Barrierefreiheit von Modellierungswerkzeugen im Zusammenhang mit physischen Beeinträchtigungen. Basierend auf diesen Ergebnissen stellt diese Arbeit den ersten Modellierungsprototyp vor, der ausschließlich über die Tastatur bedient werden kann und so insbesondere Benutzerinnen und Benutzer mit körperlichen Behinderungen in die konzeptionelle Modellierung einbezieht und die Teilnahme daran ermöglicht.

Abstract

According to the World Health Organization, the number of disabilities is increasing steadily. Additionally, as the internet has become a fundamental aspect of our daily lives, it is essential to ensure that all individuals, regardless of their abilities, have equal access to different information and services. Despite a growing effort to promote inclusion worldwide, there are still inaccessible tools and technologies in the field of computation. This results in unequal access to information systems and products for many people with disabilities.

Consequently, accessibility should also be a key element, particularly in conceptual modeling, since web modeling tools have grown in popularity, as these tools have become essential in various fields, allowing professionals to work together, create, and build intricate web-based systems. In this regard, web accessibility is crucial for web modeling tools, as it ensures that users with disabilities can contribute fully to the modeling process and have equal chances to benefit.

This thesis presents existing web accessibility issues in conceptual modeling research by identifying current research gaps and delineating a vision toward more inclusive, i.e., disability-aware conceptual modeling, based on a systematic review of the literature and current modeling tools. One key finding relates to a gap in research and tool support concerning physical disabilities. Based on these results, this thesis presents the first modeling prototype that can be used keyboard-only, thereby including users with physical disabilities to engage in conceptual modeling.

Contents

Kurzfassung **vii**

Abstract **ix**

Contents **x**

1 Introduction **1**
1.1 Problem Statement and Motivation 1
1.2 Aim of the Work 2
1.3 Methodology 4
1.4 Structure of the Work 5

2 Literature Research and Systematic Review **7**
2.1 Research Method 7
2.2 Research Scope 9
2.3 Search process 10
2.4 Analysis & Synthesis of the selected publications 14
2.5 Summary & Interpration of the results 14

3 State-of-the-Art Research and Literature **25**
3.1 Introduction 25
3.2 Understanding & Advancing Web Accessibility 25
3.3 Inclusivity in Conceptual Modeling 31
3.4 Conclusion 37

4 Disability-Awareness in Web Modeling Tools **39**
4.1 Introduction to Web Barriers: Understanding the Challenges 39
4.2 Evaluation Framework 47
4.3 Results & Findings 70
4.4 Conclusion 89

5 Keyboard-Only Prototype & Evaluation **91**
5.1 Introduction 91
5.2 Theory & Background 92

5.3 Conception 98
5.4 Evaluation 114
5.5 Conclusion 117

6 Conclusion **119**
6.1 Summary 119
6.2 Future Work 120

7 Appendix **123**
Appendix 1: Selected Publications 124
Appendix 2: Commercial Software and Tools 127
Appendix 3: Additional support tools and extensions 128

List of Figures **129**

List of Tables **131**

Glossary **133**

Bibliography **135**

CHAPTER 1

Introduction

This chapter aims to give a deeper insight into the problem that this work will address. Firstly, this chapter introduces the domain of this work by giving a short insight into the Problem Statement and Motivation. This description will explain this work's aim and the applied methodology. Lastly, this chapter includes an overview of the remaining chapters of this thesis.

1.1 Problem Statement and Motivation

The advance of digitization has changed and shaped how we work and live, supporting or even replacing daily activities with digital services. As a result, digital media is an essential part of everyday life. However, it is not only technologies that are growing; the number of people with disabilities is also increasing along with the world population. Over the years, the World Health Organization (WHO) has published different reports on disability. The latest WHO article [1] states that about 16% of the world's population live with some form of disability. The reasons for this increase in disabled people include the ever more aging population and the rise in chronic health conditions. All this leads to the demand that digital accessibility must be unrestricted, as stated by Tim Berners Lee, the Director of World Wide Web Consortium (W3C) and inventor of the World Wide Web, in 1997 [2]:

> " The power of the Web is in its universality. Access by everyone, regardless of disability, is an essential aspect. "

[1] WHO - Disability, https://www.who.int/news-room/fact-sheets/detail/disability-and-health, (Access: 06.07.2023)

[2] W3C, https://www.w3.org/Press/IPO-announce, (Access: 31.07.2023)

Even though different measures have been taken, as reported by the European Commission [3], which did enhance Web accessibility and inclusion has improved but, there still exist people with disabilities who experience difficulties in their daily task on the web.

This is not surprising as studies show that this basic demand is not fulfilled to its finest. According to the 2023 report of Web Accessibility In Mind (WebAIM) on accessibility, around 96.3 % of the analyzed top 1.000.000 web pages contain WCAG 2 errors [Web23]. This checklist outlines suggestions for applying accessibility principles [Web21]. In addition, further developments in web engineering are increasing the complexity of web pages, which, consequently, makes the achievement of accessibility for everyone all the more complicated [PFdMF21, Web22a].

Even though there has been an increasing effort to ensure inclusiveness on the Web, more is needed, especially in conceptual modeling research. The software and web engineering community has already acknowledged the importance of accessible applications and, therefore, introduced different solutions, such as web accessibility guidelines or test frameworks, to make their application accessible to a wider audience. The situation is quite different in conceptual modeling research, where accessibility is so far essentially a unresearched area. This makes modeling less inclusive in terms of the diversity and heterogeneity of modelers and prevents many persons with disabilities from taking part in conceptual modeling.

This thesis contributes to increasing the awareness of the importance and relevance of accessibility and initiating conceptual modeling research about this topic. Furthermore, it aims to address the challenges faced by physically disabled users of web modeling tools. The concrete problem this work intends to solve is the heavy reliance on mouse-based interactions in web modeling tools, making it difficult for physically disabled users to engage in modeling fully. The objective is to develop a tool that provides full keyboard support, enabling physically disabled individuals to participate in the modeling process easily. The focus is on creating well-designed keyboard-only interaction possibilities that cater to the needs of users who may have difficulties or be unable to use pointing devices, complex keyboard shortcuts, or react quickly. The ultimate goal is to enable full engagement in modeling for physically disabled persons and improve accessibility to web modeling tools.

1.2 Aim of the Work

This thesis aims to highlight the importance of disability research in conceptual modeling and to reach more awareness about the barriers and problems that disabled persons encounter in the use of these kinds of tools. This goal will be achieved through the systematic literature review and tool evaluations. Identifying the hurdles and providing proposals for improving accessibility will be a valuable contribution to the community of

[3] European Commission, https://digital-strategy.ec.europa.eu/en/news/web-accessibility-and-inclusion-has-improved-some-people-disabilities-still-experience-difficulties, (Access: 07.07.2023)

disabled and impaired users. The specific target group of this thesis is especially humans with physical disabilities. This group will be supported with a keyboard-only prototype for a modeling editor, to support tool interactions only via keyboard. This prototype contribution will simplify the workflow for especially, physically disabled users who have had difficulties using current state-of-the-art tools. In addition, it provides access to modeling for all other disabled or impaired users who have not been able to use this kind of modeling tool until now due to the existing barriers.

In the **theoretical part**, this work identifies the current state of research and literature and reaches awareness for accessibility in the field of conceptual modeling by addressing the research gaps in this area.

The following results are provided in this thesis:

- A comprehensive **systematic literature review (SLR)** about web accessibility regarding the field of conceptual modeling. This step aims to gather information about the research state of this topic and to identify and highlight the current research gaps. Furthermore, valuable contributions and insights from the retrieved state-of-art research and literature are investigated in more detail to gain knowledge for further use in this work.
- A comprehensive **tool evaluation** on well-known web modeling tools regarding their disability support for different disability types. This analysis aims first to get an overview of the accessibility solutions that each tool provides. Secondly, this task is intended to show the problems and barriers that the discussed disability types can encounter when using these modeling tools.

In the **practical part** of the thesis, a keyboard-only prototype is implemented, especially focusing on the disability type, which is disregarded by research (and in the literature about it) so far. Due to the results of the theoretical part of the work and the found gaps, the focus of this prototype is on physical disabilities.

This contribution is valuable both for theoretical research and in practice, as it can be used for future research and already be used for modeling tasks. The expected result of this thesis will be the keyboard-only prototype, which extends the workflow diagram editor, based on the Graphical Language Server Platform (GLSP) [4]. GLSP is heavily used in industry and academia to realize web-based modeling tools with advanced visualization and interaction features. In addition to that, this prototype is not only part of the workflow diagram editor but its newly introduced features have also been generalized and released as part of GLSPs' open-source project, which is ready to be applied and used.

This keyboard-only prototype introduces new interaction possibilities, which contributes to making this modeling tool more accessible. The main goal of this prototype is

[4]Eclipse GLSP, `https://www.eclipse.org/glsp/`, (Access: 18.08.2023)

to provide especially physically disabled users with a more efficient and easy-to-use experience without needing a pointing device or complex shortcuts, enabling keyboard-only interactions for tasks used in modeling editors. These new interaction possibilities are based on the identified accessibility barriers and gaps in the theoretical part. (cf. [CLB22]).

1.3 Methodology

The methodological approach involves the following steps:

- **Literature Review**
  The current state of research and literature about (web) accessibility in conceptual modeling is explored and documented through a systematic literature review (SLR). In this context, research questions are specified and answered through a comprehensive process. The goal is to find relevant publications to gather knowledge about the previously mentioned topic.

- **Conceptualization**
  In order to form a foundation for the prototype, it is first necessary to gain knowledge about existing web modeling tools. For this purpose, an evaluation of existing web modeling tools has been conducted. This evaluation's aim was to find out which technical solutions for web accessibility are already available. It discovers which of the existing disability types would or would not be suitable and what these barriers are. For analyzing the tools, the criteria proposed by the Web Accessibility Initiative (WAI) [5] are used. While using the WAI criteria is a useful approach for analyzing the accessibility of web modeling tools, it is important to keep in mind that this method may not reveal some limitations and barriers. For example, the WAI criteria may not fully capture the subjective experiences and perspectives of physically disabled users and the challenges they face in their day-to-day interactions with these tools. Even testing with a representative group of users would not lead to a completely objective evaluation as the results would differ depending on the diversity of disability types and the unique requirements of different users. Therefore a solution in an absolute sense is not possible, as the subjective bias would be present again. However, these criteria provide a reasonable basis and summary of the most important aspects that should be considered concerning web accessibility for each of the given disability types, and in terms of this thesis specifically for physical disabilities.

  The aim of this analysis is to figure out which kind of accessibility features shall be implemented as a contribution to the GLSP platform. The criteria which are not fulfilled satisfactorily are selected and revised to plan and design new accessible

[5] WAI - Diverse Abilities and Barriers, https://www.w3.org/WAI/people-use-web/abilities-barriers/, (Access: 18.08.2023)

features for this prototype, e.g., missing or complex shortcuts, insufficient time limits, and many more.

- **Implementation & Evaluation**
  The keyboard-only prototype is developed based on the investigated gaps in research and the found barriers of the tool evaluation (based on the previously mentioned steps *Literature Review* and *Conceptualization*). It includes features that extend the existing modeling tools in GLSP to enable interactions specifically targeting physical disabilities. The focus of this prototype is to provide corresponding keyboard alternatives for the typical interactions of a modeling tool, such as the typical CRUD (create, read, update, delete), Model Exploration (e.g. Move, Zoom), and Model Navigation functionalities for elements, connectors, and other objects. This tool aims to provide users with a more efficient and user-friendly experience without needing a pointing device or complex shortcuts. This prototype enables interactions through only the keyboard for tasks like navigation, focusing, and searching, allowing physically disabled users to fully engage in the modeling process and enjoy the benefits of web modeling tools. Developing these new features required the creation of new concepts and design ideas. It is essential to take into account the unique challenges faced by physically disabled users, as well as their requirements for accessibility and ease of use. This involved exploring innovative solutions that would meet the unique needs of physically disabled users and ensure that the keyboard-only prototype is accessible, inclusive, and user-friendly.
  In conclusion, the keyboard-only prototype is analyzed and evaluated based on the same approach as the tool evaluation from section *Literature Review*.

1.4 Structure of the Work

This thesis is structured into the following five chapters:

Chapter 2 Systematic Literature Review This chapter describes the process and results of the conducted systematic literature review to gain insight into the current state of research regarding web accessibility in conceptual modeling.

Chapter 3 State-of-the-Art Research and Literature In this chapter, a detailed description and analysis can be found regarding the retrieved selected contributions, which align with the objectives of the search as described in the previous Chapter 2.

Chapter 4 Tool Evaluation This chapter provides a comprehensive description of the evaluation's preparation, process, and findings focused on ten well-known web modeling tools. The aim is to understand how state-of-the-art web modeling tools perform regarding their disability support.

Chapter 5 Prototyp & Evaluation This chapter provides information and further details about the keyboard-only prototype, which was implemented during this work to enhance accessibility regarding the physical disability type. A descriptive evaluation will also be given, addressing the strengths, weaknesses, and limitations.

Chapter 6 Conclusion This last chapter concludes this thesis with a summary and an overview of remaining issues and potential future work.

CHAPTER 2

Literature Research and Systematic Review

Accessibility is critical in inclusive design, and understanding the current research in this area is essential. This chapter provides a comprehensive overview of existing research and literature on web accessibility in the context of conceptual modeling. As outlined in the following sections, the systematic literature review aims to uncover the current research and work on accessibility issues in model engineering. The review guides the methodology for this thesis and contributes to existing knowledge in the field.

2.1 Research Method

The research method employed in this thesis is a systematic literature review (SLR). A *systematic literature review* is a structured process for identifying relevant literature from various sources on a specific topic, domain, or research questions. According to Kitchenham et al. [KBB+09], this research method typically consists of the following main steps: starting with defining the *research scope* or *questions*, planning and conducting the *search process*, analyzing and filtering the retrieved documents by *inclusion and exclusion criteria*, collecting the data and finally analyzing the data followed by a presentation of the results. This thesis follows the workflow mentioned above by Kitchenham et al. and adapts it to the requirements of this particular work. The process of the SLR is illustrated in Figure 2.1.

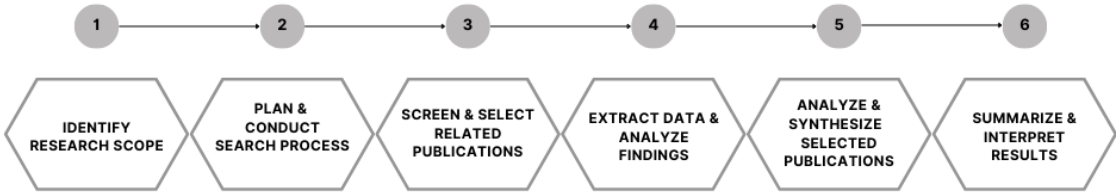

Figure 2.1: The steps of the systematic literature review applied in this thesis

This thesis research process consists of six steps and each can be summarized as follows:

- **Step 1: Identify the Research Scope**: This step is essential for preparing the search process. Its primary objective is to determine the research questions and precisely define the topic that will be the focus of the search.
- **Step 2: Plan and conduct the Search process**: In this stage, the methodology and sources for conducting a thorough search of relevant literature and publications will be established. Subsequently, it also includes the conduction of the search process.
- **Step 3: Screen and select related publications**: This step eliminates non-relevant findings by screening the retrieved publications and filtering them according to the predefined exclusion criteria (cf. Section 2.3.2).
- **Step 4: Extract data and analyze findings**: In this stage, the collected publications and literature will be analyzed and systematically organized. Studies are then categorized based on their characteristics and commonalities to uncover trends, themes, and common characteristics.
- **Step 5: Analysis & Synthesis of the selected publications**: This step thoroughly examines the selected publications for a more detailed and comprehensive understanding of the subject matter. This analysis enables a broader perspective on the current knowledge about the objective.
- **Step 6: Summary & Interpration of the results**: Lastly, the findings of the systematic literature review will be summarized and interpreted in light of the research questions and depending on the extracted data.

The steps mentioned above and their content are explained in detail in the upcoming sections. With the help of this systematic review, it is possible to obtain a comprehensive summary of existing literature about accessibility in conceptual modeling and find answers to the defined research questions.

2.2 Research Scope

In this phase, the primary outcome is to define the research questions and all the necessary tasks to address them through the literature review.

2.2.1 Preparation

This step is accomplished as a preparation for the initial search process. This phase aims to perform a rigorous keyword-based search to retrieve relevant documents or resources related to the particular topic of web accessibility and accessibility in conceptual modeling. The keywords used for this information retrieval process are listed in the paragraph below.

search keywords = (*accessibility* ∨ *web accessibility* ∨ *accessible model editor* ∨ *accessibility model engineering* ∨ *accessible diagram design* ∨ *accessible modeling tools* ∨ *accessible software* ∨ *accessibility software design*)

The keyword-based search is conducted mainly using Google Scholar. The search helped gain insight into existing documents, guidelines, and other subject-related terminology, which is used to create and improve the definition of a suitable search query (cf. Section 2.3.1). Furthermore, this additionally gained information helps to understand the content of the retrieved publications. The collected terminologies and information are listed below.

new terms = {*Web Content Accessibility Guideline (WCAG)*, *Web Accessibility Initiative (WAI)*, different *disability types* (cf. Table 2.3)}

2.2.2 Research Questions

This systematic literature review defines and addresses the following research questions.

- **RQ-1: What is the state of research and its evolution regarding web accessibility in the field of software engineering & information systems?** The aim is to gain a general overview of the research and work done in accessibility regarding software engineering and information systems over the years and to acquire basic knowledge about web accessibility.
- **RQ-2: What is the state of research and its evolution regarding web accessibility in the field of conceptual modeling?** This question aims to gain an overview of existing research and its evolution regarding web accessibility, particularly in conceptual modeling.
- **RQ-3: Which disabilities or impairments are covered in the existing literature?** This question aims to determine which disability and impairment types are predominantly discussed in the existing literature.

- **RQ-4: Which solutions are proposed to improve web accessibility?** This research question focuses on the purpose of existing software designs, solutions, or other technical or theoretical artifacts for improving (web) accessibility.
- **RQ-5: How effectively do current web modeling tools support individuals with *(i) visual*, *(ii) physical*, and *(iii) cognitive, learning, & neurological* disabilities in terms of usability, navigation, and user experiences?** This research question aims to assess the level of support that existing web modeling tools provide for users with *(i) visual*, *(ii) physical*, and *(iii) cognitive, learning, & neurological* disabilities focusing on usability, navigation, and overall user experiences. The study will quantitatively evaluate these tools against established accessibility standards by the Web Accessibility Initiative (WAI), and seek to identify specific areas where improvements can be made to enhance the inclusivity and accessibility of these tools for users with the mentioned disabilities.

2.3 Search process

The process is a manual search that uses a search string defined out of a set of search criteria and keywords (cf. Section 2.3.1) to identify relevant literature. In order to capture only relevant studies, additional exclusion criteria (cf. Section 2.3.2) are defined, and the findings are filtered accordingly.

2.3.1 Definition and Execution of the Search Query

The search string combines keywords related to disability and web accessibility *(D)* and conceptual modeling (*CM*). Various boolean operators, alternative spellings, and plural forms, as well as database adaptations, are used to make this connection.

The search query is defined as follows:

$$\text{Query} = (\bigvee CM_i) \wedge (\bigvee D_j) \textbf{ \textit{where}}$$

$$CM_i \in \{\textit{"Modeling Method"} \vee \textit{"Modelling Method"} \vee \textit{"Modelling Tool"} \vee \textit{"Modeling Tool"} \vee \textit{"Diagram Tool"} \vee \textit{"Modeling Editor"} \vee \textit{"Modelling Editor"} \vee \textit{"Diagram Editor"} \vee \textit{"Web Modeling"} \vee \textit{"Web Modelling"} \vee \textit{"Editor"}\}$$

and

$$D_j \in \{\textit{"Accessibility"} \vee \textit{"Disabilit*"} \vee \textit{"Impairment*"} \vee \textit{"Accessible Internet"} \vee \textit{"WCAG"} \vee \textit{"Web Content Accessibility Guideline"}\}$$

This search string was executed on some well-known scientific databases such as *Scopus*, *IEEE*, and *ACM*. The query was adapted according to the syntax of the databases, such that it would find all results contained in any or all of the *Title, Abstract* and *Keyword* of the literature. Additionally, the query was not constrained to specific years.

2.3.2 Exclusion Criteria

The results from different scientific databases were merged, and duplicates were eliminated. The following list of exclusion criteria is applied to identify and filter out irrelevant publications:

- **EC-1:** Publications not written in English.
- **EC-2:** Publications not related to the subject areas Computer Science or Engineering.
- **EC-3:** Publications with less than four or more than 60 pages.
- **EC-4:** Publications not accessible as full text or non-scientific papers (e.g., posters, extended abstracts, and similar).

2.3.3 Screen and select related publications

Reading Abstracts

This stage is intended to identify those publications especially relevant to the research scope of this literature review (cf. Section 2.2). To accomplish this, the abstracts of the publications are screened and discarded if they are irrelevant to the research question or do not fit its purpose. The relevance of the publications is determined according to the following categorizations:

- **YES** - if the abstract is suitable for the literature review scope.
- **NO** - if the content of the abstract is clearly out of scope.
- **NOT SURE** - if both the title and abstract do not have enough information to decide the relevancy of the publication and further investigation is needed.

The publications are further categorized into the following categories: **(i) related to accessibility & conceptual modeling**; **(ii) related only to (web) accessibility**; **(iii) not directly relevant to conceptual modeling**. Table 2.1 presents the definition of each category. The whole process is documented in a spreadsheet, and using categorization and comments simplifies filtering relevant and irrelevant publications.

Table 2.1: Categories used for the initial mapping

Category	Description
not directly relevant to conceptual modeling	This categorization applies to publications whose subject area is not directly related to conceptual modeling or web accessibility but discusses different accessibility solutions (e.g., smart or assistive technologies) that would provide valuable solutions or insights which could be possibly transformed and adapted to the domain of web accessibility in conceptual modeling.
related only to (web) accessibility	This categorization applies to publications discussing accessibility in a broad sense or focusing on web accessibility in any other domain than conceptual modeling.
related to accessibility & conceptual modeling	This categorization applies to publications that specifically address accessibility within the context of conceptual modeling and encompass all related subject matters within this domain.

Related Publications

Along with the search string, the search for relevant works is supplemented by two early and influential works (Luque et al. [LVPF14], and Wildhaber et al. [WSGK20]) to conduct an additional search for related papers via ConnectedPapers [1]. With ConnectedPapers, it is possible to explore the connections between scientific publications and discover relevant findings that have remained undiscovered by the search string. These two papers are selected because they specifically deal with accessibility problems in conceptual modeling and are likely to lead to further publications in this area.

2.3.4 Extract data and analyze findings

This step aims to obtain an overview of the selected publications' content to select only relevant and significant studies. This is achieved by screening the content of scientific works. Further, the publications are categorized more precisely in this step so they can later be assigned to the research questions. The publications are categorized by article metadata as described in Table 2.2. This step is applied to get an overview of the selected publications' distribution.

Table 2.2: Article Metadata used for classification

Category	Description
Country	The country in which this publication was published or where the authors are or were based at the publication date.
Document type	The type of published document, including *Conference Papers*, *Journal Articles*, or *Book Chapters*.
Subject area	The subject area describes the branch of knowledge in which the document is being published or sets its focus, e.g., *Visualization & Computer Graphics*, *Mathematical*, or *Web Engineering*.
Domain	The domain describes the field in which the content or the provided results or outcome of the work deals with, e.g., *Education*, *Technical*, or *Health*.

[1] ConnectedPapers, `https://www.connectedpapers.com/`, (Access: 18.08.2023)

Next, to identify patterns, trends, and themes across the retrieved publications, they are classified further according to topic-related categories, which include **disability and impairment types**, the **proposed solutions**, and the topics or issues that the **publication focuses** on, which are explained and defined in more detail in the next sections.

Disability and Impairment Types: Each study is classified according to disability and impairment types to determine which disabilities and impairment ranges are addressed in the selected publications. In general, there are many different types of disabilities and impairments. Disabilities are as individual as those who are affected by them. Depending on their severity, some are temporary and minor, while others last longer or are permanent. There are various causes for disabilities and impairments, including genetic disorders, injuries, and illnesses. Disabilities can be grouped into **five general categories** according to the Web Accessibility Initiative (WAI) [2] as defined in Table 2.3. Of course, it is not always possible to assign disabilities to one of these particular categories, as multiple disabilities, changing abilities, and situational limitations also exist.

Table 2.3: Classification of disability types by the Web Accessibility Initiative (WAI)

Disability	Description
Auditory	A person experiencing different extents of hearing loss.
Cognitive, Learning, & Neurological	A person experiencing neurodiversity, neurological disorders, behavioral, or mental changes. This may affect any part of the nervous system, such as speaking or hearing ability, or problems in comprehending information.
Physical	A person experiencing impeded movement, sensation, or control caused by muscular weakness, pain, limitation or lack of coordination, joint disorders such as arthritis, or missing limbs.
Speech	A person with a disability to speak clearly and be comprehended by others (e.g., difficulties in loudness or clarity of speech).
Visual	A person experiencing different extents of vision loss in one or both eyes (i.e., "low vision"), severe and uncorrectable vision loss in both eyes (i.e., "blindness"), or lack of sensitivity to brightness or (specific) colors (i.e., "color blindness").

Proposed solutions: This classification category determines if and which solutions are proposed or considered in the selected publications. The results were grouped into the classes described in Table 2.4. This classification is beneficial to discover whether these publications already offer solutions to the web accessibility problem or, on the other hand, which gaps in the provided solutions exist.

[2] WAI - Diverse Abilities and Barriers, https://www.w3.org/WAI/people-use-web/abilities-barriers/, (Access: 18.08.2023)

Table 2.4: Classification types for the proposed solutions

Subcategory	Description
Theoretical	This category is used for all publications that introduce discussions, methods, prototypes, possible solution approaches, and similar without technical artifacts or implementations.
Practical	This category is used for all publications that propose specific implementations, tools, and similar technical artifacts or their evaluation.

Focus and type of publication: This classification determines the types of papers, such as whether they focus only on literature, research work, solutions, or practical studies. Table 2.5 describes those classification types in detail. This categorization aims to determine the number of publications that offer new or proposed solutions compared to those that only provide an overview of the accessibility problem and present current development and research without solution approaches.

Both categories **Proposed Solutions** and **Focus and type of publication** have the goal of identifying the gaps in finding solutions for web accessibility problems in the context of this thesis.

Table 2.5: Classification for the focus and type of the publications

Subcategory	Description
Implementation/ Technical Aspect	A publication is mapped into this category if the conclusion proposes a specific tool, application, or any other ready-to-use software artifact. This category also includes code snippets, software extensions, and similar implementations.
Explanation / Solution Proposal/ Theoretical	A publication is mapped into this category if it contains theoretical artifacts, such as methods or solutions proposals to overcome an explained problem or similar.
Literature Review/ Discussion/ Evaluation of other researches	A publication is mapped into this category if the main part of the work summarizes or presents existing research or other publications. This includes problem discussions and other similar document reviews.
Tool Evaluation	A publication is mapped into this category if it evaluates or describes existing tools in its work.

2.4 Analysis & Synthesis of the selected publications

This phase involves a detailed examination of the final selected papers, evaluating their content as well as determining connections and statements between the papers. The research state and the papers' connection will be explained in detail in Chapter 3.

2.5 Summary & Interpration of the results

The sections above explain each phase of the systematic literature review and which tasks, definitions, and further information were necessary to analyze and interpret the findings. This section will provide a detailed description of each step's results and answer

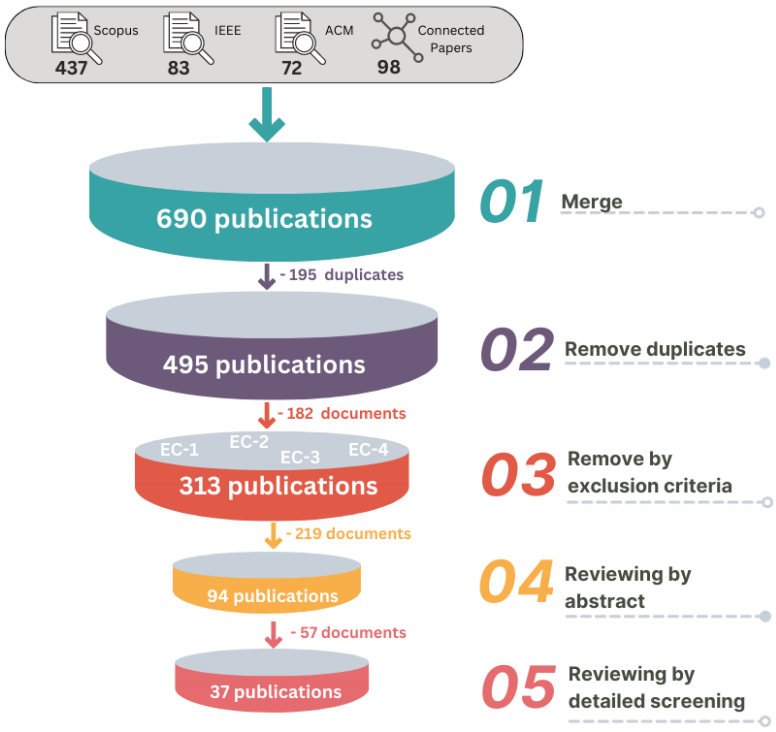

Figure 2.2: Document collection and major filtering steps

the defined research questions (cf. Section 2.2.2). Finally, a summary of the observations and analyses will be provided throughout this literature review process.

The results of the whole search process can be observed in Figure 2.2. The outcomes of each step illustrated in the Figure are summarized in the following paragraphs.

Step 1 & 2: Execution of the search query (cf. Section 2.3.1): The search was performed on 15.05.2023 on the three scientific databases *Scopus*, *IEEE*, and *ACM* and led to **592** publications. Additionally, the results retrieved from ConnectedPapers (cf. Section 2.3.3) have led to an additional **98** possibly relevant publications, which resulted in a total of **690** publications to start with (cf. Step 1 of Figure 2.2).

Afterward, all the findings were merged to eliminate duplicates. This step removed 195 duplicates based on the titles and the DOI (cf. Step 2 of Figure 2.2), leaving **495** publications.

Step 3: Filtering by exclusion criteria (cf. Section 2.3.2): Next, the filtering step with the predefined exclusion criteria filtered out 182 unsuitable documents and resulted in **313** publications (cf. Step 3 of Figure 2.2). The remaining 313 documents were analyzed based on publication years, as seen in Figure 2.3, which shows that the number of retrieved publications for this specific search result has increased significantly

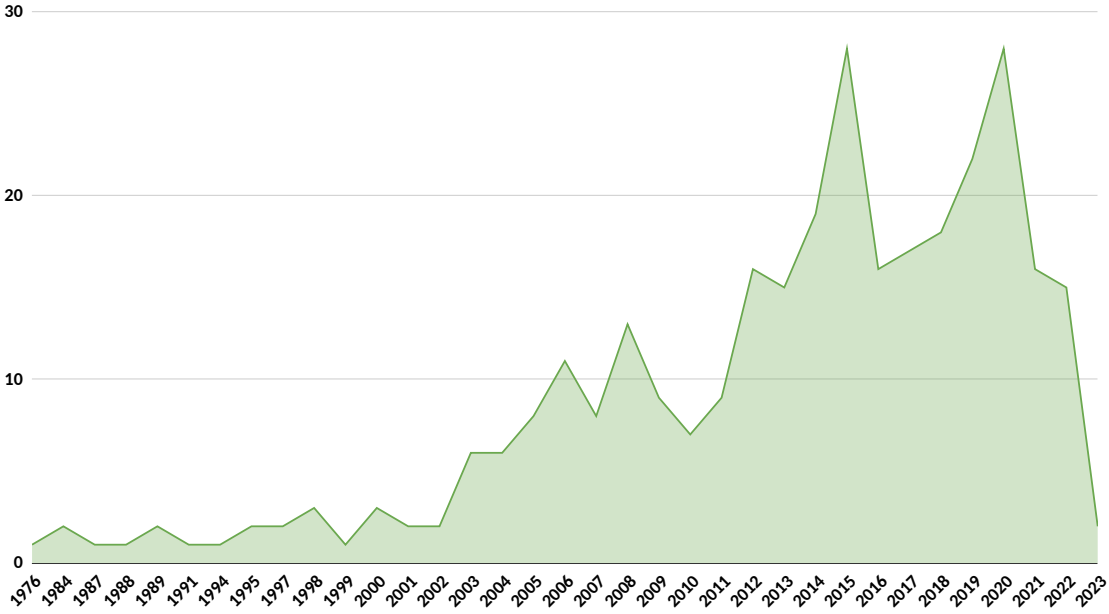

Figure 2.3: Retrieved documents categorized by publication year

Figure 2.4: Initial categorization of the 94 papers after abstract review (cf. Table 2.1)

over the years. Around 48% of the mentioned documents were published between 1976 and 2014, and even more than half, around 52% between 2015 and 2023. As a result, this topic is gaining more importance and will play an ever-greater role in the coming years.

Step 4: Screen and select related publications (cf. Section 2.3.3): In the next step, the 313 publications were further investigated to only select related publications by reading their abstracts and preliminary assigning the findings to the previously mentioned categories described in Table 2.1. With this step, 219 unrelated publications were detected and eliminated. Therefore, the last count of relevant publications is **94** (cf. Step 4 of Figure 2.2). Figure 2.4 visualizes the initial categorization of the found documents.

Undoubtedly, in Figure 2.4, it can be observed that the majority of the remaining publications after analyzing the abstracts deal with web accessibility in general. Considering the fact that there was no restriction regarding the publication year, only a few works have been published (23 publications), specifically in the field of conceptual modeling. On the other hand, there exists a wide range of documents that discuss the topic of web accessibility in different domains and subject areas, and around 47 were rated as

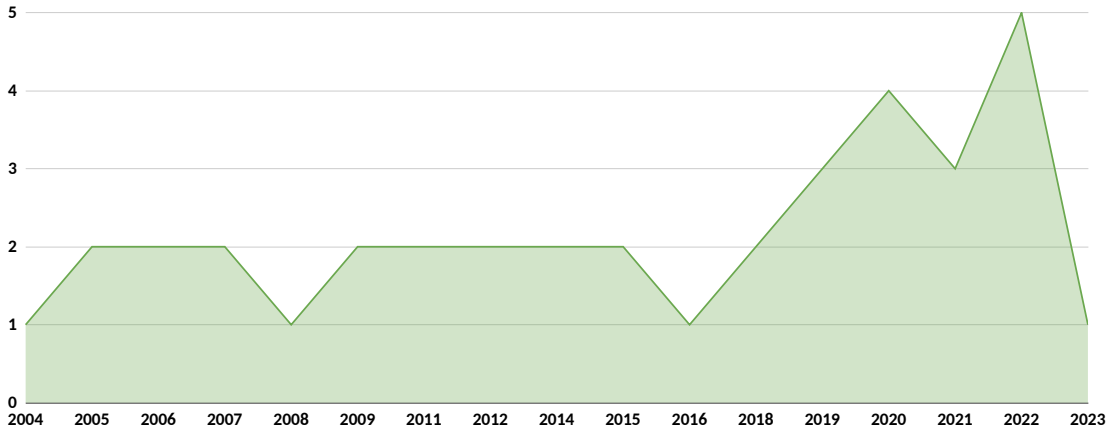

Figure 2.5: Trend of the 37 selected publications

possibly relevant. The remaining 24 documents were categorized as not directly relevant for conceptual modeling, as their content applies to domains unsuitable for this search context. Nevertheless, the obtained results require a thorough analysis to ascertain the potential for adapting and employing specific designs, processes, or similar findings within the context of interest for this thesis.

Step 5: Extract data and analyze findings (cf. Section 2.3.4): The process started with 94 documents, and after screening this set of publications, **37 documents** (cf. Step 5 of Figure 2.2) were marked as relevant. These papers have then been analyzed according to different aspects to respond to the research questions stressed at the outset

The detailed references for the 37 relevant publications can be found in **Appendix 1: Selected Publications**.

2.5.1 Findings & Analysis

The following section will present the categorization of the selected documents, their characteristics, and their types in more detail.

The diagram in Figure 2.5 illustrates the publication years of the final 37 publications. Evidently, the contributions relevant to this search topic have shown a notable increase, particularly over the last five years, while the oldest selected publications date back to 2004. This fact justifies the relatively low number of publications found, especially regarding web accessibility in conceptual modeling, as research in this domain has recently experienced significant growth.

In addition, the illustration in Figure 2.6 shows the categorization of the relevant publications according to the classes described in Table 2.1. After thoroughly analyzing the content of the publications presented in Figure 2.4, the 37 documents were selected as possibly relevant. Similar to Figure 2.5, observing the classification only for the selected

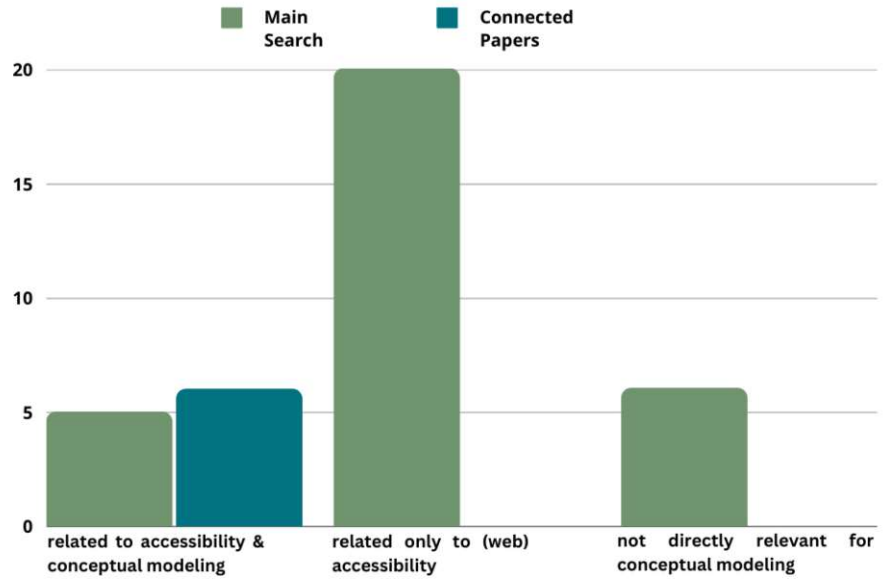

Figure 2.6: Selected publications categorized as defined in Table 2.1

works showed that the majority is focusing on web accessibility generally. Further, 11 out of 37 publications deal with the topic of accessibility in conceptual modeling. The remaining six publications were assigned to the category of not directly relevant for conceptual modeling, as they only provide valuable contributions and insights for other domains and topics but most probably could be adapted or transformed to use it to enhance the accessibility specifically in conceptual modeling. Furthermore, the illustration also presents clearly, that the search via ConnectedPaper was useful for finding publications in the field of conceptual modeling.

Furthermore, Figure 2.8 illustrates the distribution of the documents in specific subject areas and their publication source. It can be observed that around 84% of the eventually relevant papers originate from the search query while the other 16% originate from the search via ConnectedPapers. The majority of the documents were published in the subject areas of Web Engineering & Web Design & Web Content Generation (a total of 13), and Conceptual Modeling (a total of 11). It can be derived that our query was exhaustive concerning the core focus on research at the intersection of disability and conceptual modeling. At the same time, several relevant works were found in adjacent domains like disability and web engineering through ConnectedPapers.

It was evident from the retrieved information that recent research in the field of accessibility on the web has become increasingly relevant, especially within the field of conceptual modeling. However, the number of papers addressing accessibility in conceptual modeling

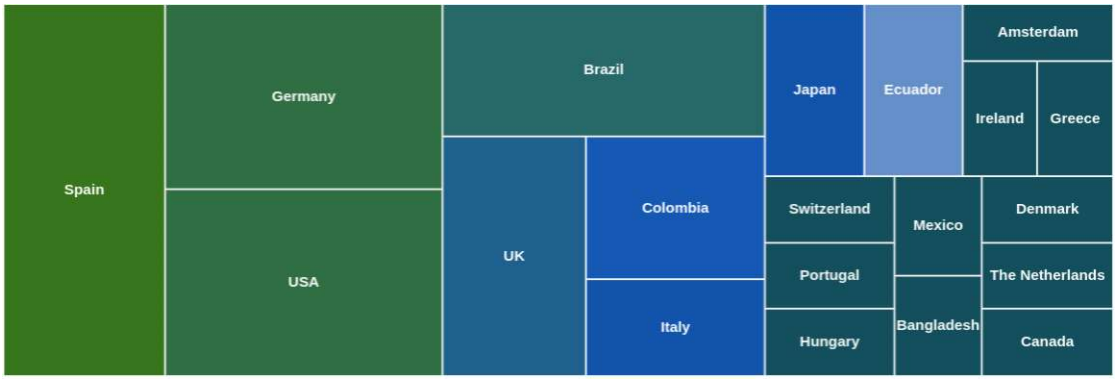

Figure 2.7: Publications grouped by countries of the contributed authors

still remains relatively small, which is most probably because this field is more or less newly growing. As a result, this thesis represents a valuable contribution to this newly developing field.

The treemap in Figure 2.7 presents the countries with the highest author contributions to the relevant publications. The illustrated number is based on the contribution of countries about the authors and not on the entire publication since, in many cases, a paper has contributing authors from different countries.

The top four countries with the highest author contributions to relevant publications in terms of this thesis originate from Spain, Germany, the USA, and Brazil. This distribution is not surprising since they are among the top countries with the highest rate of disabilities, and therefore authors from these areas are more likely to deal with this topic and think in a more solution-oriented way [3] [4] [SSO+20].

Figure 2.9 shows a categorization of the state of art of research on disability in conceptual modeling using the different disability categories (cf. Table 2.3) and the contribution types (cf. Table 2.4). The illustration is primarily intended to illustrate areas in which little to no research has been conducted, and thus a gap exists, or on the contrary, where research and publications already exist.

The black bubbles represent the total number of publications selected as relevant, whereas the green bubbles present how many out of the total publications are related specifically to conceptual modeling. Notably, most relevant publications handle visual disabilities, with 14 theoretical publications and 4 with actual technical artifacts, including tools or implementations. About ten publications generally focus on disabilities without concentrating on a specific disability category. For the remaining four disability types almost no contributions exist. A majority of the publications provide a theoretical

[3] Infographic - Disability in the EU: facts and figures, https://www.consilium.europa.eu/en/infographics/disability-eu-facts-figures, (Access: 18.07.2023)

[4] Disability Characteristics, https://data.census.gov/table?t=Disability&y=2021&tid=ACSST5Y2020.S1810, (Access: 18.07.2023)

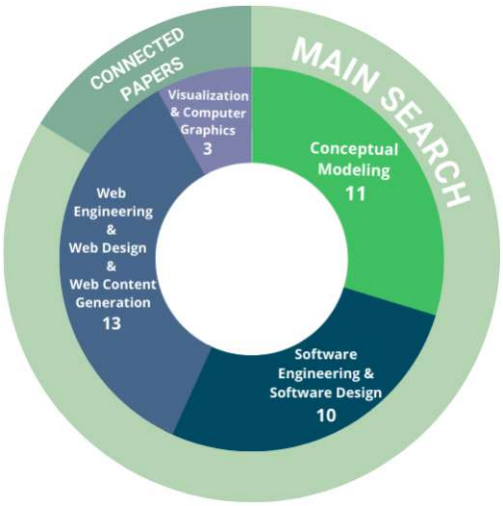

Figure 2.8: Number of publications based on publication source and subject area.

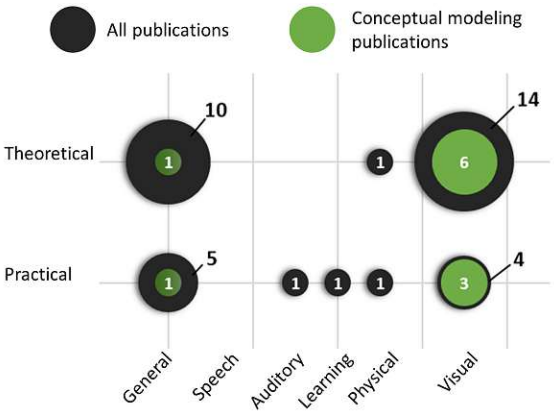

Figure 2.9: Overview of the disability types combined with the contribution type.

contribution. On the other hand, observing the publications related to conceptual modeling leads to the fact that only 11 of 37 selected papers are specifically handling modeling or modeling tools. Here, we can see that two contributions generally refer to disabilities without focusing on a particular type (marked in Figure 2.9 as General, 1/10 & 1/5), and the rest explicitly targets visually disabled users (marked in Figure 2.9 as Visual, 6/14 & 3/4). Furthermore, there was no existing research in the areas of Physical, Auditory, Speech, or Learning disabilities in the context of conceptual modeling

To sum up, it can be said that the existing research gaps about accessibility in conceptual modeling are related especially to the disability types *Physical*, *Cognitive, Learning, and Neurological*, *Speech*, and *Auditory*. Furthermore, there is a gap in providing ready-to-use technical artifacts, as around 70 % of the relevant documents are in the theoretical category and therefore do not propose specific implementations, tools, and similar technical artifacts, which would solve or improve disabled users' limitations or restrictions.

2.5.2 Results of the Research Questions

The previously defined research questions are answered in the following paragraphs based on the findings. The combined investigation of research questions **RQ-2** to **RQ-5** serves collectively as a response to **RQ-1**, effectively portraying the present state-of-the-art concerning this particular research objective.

- **RQ-1: What is the state of research and its evolution regarding web accessibility in the field of software engineering & information systems?** The publications in web accessibility have increased significantly over the years, based on Figure 2.3. The topic gains relevance with an increasing number and

diversity of available publications. The Figures 2.4 and 2.6 show the distribution of the documents grouped by the categorization of their topics (cf. Table 2.4). Through the Figures mentioned above, it is evident that most of the retrieved contributions relate to web accessibility in general. Furthermore, observing the selected publications (cf. Figure 2.8) also among this set of works, most of them can be assigned to the subject areas Web Engineering (13 out of 37) and Software Engineering (10 out of 37). At the same time, several relevant works were found in adjacent domains like disability and web engineering through ConnectedPapers. For this reason, it can be concluded that (web) accessibility is a topic that gains in relevance with an increasing number and diversity of available publications from day to day. Furthermore, the remaining research questions (RQ-2 to RQ-5) collectively contribute to addressing the research question of RQ-1.

- **RQ-2: What is the state of research and its evolution regarding web accessibility in the field of conceptual modeling?** Here is an entirely different situation compared to the outcomes of **RQ-1**. According to Figure 2.4 and 2.6, publications related to accessibility in conceptual modeling make up only a small fraction of all publications, which concludes that the research in this particular field is not far developed yet. Additionally, it is observable that most publications selected as relevant specifically in this area needed detailed research as they were retrieved with the additional help of ConnectedPapers, rather than with the defined search query (cf. Figure 2.6).

  The chart in Figure 2.6 illustrates only the selected and relevant publications grouped by which field their content is focusing on or setting the target (cf. Definition of classification groups in Table 2.1). According to this chart, it is clear that also among this set of publications, more than half of the findings (20 documents out of 37) deal with the topic of web accessibility more generally compared to the found literature dealing specifically with web accessibility issues in the field of conceptual modeling (11 documents out of 34).

- **RQ-3: Which disabilities or impairments are covered in the literature?** As seen in Figure 2.9, the relevant publications are not covering most of the disability types (cf. Table 2.3). The figure clearly presents that among these publications, visual disabilities are well represented. On the other hand, the second largest group of publications discusses disabilities in a generic sense without focusing on concrete disability categories and their needs. Observing publications specifically targeting conceptual modeling, it is clear that visual disability is also well represented compared to the disability groups **Speech**, **Auditory**, **Learning**, and **Physical**, which have gaps regarding research for modeling related topics.

  Regarding the contribution types, there is also gap illustrated, as the majority of the publications are assigned to the category **Theoretical**, which leads to the result, that there is an evident lack of ready-to-use or implemented solutions.

As a result, it is evident that for the majority of the existing disability or impairment types, there is almost no research, solution approaches, or even realized solutions in the field of conceptual modeling.
To sum up, this leaves two different gaps, namely gaps in research and literature ...

1. ... especially targeting the field of conceptual modeling.
2. ... regarding different disability and impairment types than visual.

- **RQ-4: Which solutions are proposed to improve web accessibility?** The relevant papers have various foci. A detailed description and analysis of the valuable contributions and insights can be found in Chapter 3. Nevertheless, the analysis shows that most relevant publications (25 out of 37) mainly discuss different theories or propose potential methods. Only 12 of 37 relevant publications present any existing tool or implementation (cf. Figure 2.9). Furthermore, it is distinctly observable that the number of contributions specifically targeting conceptual modeling is relatively low, with only 11 out of 37 publications, with only four papers proposing practical input in this research field (cf. Figure 2.9). Additionally, there is a clear gap regarding implementations and solution contributions specifically targeting physical, learning, auditory, and speech disabilities, as no relevant and suitable papers were found here.

- **RQ-5: How effectively do current web modeling tools support individuals with *(i) visual*, *(ii) physical*, and *(iii) cognitive, learning, & neurological* disabilities in terms of usability, navigation, and user experiences?** An extensive assessment of ten well-known web modeling tools and the retrieved findings and analysis answering this research question can be found in Chapter 4. In summary, the tool evaluation identified a critical criterion for physical disability support, **full-keyboard support (P1)**, that none of the assessed web modeling tools met. Physical disability support had the highest number of unsatisfactory tools. Conversely, visual and cognitive, learning, neurological disability support demonstrated sufficient assistance, with only minor issues with specific tools.

The outcomes of this systematic literature review show that there are valuable contributions and insights in the field of web accessibility, and the topic is of increasing importance, so research in this area will most probably continue to grow. While the web and software engineering communities have made significant contributions with standards, methods, and tools, conceptual modeling research is currently scarce and focused on visual disabilities. Even if the area of conceptual modeling research is slowly gaining interest, there are still apparent gaps and, above all, missing concrete solution proposals or solutions to remove the barriers or limitations out of the way that disabled users encounter while dealing with models or modeling tools. In this area, the work is still in the early stages, as the existing ones focus mainly on blindness and other kinds of visual impairments and ignore entirely the other presented disability types (cf. Table 2.3).

In conclusion, the SLR has highlighted the research gaps in realizing more inclusive conceptual modeling. Furthermore, a detailed description of the selected publications, which present the state-of-art for web accessibility and accessibility regarding conceptual modeling research, can be found in the next Chapter 3.

CHAPTER 3

State-of-the-Art Research and Literature

This chapter presents a comprehensive analysis of 37 selected papers identified as highly relevant to the research topic of web accessibility in conceptual modeling, as described and retrieved through the Systematic Literature Review (cf. Chapter 2). This chapter provides a detailed overview and synthesis of these contributions' content, methodologies, and findings as they relate to the research objectives of this thesis. Critically examining these papers aims to gain a deeper understanding of the existing knowledge landscape, enabling identifying the gaps and opportunities for further investigation.

3.1 Introduction

A list of relevant publications can be observed in **Appendix 1: Selected Publications**. After an in-depth analysis of the selected contributions, they are assigned and grouped thematically according to their content, to the topics **Understanding & Advancing Web Accessibility** (cf. Section 3.2) and **Inclusivity in Conceptual Modeling** (cf. Section 3.3).

3.2 Understanding & Advancing Web Accessibility

This section reviews publications that examine web accessibility from various perspectives and fields and provide a general overview and insight.

Understanding Web Accessibility is essential before enhancing it. Usability and accessibility are often conflated, but those terms differ in meaning. Tanaka et al. (2005) [TBdR05] highlighted this distinction. Usability focuses on optimizing the ease of use for the general user base, while accessibility aims to accommodate the needs of special-needs

users. Even though the meanings are different, both approaches do influence each other. When software exhibits usability issues for non-disabled users, it will likely present even greater challenges for those with special needs. The authors of this paper suggest that by combining usability and accessibility evaluations and implementing appropriate solutions, both aspects can be significantly improved.

Ruth et al. (2011) [RJ11] comprehensively classified limitations associated with disabilities, barriers, and application classes. They categorized applications into three groups: form-based, extended form- or editor-based, and media-rich, and outlined common disability barriers that are likely to be encountered. Specifically focusing on web modeling tools, these tools belong to the extended form- or editor-based application class. For this application class, the researchers identified some typical barriers, including font sizes, visibility of editor functions, and alternative participation methods. In the course of evaluating web modeling tools for this thesis (cf. Chapter 4), it was found that these barriers identified in 2011 still persist. Despite being recognized some time ago, these barriers and limitations continue to pose challenges for users with disabilities when utilizing such applications. These persistent challenges highlight the ongoing importance of research in this area and the need for continued efforts to find effective solutions.

Still, in the newest publications, it is observable that various limitations and barriers exist that people with disabilities encounter in using different services or media on the web. Due to these challenges, users with special needs often feel excluded and are incapable of participating in today's advanced technology. The most crucial aspect here is not to see accessibility as an additional feature to be added afterward to the web product but to ensure the design and creation of accessible web applications right from the beginning. Among other publications, Zdenek et al. (2019) [Zde19] also emphasizes that people's disabilities should be seen as a form of diversity rather than an issue, problem, or barrier. Adopting this perspective in both practices and theories promotes a positive outlook on inclusion, eliminating the notion of it being an extra burden and instead considering it as a valuable addition.

Albusays et al. (2021) [ABD+21] emphasize the significance of accessibility in software engineering. They stress the need to prioritize inclusion and representation of underrepresented groups in research, especially in the software engineering industry, where professionals from different backgrounds collaborate. To improve accessibility, it is essential first to understand how to achieve it. Therefore, the authors of this publication propose that inclusive research is the starting point for promoting inclusivity.

Synopsis: The publications in this section highlight the importance of research and solutions in accessibility in web applications or websites. It is observable that the challenges of web accessibility have been preserved throughout the years, and the importance of web accessibility increased over time. The main idea emphasized in these publications is the need for a fundamental shift in approaching web accessibility. Achieving web accessibility requires a new approach that involves viewing diversity as a positive aspect rather than a burdensome task. By prioritizing inclusion research and incorporating the insights

gained, web applications can be designed to be inherently diverse and inclusive right from the beginning. Embracing diversity positively and integrating inclusion research can lead to more inclusive web applications from the outset.

3.2.1 Accessible Web Creation: Building Inclusive Websites & Applications

The remaining relevant publications focus on findings ways and solutions to design or create barrier-free websites. Below, the contributions are grouped according to their primary focus or proposed solution to create accessible websites or applications.

3.2.2 Creating accessible web content with CMS

In 2007, Schulz et al. [SP07] explored the automatic creation of accessible websites using content management systems (CMS). CMS is a user-friendly software that allows non-technical users to create, manage, and publish digital content. They showed that CMS can achieve some level of accessibility but has limitations.

To overcome these limitations in 2015, Pascual et al. [PRG15] introduced a web-based tool that improves web accessibility by analyzing HTML code and providing users with specific, more precise, and actionable recommendations for addressing accessibility issues, benefiting both technical and non-technical users.

Throughout the creation of websites, the use of images especially can become an accessibility burden. This is because web pages have become increasingly graphical, and the count of used images is still growing [Web23]. Therefore, Villarroel-Ramos et al. [VRSGLM18] proposed an architectural metamodel in 2018, focusing specifically on image accessibility within online editors, emphasizing the importance of addressing accessibility concerns for different content types.

Similarly, Acosta et al. [AAVSULM18] assessed 2018 the accessibility of online content editors and provided guidelines for creating accessible web content using these editors. They addressed common barriers explicitly faced by visually impaired users and stressed the necessity of removing such obstacles.

Lastly, a newer contribution related to this section was published in 2022 by Csontos et al. [CH22]. The authors devised methods to enhance the accessibility of CMS-built websites, recognizing persistent accessibility challenges. Although their approach differed from previous studies, it aligned with the shared goal of improving accessibility within the CMS domain.

Synopsis: These research papers collectively demonstrate notable progress in improving web accessibility through user-friendly solutions, targeted enhancements for diverse content types, adherence to accessibility guidelines, and addressing specific challenges related to content management systems (CMS). Given that websites are created not only by experienced designers but also by individuals with varying levels of technical

and accessibility knowledge, the main objective of these researches was to explore ways of making web content accessible without requiring extensive expertise. To achieve this, additional automated web tools, guidelines, and assessments were utilized. In conclusion, these studies significantly contribute to advancing web accessibility. They provide valuable insights for developing web modeling tools and foster inclusive digital experiences, ensuring accessibility for all users.

3.2.3 Enhancing Web Accessibility: Optimizing Existing Content

The selected papers primarily focus on solutions for people with visual disabilities or impairments. They analyze the current state and propose appropriate accessibility measures for learning environments, various editors, and collaborative tools. The papers concentrate on improving accessibility for individuals with visual impairments, addressing diverse aspects and contexts. Despite their varied areas of investigation, these papers share a unified objective of enhancing accessibility and user experience for visually disabled individuals.

Komada et al. (2006) [KYKS06] and Biswas et al. (2009) [BR09] both contribute to improving accessibility through innovative approaches. Komada et al. developed a math document editor that combines speech output and Tacticle Graphics, enabling visually disabled students to access scientific information effectively by sound and touch. Similarly, Biswas et al. provided a user model to design and evaluate interfaces for people with visual impairments, considering eye movement patterns and visual search strategies, which can be calibrated to individual users and is accurate enough to select the best interface based on visual search time. Both studies aim to enhance accessibility by tailoring interfaces to the specific needs of visually impaired individuals by providing new solution approaches.

On the other hand, some publications focus on analyzing or extending existing solutions to enhance their accessibility state.

In Buzzi et al.'s (2014) [BBL+14] study, the focus shifts to the challenges faced by blind individuals when using screen readers with collaborative editing tools. This research emphasizes the importance of clear feedback and alerts, such as marking modified parts of the document and providing information about the status of collaborators. On the other hand, the newest publication by Lee et al.(2022) [LPS+22] stressed the importance of designing online collaboration tools with low-vision users in mind. Their tool, MagDocs, displays comments and document changes alongside the relevant text to accommodate low-vision screen magnifier users. Both publications focus on the effective utilization of assistive technologies. Their findings reveal that simply providing assistive technologies is insufficient if the underlying systems are not designed to be compatible with these aids. By addressing these challenges, collaborative editing tools can be more accessible and inclusive for individuals with low vision or blindness.

Similarly, Baker et al. (2021) [BNB21] also had an interest in analyzing the accessibility of collaborative tools. They comprehensively reviewed technologies used by blind or visually

impaired individuals for collaborative writing and editing in professional settings. Their findings shed light on the typical barriers faced in editing, writing, and collaborating, such as difficulties in reviewing tracked changes and comments, cognitive overload, navigational problems, and more. By documenting the usage of various technologies, they provide insights for developing accessible, collaborative editing tools that address the specific needs of visually impaired individuals.

While most of the previously mentioned papers deal with editors-like systems, another contribution with a focus on accessible visualization was provided by Gosling et al. (2021) [LWLG21]. They developed a grammar-based toolkit for genomics data visualization. This toolkit covers various visualization types and offers shortcuts to simplify the creation process, eliminating the need to specify every property individually. It enables users to generate informative and visually appealing visualizations for genomics data in an accessible way.

The only publication in this set dealing with a different disability type than visual is by Morato et al (2020). [MCSC+20]. They created a web tool to enhance text readability based on easy-to-read guidelines. It detects difficulties, visually indicates barriers, and offers recommendations for improvement, particularly benefiting individuals with cognitive and neurological disabilities.

Synopsis: These research papers share a common objective of improving accessibility, particularly for the visually impaired, and one paper for individuals with cognitive disabilities. They address various aspects like document editing, collaborative tools, and visualizations to create more inclusive environments. The publications highlight the need for tools to be well-structured, compatible with assistive technologies, and designed with built-in accessibility features to foster inclusivity for individuals with disabilities, specifically focusing on working in a collaborative environment. Providing textual alternatives for visualizations is a common solution to enhance accessibility. Additionally, it is observable that common issues have remained over the years, as the publication years are between 2006 and 2022, and this topic is still currently important.

Enhancing Web Accessibility in Learning Environments

The research papers by Avila et al.(2012, 2014) [ABFG12, ABFG14], Sato et al.(2009) [SKTA09], and do Nascimento et al.(2019) [dNBdOBdMO19] share a common goal of improving accessibility, especially in the context of learning environments. Those researches emphasize that accessibility is crucial not only for industry tools but also for various learning tools to ensure inclusivity for disabled learners or students. The described publications below were presented between the years 2009 and 2019. The time interval between the publication years also shows that the issue and goal have remained almost the same over the years.

The study by do Nascimento et al. [dNBdOBdMO19] focuses on investigating the accessibility of Learning Management Systems (LMS) for blind students. The research

revealed limited accessibility features within LMS platforms, making it challenging for blind students to participate in online courses effectively. Their work emphasizes the importance of addressing accessibility barriers in educational technology platforms.

Addressing the previously mentioned gaps was achieved in the research by Avila et al. [ABFG12, ABFG14]. The authors introduced the Web Content Accessibility Plugin, utilized in a learning environment editor, to guide teachers in creating and editing web content that meets the minimum accessibility requirements for all users, regardless of their disabilities. Their focus was on enhancing accessibility in the educational setting.

Instead of creating extra accessible content, Sato et al. [SKTA09] had the idea to enhance the accessibility of already existing content. The researchers proposed a reading flow technique to improve the accessibility of web pages and other documents by visualizing a path that "serialize" the content. This technique facilitates easier navigation and comprehension of information, benefiting users with diverse abilities.

Synopsis: Collectively, these papers demonstrate efforts to enhance accessibility in learning environments, specifically focusing on tasks like educational content creation and document readability. By considering their insights, it is possible to gain a comprehensive understanding of the challenges faced by individuals with disabilities and the advancements made to promote inclusive and accessible experiences in the educational context.

3.2.4 Testing Web Accessibility: Evaluating Digital Inclusivity

Web accessibility goes beyond creating accessible content or web services; it also involves testing the fulfillment of accessibility features. Two of the selected publications specifically focus on accessibility testing.

Weber et al.(2007) [WW07] and Leporini et al.(2007) [LPS07] both recognized the importance of web accessibility a long time ago and advocated for its testing and implementation within content creation tools. Although their approaches differ, they share a common objective of improving accessibility practices.

In terms of testing web accessibility, Weber et al. [WW07] emphasize the significance of considering accessibility testing as a distinct process from traditional web testing. They highlight the need for specific methodologies and techniques specifically designed to ensure accurate and reliable accessibility testing. Meanwhile, Leporini et al. focus on implementing accessibility guidelines within content creation tools. They address the challenge of managing multiple guidelines by proposing an environment that simplifies guideline implementation, making it easier for content creators to adhere to accessibility standards.

Both papers underline the importance of providing content creators with the necessary tools and resources to ensure web accessibility. Weber et al. stress the need for appropriate testing methodologies, while Leporini et al. emphasize the significance of a simplified implementation environment for guidelines. By aligning their efforts, these studies

contribute to the advancement of web accessibility practices and the creation of more inclusive digital content.

Synopsis: In conclusion, the papers highlight the common goal of promoting web accessibility through testing methodologies and guideline implementation within content creation tools. Their research provides valuable insights into ensuring accessible digital content and can guide developers, designers, and content creators in improving web accessibility and fostering inclusive online experiences.

3.2.5 Accessible Coding Environments

In recent years, several research papers have also aimed to enhance coding experiences for individuals with disabilities, addressing various challenges and proposing innovative solutions.

Paudyal et al. (2022) [PCWFP22] and Munoz et al.(2023) [MBE+23] both recognized the need to improve coding accessibility for individuals with physical and motor disabilities. Paudyal et al. developed a prototype code editor with multimodal voice control, tailoring code navigation features for speech interactions. The aim was to integrate voice-controlled methods with traditional input devices such as keyboards and mouse, which was efficient according to their evaluation. Munoz et al. also contributed to inclusive coding by developing a software tool called Mancodev. This tool enables individuals with motor disabilities to write source code using voice commands, accommodating those with limited physical dexterity. By leveraging voice-controlled programming, Munoz et al. aimed to empower individuals with motor disabilities to engage in coding activities effectively.

Ehtesham et al.(2022) [EUHMB22] introduced a grid and keyboard-based coding approach, utilizing a tabular structure to represent source code, facilitating organized and navigable information.

All three studies sought to create inclusive coding environments, making coding more accessible for blind individuals.

Synopsis: These research papers collectively highlight the shared objective of creating inclusive coding environments for individuals with disabilities. They emphasize the importance of adapting coding practices and tools to meet diverse needs through multimodal voice control, grid-based coding, or voice-command programming. By addressing accessibility barriers, these studies contribute to fostering equal opportunities and promoting inclusivity in the coding domain.

3.3 Inclusivity in Conceptual Modeling

This section includes and describes all the publications handling accessibility issues, specifically aiming at conceptual modeling. From the 37 publications, **11** are assigned to this particular section. Furthermore, out of these results, it is observable that this

particular topic is gaining significant importance as almost half of the publications were published in the last four years (5 out of 11, between 2019 and 2021). Additionally, it is remarkable that all of the mentioned contributions focus on visual disabilities or impairments, mainly targeting blindness. On the other hand, most of these publications (6 out of 11) directly address the usage of UML diagrams specifically.

3.3.1 Accessibility State & Issues in Conceptual Modeling

The inclusion of disabled users in conceptual modeling tasks has gained interest due to advancements in modeling editors and the importance of visualizations in software development. However, individuals with visual disabilities face significant challenges in working with diagrams and collaborating effectively in the software development process.

Various notations, syntaxes, and diagrams exist for different purposes in software development. However, the current state-of-art contributions mainly focus on UML diagrams. UML diagrams are widely used and are considered as the standard for visualizing and documenting software systems. They provide a standardized way to represent software design, capture requirements, define system architecture, and model system behavior. By making UML diagrams accessible to visually impaired individuals, they can actively participate in software development and collaborate with sighted developers [LVPF14].

Luque et al. (2014) [LVPF14] highlight the need for the inclusion of visually impaired users in conceptual modeling for education and industry. However, UML modeling tools still lack accessibility for visually impaired individuals, despite some attempted solutions. Making UML diagrams accessible remains a challenge, and collaboration between sighted and visually impaired users requires further consideration for a seamless working environment that accommodates everyone's abilities.

Torres et al. [TB19] conducted a systematic literature review in 2019, which analyzes the previously mentioned lack of accessibility solutions in modeling. Their goal was to find publications from the past six years that offer approaches to making diagrams accessible for blind people while considering interaction and collaboration possibilities. The review revealed a significant number of individuals with visual impairments who could benefit from accessible diagrams. The most common existing solutions involve using audio descriptions or tactile diagrams (cf. Tacticle Graphics) with embossed or raised surfaces to represent visual information. However, the review also identified a gap in providing solutions for collaboration between disabled and non-disabled users. Existing literature primarily focused on the perception of diagrams or models rather than facilitating collaborative work. Regarding the approaches, similar solutions have also resulted in the systematic literature review of this work (cf. Chapter 2).

In summary, it can be said that the **two main questions** to work on are, first, how to represent graphical elements in an understandable and interactable format for modelers with visual disabilities and secondly, how to provide a collaborative environment for users with and without visual impairments.

To find answers and solutions to these challenges, some other researchers analyzed these gaps and provided some solution approaches. These approaches aim to focus on the other senses, like sound or touch, to simplify and allow the contribution of disabled modelers through alternative ways, which will be explained below.

Inclusive Solutions: Enhancing Accessibility with Physical Aids

Some of the existing solution attempts rely on physical aids. Wildhaber et al. (2020) [WSGK20] propose a solution that allows visually impaired users to read UML diagrams on a mobile device and supports their creation. This paper demonstrates the potential of using mobile devices, specifically iPads, to make UML class diagrams more accessible for visually disabled modelers. As a result, a prototype was developed, a physical overlay for the iPad screen. This overlay is in the form of a grid, and each tile on the grid represents one class of the diagram and allows various interactions between classes, which are implemented using custom gestures.

On the other hand, Loitsch et al. (2012) [LW12] presented an approach using tactile representations (cf. Tacticle Graphics) and haptic interaction on touch-sensitive devices. The authors developed a screen explorer for UML, which preserves the layout information and textual labels of diagrams. The diagram utilizes modular audio-tactile screen explorer technology. This technology enables blind individuals to access information on the screen, such as graphical data and the relative position of geometric shapes, through pins. It allows reading UML diagrams by blind and sighted people in a software system consisting of a standard UML editor and assistive technology while ensuring coherence. The results suggest that blind individuals can retain orientation and successfully answer questions about the diagrams when using the proposed approach.

Synopsis: Both papers aim to improve UML diagram accessibility for visually impaired individuals. They employ different methods but aim to make UML diagrams easier to understand and navigate using additional physical aids. These approaches showcase technology's potential to create more inclusive environments for UML modelers with visual impairments, particularly in existing modeling editors. However, using additional physical aids, such as printing and storing them, can be time-consuming and expensive. To address this, web solutions integrated into editors or provided as web or software solutions offer a more efficient and cost-effective alternative. These solutions can enhance accessibility for visually impaired users and streamline the UML diagramming process.

Inclusive Solutions: Enhancing Accessibility with Audio & Sound

One possible solution attempt was the implementation of an auditory interface. In this specific area, some of the found publications have provided interesting approaches, which are Metatla et al. [MBKS08] in 2008, Torres et al. [TBP+20] in 2020, and De Carvalho et al. [dCA21] in 2021.

Metatla et al. [MBKS08] conducted research on providing non-visual access and manipulation of graphically represented information. Specifically, they focused on constructing relational diagrams in an auditory interface. Their approach involves using *multiple perspective hierarchies* to offer structured access to relational diagrams, presenting the same information from different angles or viewpoints. The authors designed two interaction strategies for constructing and manipulating these diagrams through an auditory interface: the *guided* and *non-guided* strategies. In the guided strategy, an agent communicates with the disabled person, guiding them through the process of editing or exploring models in a conversation-like manner. In contrast, the non-guided approach allows users to explore and locate existing models independently, then vocally command actions like creating or deleting. A study involving sighted users found that the non-guided strategy resulted in faster interaction times, while both strategies supported similar levels of diagram comprehension.

Contrasting the previous approach, which focuses only on a single user interacting with a diagram, Torres et al. [TBP+20] presented an auditory solution focusing on collaboration. The paper proposes the sonification of workspace awareness elements to enable accessible groupware features for blind people. Sonification involves transforming data relations into auditory signals with specific meanings, allowing blind users to interpret and communicate actions in a shared working environment through various sounds signaling different categories of awareness. In the study, blind participants could reasonably perceive and understand the changes made by other users in the collaborative diagram modeling application using auditory cues and awareness signals. The results demonstrated the potential of sonification in a collaborative workspace, with minor flaws and areas for improvement. Overall, the study highlights the need for inclusive groupware applications and provides a practical solution for enabling blind individuals to participate in collaborative diagram authoring activities.

The recent publication by De Carvalho et al. [dCA21] introduces a model-driven software engineering approach to building a voice/audio editor for domain-specific languages (DSLs). The prototype offers a flexible interaction model for end-users to create, read, update, and delete models through speech recognition and voice synthesis tools. It supports both vocal and non-vocal sounds, as well as gestures, expanding the interaction possibilities. Similar to previous publications, this prototype combines various aspects to provide a versatile solution applicable to any domain-specific modeling language. It enables users to perform basic CRUD operations on models using voice commands and navigate through a table approach. The prototype includes voice recognition and synthesis technologies, allowing users to record and reproduce sounds associated with specific actions. Moreover, the prototype is designed to be customizable, enabling users to adjust the voice synthesizer's volume, pitch, and speed according to their preferences and needs. Overall, this innovative approach showcases the potential of voice and audio technology to enhance modeling editors for DSLs, providing a more inclusive and user-friendly experience.

Synopsis: The last three papers share a common goal of finding alternative approaches for enhancing accessibility for visually impaired individuals, primarily focusing on leveraging auditory elements and sound-based interactions. These innovative methods include auditory interfaces, sonification, and voice/audio editors to provide non-visual access and interaction with graphical information. Notably, the approaches varied between single-user and collaborative environments. These studies aim to empower visually impaired users with more inclusive and user-friendly technology environments by prioritizing sound as an alternative medium. Their research collectively contributes to breaking down barriers and advancing accessibility for individuals with visual disabilities, ensuring equal access and participation in both individual and collaborative settings.

Inclusive Solutions: Enhancing Accessibility using alternative formats

Seifermann et al. (2016) [SG16] conducted a systematic literature review to survey textual notations for representing models and diagrams in software engineering. The survey aimed to classify different textual notations and addressed three key objectives: determining what can be edited based on notation definition, how it can be edited in modeling environments, and the level of standardization and tool support. The results showed that most notations have limited coverage of the UML, leading developers to use multiple notations to capture all aspects of UML models fully. Accessibility-wise, the survey revealed varying user editing experiences among different notations. Some notations offer a user-friendly editing experience with graphical views and real-time collaboration, while others rely more on text-based approaches, potentially requiring greater expertise to use effectively. Additionally, the paper discusses the accessibility of textual notations concerning their syntax, editors, and modeling environment. The discussion notes that textual versions of notations tend to be more intuitive for developers and require less space for representation, making them a viable option for certain modeling tasks.

The authors Cross et al. (2020) [CCD20] also explore the use of textual notations but with a different approach. They propose an online system designed to be accessible and user-friendly for visually impaired users. This system allows users to define diagrams and automatically convert their graphical components into natural language text. The goal is to enable users to fully understand each part of the diagram without the need for tactile versions or other methods. The converted text can be accessed via different screen readers, allowing users to use their preferred assistive technologies. Traditionally, graphically visualized diagrams or models lack sufficient textual information. However, this system transforms the diagram's semantics into sentences describing each element of the diagram. For instance, it might produce a sentence like "This diagram is a flow chart. Object number one is labelled as 'start' and leads onto object number two." The system underwent testing with visually impaired participants, who found it helpful in understanding and visualizing the diagrams. This approach offers a promising solution to enhance accessibility for visually impaired users, bridging the gap between graphical representations and textual comprehension.

Another approach that relies on transforming the graphical means into an alternative format was proposed by King et al. (2004) [KBC+04]. They propose an approach focused on enhancing accessibility for visually impaired users, specifically concerning UML diagrams. Their contribution involves transforming UML diagrams into XML format, which is then made suitable for visually impaired users. The challenge lies in effectively presenting the information for sighted and visually impaired users. The authors introduce the environment TeDUB (technical diagram understanding for blind people), which converts UML diagrams to the XMI format, which is a UML interchange format, and further converts it into a format that is understandable by blind users without the need for additional support or a sighted person to convert the diagram in an understandable format. However, this approach faces technical challenges due to different existing UML notations and semantic difficulties in representing the graphical information meaningfully. To enhance accessibility for visually impaired individuals, TeDUB incorporates various components, such as text fields that can be accessed via a screen reader, providing node, connection, and annotation information. A hierarchical tree structure facilitates navigation using cursor keys, offering details about node connections and the existence of siblings. The environment also supports different navigational means, context sounds, and more to improve the user experience. By addressing the specific needs of visually impaired users and transforming UML diagrams into a more accessible format, the TeDUB environment bridges the gap between graphical representations and meaningful comprehension for individuals with visual disabilities.

Lastly, Seifermann et al. (2015) [SG15] observe this accessibility issue from another perspective. They especially highlighted some problems of existing textual solutions. Existing textual concrete syntaxes are limited in at least one of the categories: *completeness*, *comprehensibility*, or *accessibility*. For instance, miming graphical elements such as arrows with a minus and a greater symbol affects accessibility. This paper proposes a collaborative editing environment consisting of an accessible textual UML editor with user support and consistency preservation mechanisms. The authors propose an accessible editing environment for UML models that combines textual and graphical editors. The provided environment is based on different syntaxes for the user's needs. Accordingly, this means that each user can choose their preferred syntax, and the changes will be converted in real-time into the preferred syntaxes of the remaining users to ensure collaboration between all users regardless of their abilities. As a result, this is achieved through a common model approach, where all information is stored in a single model that can be edited using different syntaxes. Grammar-based editing ensures consistency by providing templates and predefined editing operations. Additionally, accessibility features such as audio notifications and navigation aids for visually impaired users were provided.

Synopsis: The research papers mentioned above share a common objective of improving accessibility for visually impaired individuals by using alternative formats to convert graphical components. The main focus is on transforming complex graphical information into formats that are best suited, in that specific case, for the abilities of visually disabled users. These alternative formats often involve leveraging natural language text or other

syntaxes usable with assistive technologies to facilitate comprehension and interaction with diagrams and models.

3.4 Conclusion

The conclusions drawn from these state-of-the-art papers emphasize the shared commitment to enchance accessibility for individuals with disabilities. The studies demonstrate a continuous effort to overcome challenges and promote inclusivity in various domains, ranging from web applications and learning environments to coding and modeling. One prevalent theme across these papers is recognizing the need for a fundamental shift in approaching accessibility. By embracing diversity as a positive aspect rather than a burden, researchers strive to develop inclusive solutions right from the outset.

Despite the extensive research and review conducted on accessibility in older publications, it is evident that disabled individuals still face barriers and limitations when using various websites and web applications. These obstacles prevent a significant portion of the population from fully participating in various fields of work and education. The research papers aim to comprehensively understand the obstacles faced by individuals with disabilities and the progress made in promoting inclusive and accessible experiences across different contexts by incorporating their insights. Furthermore, the retrieved and analyzed web accessibility researches heavily focus on the visual disability type.

Issues have been detected for different web tool components, such as limitations in accessing or creating content, collaborating, or using assistive technologies. The proposed solutions mainly focus on providing additional tools to create accessible content, plugins, extensions, or concepts to integrate into existing web products. These alternatives focus on different senses, such that people who have difficulties using one of their senses can use and access web content through other senses. Therefore, providing alternatives and transforming web content into a different format is also a common approach to ensure web accessibility in conceptual modeling. The key finding of providing accessible websites or tools is to structure those properly and involve experienced designers and tool creators. Further, they should be compatible with assistive technologies to support individuals with disabilities adequately.

In conceptual modeling, the research primarily addresses visual impairments concerning UML diagrams. Proposed solutions include using audio, converting graphics into text, and creating physical aids for accessing and interacting with diagrams. However, there is a gap in applying these solutions to existing web-based modeling editors. Further research is needed to integrate accessibility features into widely used modeling tools, enabling inclusive collaboration and reducing participant workload. Developing efficient and user-friendly solutions specifically for heavily graphical and mouse-based modeling editors is not only important for visually impaired individuals. Enhancing design concepts is crucial for improving interaction and collaboration within web modeling tools.

The overall gained insights and knowledge about state-of-art research have shown a clear

literature and research gap regarding the remaining disability types (cf. Table 2.3). As a result, the solution proposals are similar and build up on each other but do not or only slightly benefit users with other disabilities. Especially looking at web accessibility research in conceptual modeling, it is observable that currently, the possible accessibility enhancements and the provided approaches are very scarce, as modeling and the usage of modeling editors are still challenging due to the use of graphical notations and heavily usage of pointing devices and keyboard in online modeling editors. Furthermore, providing a collaborative environment is more significant in conceptual modeling as these models or diagrams are used to communicate amongst various professions.

In conclusion, all of these papers emphasize the shared goal of promoting web accessibility through testing methodologies and the implementation of guidelines in content creation tools. The research presented in these papers provides valuable insights into ensuring accessible digital content and can serve as a guide for developers, designers, and content creators in improving web accessibility and fostering inclusive online experiences. However, there is a need for literature and research addressing the remaining disability types, which primarily focus on inclusivity in conceptual modeling by providing appropriate solution approaches for accessible modeling environments.

CHAPTER 4

Disability-Awareness in Web Modeling Tools

4.1 Introduction to Web Barriers: Understanding the Challenges

Web modeling tools have gained widespread popularity due to their numerous advantages. These tools facilitate faster and easier creation, presentation, refinement, and styling of various types of models and diagrams. Moreover, their ability to enable efficient collaborative work is a key highlight [PJ15]. To ensure inclusivity and accessibility, it is essential to design these modeling tools in a way that allows users with disabilities to utilize their functionalities seamlessly. By prioritizing accessibility in the design process, these tools can provide benefits to all users, regardless of their impairments or disabilities.

Therefore, this chapter aims to provide an in-depth evaluation of ten well-known web-based modeling tools with respect to the extent of their support for users with *Visual*, *Cognitive, Learning, and Neurological*, and *Physical* disabilities (cf. Table 2.3). In this case, both other disability types, *Auditory* and *Speech* are not covered, as both of these are not directly influencing the usage of web-based modeling tools, i.e., usually there are no audio or media contents nor only speech driven usage.

Nowadays, many web modeling tools are available, making evaluating each separately impractical. To address this, a curated list from the modeling languages community [1] was used to select the tools for this thesis' evaluation. Most of these tools are focused on UML due to its widespread usage, but many of these tools now also offer additional support for various other diagram types.

[1] Top online UML modeling tools, `https://modeling-languages.com/web-based-modeling-tools-uml-er-bpmn/`, (Access: 18.08.2023)

4.1.1 Key Challenges Across Assessed Disability Types

In this thesis, the evaluation criteria for web modeling tools are based on analyzing the barriers and limitations in the web for disabled or impaired individuals, as provided by the Web Accessibility Initiative (WAI) [Web17a]. Before delving into the criteria, the following section will discuss the challenges and limitations disabled users commonly encounter while using web tools or accessing web content. The comprehensive list compiled by WAI explains the reasons why certain web elements or components may present difficulties or hinder usage, as well as the types of support or alternatives needed to provide in order to enhance accessibility and foster inclusivity of web components. It is essential to note that there may be additional problematic issues and barriers for disabled users. However, the provided list of barriers primarily focuses on the most common barriers, which can also be relevant in the context of modeling tools due to the editors' structure and components. These descriptions serve as the foundation for the evaluation criteria outlined in Section 4.2.6.

The upcoming sections are structured as follows: first, a short introduction to the corresponding disability type will be provided, followed by an overview of the common barriers and problems people with these disabilities encounter while using the web. Lastly, the suggested improvements from WAI will be presented.

Visual Disability Support

The definition of visual disability can be found in the previous Table 2.3. Visual disabilities can vary in severity and acuity levels, encompassing conditions such as color blindness (which limits the perception of specific colors, such as red-green blindness or difficulty with all colors), partial sight or low vision, and complete blindness, which refers to the uncorrectable loss of vision in both eyes.

According to WAI, individuals with visual disabilities encounter challenges in how web content is presented to them. To provide appropriate support for this disability type, it is essential to offer different forms of web content presentation that can be easily customized and adapted to users' specific needs. This approach ensures inclusivity and accessibility for individuals with visual impairments or disabilities. A detailed list of evaluation criteria specifically addressing visual disabilities and impairments can be found in Section 4.2.6.

1. **Barrier: Web components that do not have equivalent text alternatives:** Visual disabilities can cause challenges in observing or reading text and elements on web pages, affecting usability significantly. Missing or insufficient alternative texts for images are also widespread, as reported by WebAIM. The 2023 report discovered that over one-third of images on tested popular web pages lacked proper alternative text [Web23]. This situation is crucial as web creators increasingly use images, symbols, and non-text elements on their pages. In just one year, the

number of images in the sample increased from 39 million in 2022 [Web22a] to 43 million in 2023 [Web23].

- **Suggested improvement: Text alternatives:** In order to meet the needs of users who rely on text-to-speech synthesis, providing textual alternatives is crucial. Alternative text enables users to understand the content of images or non-textual elements that they may not be able to perceive or interact with otherwise. Assistive technologies like text-to-speech tools can capture and convey this alternative text. Most popular web browsers nowadays [2] offer a wide range of text-to-speech tools, enabling users to access web content audibly. For example, Google Chrome's Web Store has approximately 160 text-to-speech extensions [3]. However, for these tools to be practical, web pages must be created with appropriate text alternatives. Web pages that heavily rely on images, symbols, or non-textual components without adequate text alternatives may become inaccessible to visually disabled users. This can make the page difficult or even impossible for them to use as intended.

2. **Barrier: Pages that cannot be resized, or that lose information when resized:** Some users may find the default sizes of web content unsuitable and need to adjust the size of different elements for a better user experience or to perceive them correctly. Elements that are not resizable based on user needs become a limitation. If resizing causes the web page to lose information or become unusable, it significantly impacts the satisfactory usage of visually disabled users.

   - **Suggested improvement: Customized enlarging or reducing text size and images**: Nowadays, an essential property of web pages is to allow the user to zoom in or out on the web page. Furthermore, the font size should also be easily adaptable. To achieve this goal, the presentation of web content must be independent of its structure, and the structure should be suitable for processing and presenting it differently depending on the different browsers and used assistive technologies. In that particular case, relying solely on default web browser zooming may result in information loss or make the page unusable. In conclusion, the best way to provide correct and usable sizing is to provide the user with customizability suited for the content of the particular web page.

3. **Barrier: Unsuitable colors and insufficient contrast in web pages:** This barrier is particularly relevant for users with color blindness or limited visual field, as fixed color schemes and insufficient contrast can make it challenging for them to

[2] Usage share of web browsers, `https://en.wikipedia.org/wiki/Usage_share_of_web_browsers`, (Access: 29.04.2023)

[3] Chrome Web Store, Type: Extensions, Category: Accessibility, Search string: text to speech, `https://chrome.google.com/webstore/search/text%20to%20speech?hl=de&_category=ext/22-accessibility`, (Access: 29.04.2023)

perceive text and images on the page. The WebAIM reports of 2022 [Web22a] and 2023 [Web23] revealed that a significant percentage of web pages (31.7% in 2022 and 30.4% in 2023) had instances of low-contrast text, indicating that many websites still offer inadequate contrast by default. Individuals with difficulty distinguishing specific colors or perceiving them differently than someone with normal color vision may find web pages with fixed, non-adaptable color schemes unsuitable for their needs.

- **Suggested improvement: Customizable font and colors:** In order to ensure content readability for individuals with difficulties perceiving specific color combinations, fonts, or contrast, web pages and applications should offer the option to customize fonts and colors. This may also involve providing different color schemes, such as dark and light modes, to offer users ready-to-use designs according to their preferences and needs.

Cognitive, Learning, and Neurological Support

The definition of cognitive, learning, and neurological disability (abbrev. CLN) can be found in Table 2.3. Cognitive, learning, and neurological disabilities encompass a diverse range of conditions, including neurodiversity disorders, neurological diseases, and behavioral and mental health disorders. These conditions can impact an individual's communication, learning, and interaction with their environment in various ways [Web17b].

According to WAI, for people with cognitive, learning, and neurological disabilities, web content can pose challenges if it is complex, poorly structured, or contains excessive animations and interactive elements that lead to cognitive overload. In order to enhance accessibility for these users, it is crucial to provide a clear and easy-to-follow structure, reduce interactivity, and avoid distractions such as blinking or flashing elements or provide customizable settings for disabling these kinds of functionalities. For a detailed list of evaluation criteria specifically addressing cognitive, learning, and neurological disabilities, refer to Section 4.2.6.

1. **Barrier: Lack of clear content structure for overview and orientation:** Individuals with cognitive disabilities may face challenges when navigating and comprehending websites without proper organization and structure. Complex navigation mechanisms and page layouts can pose usability problems for disabled and non-disabled users, making web applications difficult to use or even inaccessible for some individuals with disabilities. Unlike barriers related to colors or fonts, it is not easy to add additional options which can be enabled to provide a more intuitive and simplified structure of the website itself. This barrier also encompasses issues with unpredictable link targets, functionality, and overall interaction. The unsuitable or complex structure of web components can lead to unexpected behaviors, causing frustration and difficulty completing tasks. Inconsistent labeling of forms, buttons,

and other elements further compounds the problem, potentially causing confusion and hindering users' ability to understand and interact with the content efficiently. Addressing these issues is crucial to ensure a more inclusive and accessible online experience for individuals with cognitive disabilities.

- **Suggested improvement: Clearly structured content and overview:** To improve the accessibility of web content for individuals with cognitive disabilities, careful planning and clear design are essential. Organizing the website's content in a structured manner, with well-defined groupings and a clear hierarchy, helps users easily comprehend the relationships between different sections. This should be supported by descriptive headings and subheadings, which clearly label sections, provide users with a quick overview of the content, and enable them to navigate directly to the desired sections.
- **Suggested improvement: Uniform Pattern, Textual & Visual Aids:** In order to address issues related to unpredictable link targets and functionality, it is crucial to ensure that links are clearly labeled with descriptive text, accurately reflecting their destination. Consistent interaction patterns should be maintained across the website, ensuring that similar elements behave uniformly across different pages or sections. Providing visual or auditory feedback when users interact with links or elements confirms that their actions have been recognized, enhancing predictability and reducing uncertainty in the overall interaction. However, excessive feedback should be avoided, and users should be able to turn off sounds, animations, or motions if applicable.
- **Suggested improvement: Consistent labeling of forms, buttons, and other elements:** To avoid inconsistent labeling of forms, buttons, and other components the usage of labeling conventions and guidelines should be considered to ensure consistent and meaningful labels for forms, buttons, and other interactive elements throughout the website.

2. **Barrier: Complex or text-heavy web content:** Complex sentences with uncommon or hard-to-read words, lengthy text without visual aids like images or graphs, and unstructured or unhighlighted text can present challenges for some users. This overly complicated textual content may hinder disabled users from effectively using the website or accessing its content. The cognitive load imposed by such distractions can make it challenging for users, particularly those with cognitive or learning disabilities, to comprehend the provided textual content.

   - **Suggested improvement: Avoid text-heavy content:** In order to address this barrier, a straightforward solution is to avoid lengthy and text-heavy passages by splitting or rewriting them. Using more common synonyms and highlighting essential text parts can also improve comprehension and reduce reading time.

- **Suggested improvement: Supplementary visual aids:** When dealing with text-heavy content that cannot be avoided due to information loss or other reasons, it is essential to supplement the textual content with suitable visual aids. Illustrations, graphs, or images can enhance understanding and assist users who struggle to gather information from the text alone. These visual aids provide additional support and improve accessibility for a broader audience.

3. **Barrier: Usage of different audio, motions, animation and flickering content:** For individuals with cognitive or neurological disabilities, excessive animations, blinking, flashing elements, and background audio can lead to cognitive overload and distraction. Websites that automatically enable these features without the option to deactivate them present a significant barrier for such users.

   - **Suggested improvement: Options to deactivate distracting audios, animations and similar:** Websites with animations, blinking, flashing elements, or background audio should provide an easy possibility to disable or control those multimedia to improve accessibility and reduce cognitive overload.

Physical Support

The definition of physical disability, also known as motor disabilities, can be found in Table 2.3. Physical disabilities encompass a broad spectrum of dysfunctions, ranging from weakness and limitations in muscular control and sensation to joint disorders, pain that impedes movement, and even missing limbs. These conditions can affect an individual's ability to interact with the web content, particularly in terms of reaction time and insufficient or non-existing support for using hardware, such as a mouse, mouse-like, keyboard or keyboard-like devices, which are mostly used the ability of typing, click and therefore use the websites as intended [Web17c].

According to WAI, the primary issue for physically disabled users interacting with websites is insufficient support for commonly used pointing or input devices and similar assistive technologies. Additionally, some users may experience difficulties with their reaction time, requiring more time to complete tasks or respond. Moreover, web components that require precise handling and rely on fine motor control may pose challenges for some physically disabled individuals, making it difficult for them to interact with these components effectively. For a detailed list of evaluation criteria specifically addressing physical disabilities, refer to Section 4.2.6.

1. **Barrier: Missing full keyboard support:** One of the essential features relevant for most users with physical disabilities or impairments is full-keyboard support. People with disabilities who rely on keyboard navigation face significant challenges when websites do not provide full keyboard support. The main struggle here is to

access and interact with web content effectively, encountering barriers that hinder their ability to navigate, input information, and activate functionality. Without comprehensive keyboard support, individuals with disabilities may be excluded from fully engaging with online resources and services.

- **Suggested improvement: Full-Keyboard Support:** The suggested improvement to address this barrier is offering full-keyboard support. This means ensuring that all interactive elements, menus, and controls can be accessed and operated using only keyboard input [4]. Users should be able to navigate the website and access all functionalities without a mouse or other pointing device. This can be achieved by implementing meaningful keyboard shortcuts that execute the intended functionalities effectively. Following the keyboard accessibility standards provided by the Web Content Accessibility Guideline (WCAG) 2.1 can be a way to introduce valuable and satisfactory keyboard support [Web18]. The keyboard support should complement the usage of mouse or mouse-like devices. Furthermore, providing appropriate key alternatives, especially for a website's most essential or frequently used functionalities, can enhance usability for disabled and non-disabled users who prefer using the keyboard for efficient work. Clear and easy-to-access documentation or instructions should be available to guide keyboard shortcuts, access keys, and any unique navigation features.

2. **Barrier: Insufficient time limits to complete tasks:** The second significant barrier is the presence of insufficient time limits for users to complete tasks or respond. This creates obstacles, especially for individuals with physical or also cognitive disabilities who may require additional time. Without adequate time, website operations cannot be accomplished successfully, and tasks may remain unfinished. Moreover, inadequate time limits can cause stress, rushed decision-making, and exclude these individuals from accessing and engaging with web content effectively. Providing sufficient time for users to complete tasks ensures inclusivity and allows for equal participation.

   - **Suggested improvement: Extend reaction time limits:** In order to avoid user dissatisfaction, it is essential to remember that people with physical disabilities may require more time to type, click, or carry out other interactions. The web interfaces should accommodate slower input and provide sufficient time for users to complete tasks, i.e., there should be no interaction that stresses the user to finish some task or to respond in a specific time manner. This means the interactions or responses should not have a time limit or fail if the user's reaction is slower than expected. If avoiding time limits is not an option, another solution would be to provide users with adjustable or extended time limits. Additionally, including clear instructions and progress indicators

[4] Keyboard Accessibility, `https://webaim.org/techniques/keyboard/`, (Access: 26.07.2023)

can help users understand the time available and the task completion status, enabling them to manage their time effectively.

3. **Barrier: Missing, unpredictable, or overly complicated navigation or visual indicators:** Websites must provide visible indicators of the current focus, such as highlighting or underlining. This lack of clear indicators and cues makes it difficult for users to understand the layout and organization of the website, leading to confusion and navigational challenges. Additionally, users with physical disabilities should be able to use effective navigational aids to facilitate easier navigation through the web content. In order to achieve this, some mechanisms should exist to skip over repetitive blocks, like page headers or navigation bars, which are also essential for efficient browsing. It is worth mentioning that visual indicator cues and navigation algorithms which are practical and easy to access for disabled users will improve user satisfaction and efficiency in using the web application as intended.

   - **Suggested improvement: Navigation & visual cues or indicators:** In order to overcome this barrier, the web application should implement navigation algorithms suitable for the context and purpose of use. Clear and consistent navigation is crucial, achieved by consistently displaying navigation menus and buttons across pages. Providing clear labels and visual cues, such as highlighting the current page or section, helps users understand their location within the website. Efficient navigation through web components, skipping over blocks, or reaching specific parts of the website should be facilitated. However, effective navigation relies on a well-structured page layout. Pages should be organized logically, with clear hierarchies, and supported by appropriate headings and labels. This barrier also relates to other accessibility aspects, such as providing text alternatives for descriptive web components and ensuring keyboard accessibility. Visual and non-visual orientation cues are essential to show users their current focus, available navigation options, and their current position during navigation.

4. **Barrier: Clickable areas relying on precision & typing mistakes:** This barrier refers to the challenges faced by people with physical disabilities when clicking on small areas or typing accurately. These actions require precise fine motor skills, resulting in slower reaction times, more errors, or difficulties in completing tasks as intended.

   - **Suggested improvement: Large clickable areas & error correction options:** For this barrier, the most straightforward solution is to provide large clickable or typable web components to avoid high precision control by the user. If providing large enough components is impossible, e.g., due to limited screen sizes or similar, some alternatives should be provided, e.g., a keyboard shortcut alternative, which does not require precise handling with pointing devices and

is less error-prone. In addition, the error correction option, i.e., re-clicking and re-typing, should be possible for all provided interactive elements and should not be designed in an overly complicated manner.

4.2 Evaluation Framework

This section describes the evaluation framework in detail. The evaluation framework includes all the essential key elements and processes that guide the overall assessment process. The presented road map, in the following, ensures that the evaluation is systematic, comprehensive, and aligned with its intended objectives.

4.2.1 Evaluation Purpose

Context, Objectives and Scope: This assessment provides insights into the current accessibility status of commonly used web modeling tools. This evaluation offers an initial introduction to the topic, revealing undiscovered gaps, barriers, or limitations faced by disabled modelers when using web tools. It is crucial to ensure that these tools are inclusive and provide equal access for individuals with disabilities so that each can effectively engage with and benefit from them.

The goal is to evaluate the compliance of these tools with accessibility standards, their compatibility with assistive technologies, and their overall usability for individuals with disabilities. This intention is achieved by addressing specific accessibility aspects for different disability types, such as *visual*, *physical*, *cognitive, learning, and neurological* accessibility. A detailed description of the evaluation's fundamental aspects, including the applied evaluation criteria, can be found in each of the following sections. While the evaluation focuses on a specific set of modeling tools, it provides valuable insights into the broader challenge of accessibility in web modeling tools.

Expected outcomes: This evaluation is expected to identify the initial status of accessibility strengths and weaknesses of the evaluated tools and provide recommendations for enhancing their accessibility. The evaluation findings will raise awareness about accessibility in conceptual modeling and offer actionable insights for developers and designers to make necessary improvements. The systematic literature review (cf. Chapter 2) revealed a lack of research for the disability types *Physical*, *Cognitive, Learning and Neurological*. Consequently, this tool evaluation's scope includes these disability types.

The ultimate goal is to develop a **prototype** (cf. Chapter 5) that addresses the weaknesses identified in the assessment and fills the research gaps identified in earlier chapters (cf. Chapter 2). This initial assessment provides essential insights for defining the following steps and identifying accessibility issues in conceptual modeling. The findings will be valuable for stakeholders, including tool developers, designers, researchers, and users.

4.2.2 Evaluated Tools

Table 4.1 shows the web modeling tools used for this evaluation. Further details about the used plans and version and their references can be found in **Appendix 2: Commercial Software and Tools**. To sum up, the free plans or trial versions were used for each of the tools if the tools were not freely accessible. The evaluation is conducted on the web browser Google Chrome. The intended use of the evaluated tools is for creating and sharing diagrams and visual representations of information in various types.

Table 4.1: Evaluated Web Modeling Tools

Tool	**Company**
Lucidchart	Lucid Software
GenMyModel	Axellience
Gliffy	Perforce
diagrams.net	JGraph Ltd.
Creately	Cinergix Pty. Ltd.
Cacoo	Nulab Inc.
UMLetino	Open-Source Community
Diagramo	Diagramo Ltd.
miro	RealtimeBoard Inc.
BPMN.io	Camunda Services GmbH

4.2.3 Evaluation Questions

The upcoming section introduces evaluation questions designed to guide the assessment of modeling tools' compliance with accessibility standards. These questions cover various aspects, including common barriers and limitations faced by disabled users, compatibility with assistive technologies, and the overall user experience for individuals with disabilities. By addressing specific aspects related to *visual*, *physical*, *cognitive, learning, and neurological disabilities* (also referred to as *assessed disability types* in the upcoming sections), these questions seek to gather insights for assessing and identifying gaps, barriers, and limitations that different user groups may encounter.

- **EQ-1: Which accessibility barriers and limitations for people with the assessed disability types [5] do exist *most* frequently that would block or complicate the usage of the tools in the intended way?** This particular evaluation question helps to highlight those criteria which are not satisfactory or not at all satisfied by the majority of the evaluated tools. The gathered information will help discover the most significant gaps in common web modeling tools and present the properties or functionalities that should be taken care of.

[5] includes *visual*, *physical*, *cognitive, learning, and neurological disabilities*

- **EQ-2: Which accessibility barriers and limitations for people with the assessed disability types [5] do exist *least* frequently that would block or complicate the usage of the tools in the intended way?** This evaluation question forms the second part of the previously described **EQ-1**, and it is used to determine those criteria which are satisfactorily fulfilled by the majority of the evaluated tools. The gathered information will help to find out the functionalities or alternative ways in which most of the commonly used web modeling tools managed to provide and eliminate some particular accessibility issues.

- **EQ-3: Which accessibility functionalities do the evaluated tools provide?** This question aims to determine if and which additional functionalities each tool offers to improve accessibility. While the assessment using the evaluation criteria helps to identify if certain disability types are being supported, this particular question is used to determine which accessibility functionalities are included and how they work.

4.2.4 Evaluation Methodology

The evaluation has been conducted through *Observations* and *Experiments*. Each of the selected web modeling tools was evaluated according to the evaluation criteria for the assessed disability types [5] by checking if the given condition is **satisfactorily fulfilled**, **partially**, or **not fulfilled**. A single evaluator evaluated the web modeling tools. The given set of tools is evaluated against the criteria described in Section 4.2.6 and in the guidance of the evaluation questions described in Section 4.2.3. The outcomes are documented based exclusively on the judgment and expertise of the evaluator.

Barriers and Limitations in the Evaluation Process: The reliability of the evaluation is not affected by the single evaluator, as even if the evaluation had been conducted with a group of disabled people, it would include subjective bias, as every person's disability is unique, and how they are affected by their limitations and barriers would influence the outcomes. Subjective bias can be minimized through careful planning, training, and transparency. A certain degree of bias is inevitable, but steps can be taken to mitigate its impact and ensure the analysis is as objective as possible. Nevertheless, it is crucial to establish clear and well-defined evaluation criteria using guidance and transparency to minimize subjective bias through individual interpretation. Supporting all possible combinations of disabilities, including their varying severity, would be challenging due to their diversity, and it may hinder finding a more general solution that would support a more extensive user base. As a result, the assessment criteria are organized based on specific disability types. However, it is worth noting that providing support for specific disability groups can also benefit individuals who experience multiple disabilities.

4.2.5 Evaluation Process

The following steps are followed for each of the selected web modeling tools (cf. Table 4.1) in order to accomplish the tool evaluation:

Step 1: Preparation of the evaluation components

The first step of the evaluation process is preparing the evaluation criteria. As multiple tools are compared, it can be challenging to keep track of all the relevant criteria and ensure that each evaluated tool is handled objectively. The evaluation criteria are based on objective evidence and standards by the Web Accessibility Initiative (WAI). The WAI has explored and researched the vast diversity of people and their abilities for many years and published this valuable knowledge as a guide. This guide presents some web accessibility barriers that people commonly experience because of inaccessible websites and web tools for each of the classified disability types (cf. Table 2.3). More specifically, these assessment criteria offer insights into the barriers and limitations that users with specific disability types typically encounter while using the web. Additionally, they suggest features and solutions that can enhance accessibility and make the web content or tool more inclusive for these users.

These valuable insights and information are used and adapted as evaluation criteria for this tool evaluation. A detailed overview of the applied evaluation criteria categorized by the disability types can be found in Section 4.2.6. Moreover, as not every evaluation criterion can be manually checked, additional software, primarily browser extensions, are used to automate the assessment objectively. The references and further details about the used browser extensions can be found in **Appendix 3: Additional support tools and extensions**. The overview in Section 4.2.6 describes how each extension was applied and for which evaluation criteria it was used.

This systematic comparison provides this thesis with a consistent evaluation, making it easier to compare and contrast each modeling tool. In addition to that, the support from different web extensions and the guidance with the defined evaluation questions (cf. Section 4.2.3) completed this evaluation.

Step 2: Conducting the evaluation

The following paragraphs describe the steps applied for each assessed web modeling tool.

To establish a consistent understanding, the following section describes the features and components of a web modeling tool that are considered **significant for operation** within the context of this evaluation.

- **Creating & Editing Models:** The most essential feature of web modeling tools is the ability to create and edit models. These tools allow users to create new models from scratch or use pre-designed templates. Users should be able to add,

delete, and modify various model elements, such as shapes, icons, connectors, and text boxes.

- **Connecting & Linking Elements:** Connecting elements in a model to represent relationships or dependencies is essential. Therefore, it should be easy for users to draw connectors and lines and to establish connections.
- **Resizing & Scaling:** This category comprises functionalities that enable users to resize modeling elements, including shapes, icons, and text boxes. It also includes the ability to resize the entire model or specific parts, which can be beneficial for managing large or complex diagrams.
- **Grouping & Nesting:** This category describes the feature to allow users to group related elements, allowing them to treat the group as a single entity, e.g., via nesting or other possibilities for hierarchical structuring of components.
- **Layout & Alignment:** This feature describes the operations provided for different layout and alignment possibilities, depending on the context and user needs. For example, users should be able to arrange elements on the canvas using predefined layouts or by manually adjusting element positions.
- **Searching & Navigating:** This feature includes the essential functionality of allowing users to search for specific elements, text, or attributes within the model. Additionally, the navigation functionality should support users in efficiently exploring the tool and the model.
- **Diverse Tool Support:** This includes all features essential for supporting the user in creating, editing, sharing, or collaborating with the models. Some examples of these functionalities are importing or exporting, printing, sharing, publishing, and collaborating in real time.

1.) Check conditions of the evaluation criteria: As described in Section 4.2.6, for each of the disability types, the conditions were checked for fulfillment. Depending on the criteria and condition, the outcome would be that specific conditions are either **satisfactorily fulfilled**, **partially**, or **not fulfilled**. If applicable, this step is also supported by additional extensions (cf. Section 4.2.6). This step consists of two subphases. Each assessment criteria is applied against (i) the *Tool support & Graphical User Interface* and (ii) the *Canvas & Model* of the modeling tool.

- **Evaluation of the Tool Support & GUI:** In this phase, each criterion will be evaluated against the provided support by the tool and its graphical user interface. Generally, these kinds of tools' GUI exist out of a menu header, footer, and different side or panel menus, including different types of interactions. The evaluation results for each tool can be observed in the first column (Column TG - Tool Support & GUI) of Table 4.17. The fulfillment of the evaluation criteria (cf. Section 4.2.6) are

assessed manually and supported by browser extensions for some of the criteria as explained in Section 4.2.6.

Below is a list of components and functionalities assessed during the evaluation phase of *Tool Support & GUI*. For each of the evaluation subjects and the applied evaluation criteria, the following elements and their provided functionalities, typical for web modeling tools were observed from the point of view of *Tool Support & GUI*.

It is important to note that the descriptions provided below are specific to the assessed web modeling tools and may not apply universally to every other web modeling tool.

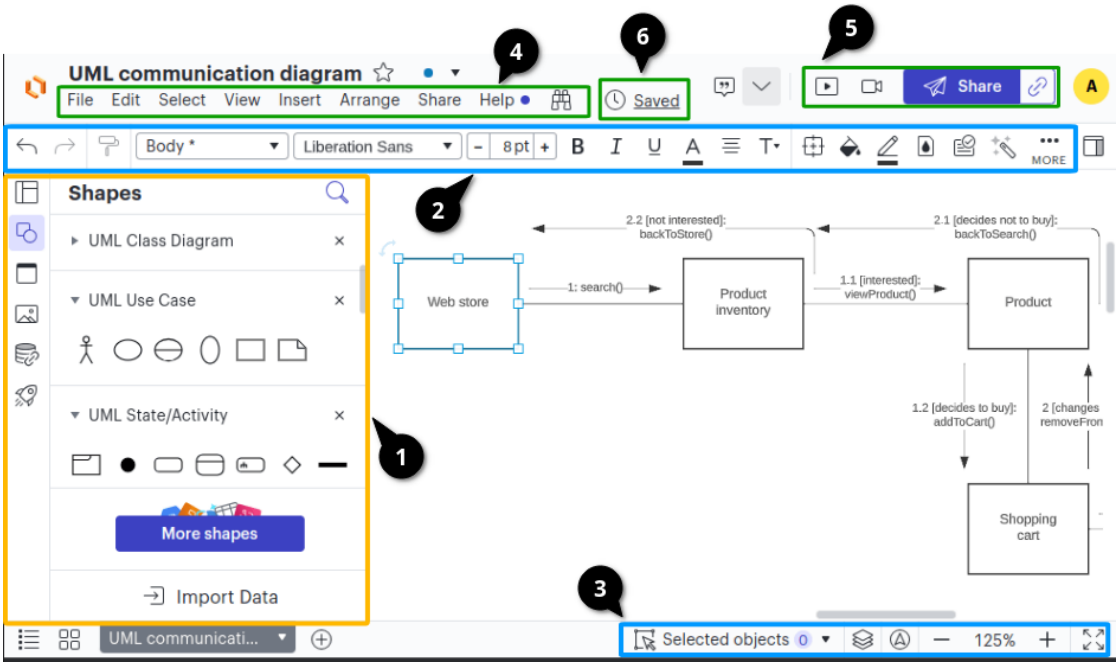

Figure 4.1: Example Snippet of Lucidchart for *Tool Support & GUI* elements

– **Diagram Creation:** The ability to create various types of diagrams and models is the primary purpose of web modeling applications. Depending on the domain, the number of supported diagrams will vary. From the perspective of *Tool Support & GUI*, the diagram creation generally includes the following web elements and components:

  * **Shapes, Symbols, and Labeling:** The essential need for modeling is to provide various shapes and symbols according to the supported diagram types. The web modeling tools mainly provide a collection of pre-defined shapes, symbols, connector lines, and similar in the form of a separate, structured menu, generally provided as **side panels** (example snippet is

marked as **1** in Figure 4.1). Often *Drag & drop* functionality is provided for adding elements to the canvas.

* **Pre-Defined Templates:** Some web modeling tools provide pre-defined shapes, symbols, connectors, or whole pre-defined model templates. Generally, these are structured as libraries and can be accessed directly through the web application. This menu option is a part of the previously mentioned **side panel** where the other pre-defined diagram elements can also be accessed or found at the start of a new diagram project (Pre-Designed Templates in Lucidchart can be found in Figure 4.2).

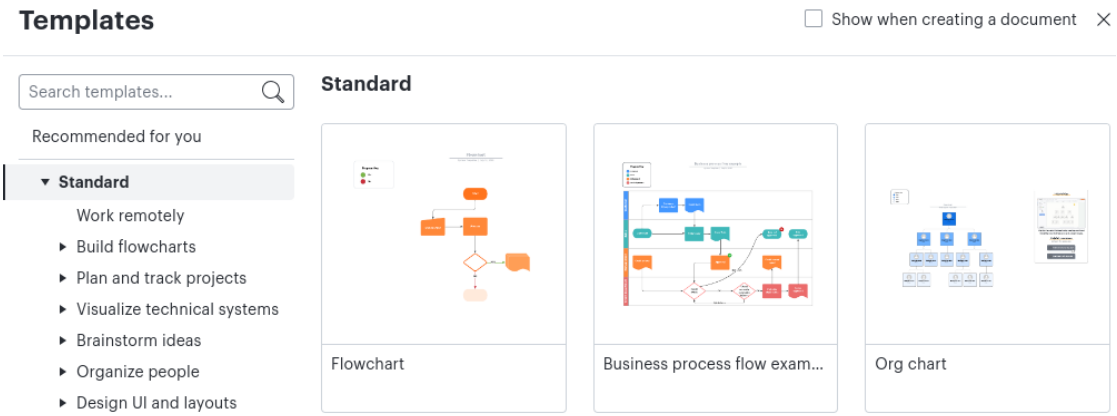

Figure 4.2: Snippet from Lucidchart: Template Gallery

– **Diagram Editing and Formatting:** Equally important to the creation of diagrams is the ability to adapt and modify them. For this purpose, the web modeling tools offer various options for editing and formatting diagram elements. The main functionalities for editing and formatting include the following:

  * **Formatting:** The formatting option includes manipulating font-related issues, such as font types, sizes, and colors. This feature is generally provided in the form of a menu bar in the **header** or as part of the menu, which provides the shapes and symbols as described previously in the functionality *Diagram Creation - Shapes, Symbols, and Labeling* (example snippet is marked as **2** in Figure 4.1).
  * **View:** The view-related options allow different editing actions with the diagram, such as resizing, rotating, aligning, distributing, grouping, and arranging objects. This setting is mainly located in the **footer** menu of the corresponding web modeling tool. It often includes options like zooming, page layout adaptions, adding new pages, and using rulers, grids, or similar constructs. In the context of modeling tools, many of them also incorporate a mini-map feature. This mini-map gives users an overview

of the entire diagram and offers simplified navigation options (example snippet is marked as **3** in Figure 4.1).

– **General Tool Options:** This category includes interaction possibilities directly with the web tool itself, such as:
  * **Navigation Menu** that links to different sections or features of the application, such as Home, File Management, Diagrams, Settings, and Help (example snippet is marked as **4** in Figure 4.1).
  * **Collaboration and Sharing** is used for sharing diagrams with others, either by granting view-only access or allowing collaborators to edit. Users can share diagrams through links, email, or integration with collaboration platforms (example snippet is marked as **5** in Figure 4.1).
  * **Version Control and History** for tracking and managing different model versions, allowing users to revert to previous states, compare changes, and view the history of edits (example snippet is marked as **6** in Figure 4.1).
  * **Dialogs and Notifications** which are displayed for informational or guidance purposes and are triggered by user actions (example snippet can be found in Figure 4.3).

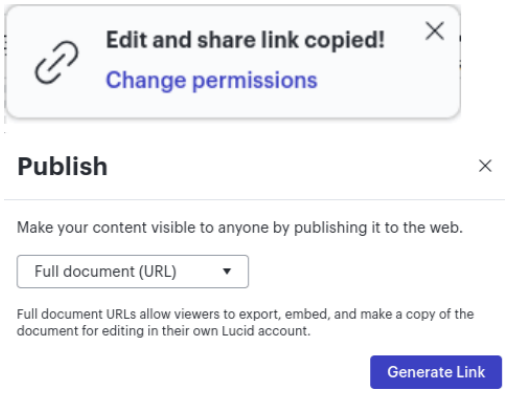

Figure 4.3: Example Snippet of Lucidchart: Dialog and Notification

- **Evaluation of Canvas & Model:** In the second step, the evaluation is focused only on the interaction between the canvas and the model. Since the tools can handle various diagram types and modeling languages, a default workflow is established to ensure a consistent process for each tool, enhancing the comparability of the results.

  The outcomes for each tool can be observed in the second column (Column CM - Canvas & Model) of Table 4.17. The fulfillment of the evaluation criteria (cf. Section 4.2.6) is checked manually and supported by browser extensions for some of the criteria as explained in Section 4.2.6.

  The default workflow was about creating a simple test UML class diagram, if applicable, i.e., two test classes with properties and a relationship between them

with multiplicities as a starting point. An example of the test default workflow is illustrated in Figure 4.4. An exception was made for two web modeling tools, namely Miro, which only supports UML class diagrams in the premium version, so a diagram was created using similar shapes and relations. Secondly, Diagramo only supports UML state diagrams, so this was used instead. The basic workflow was to check the CRUD functionalities of the diagram, i.e., create a diagram with two shapes and a relation, update, delete, and transform as applicable for the respective evaluation criteria, e.g., adding more classes, attributes, operations, and relationships.

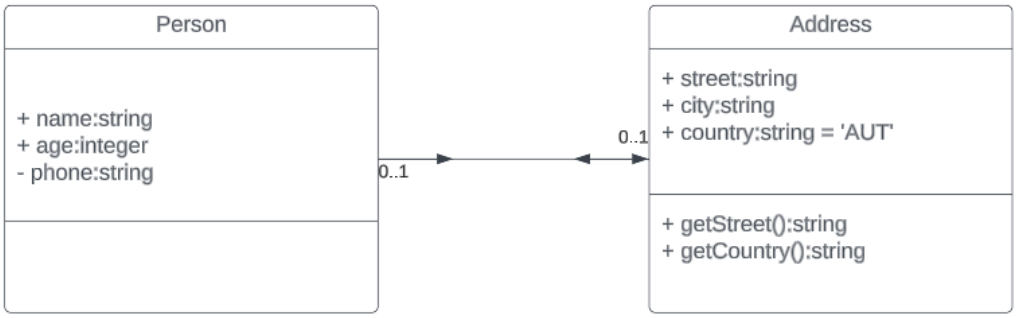

Figure 4.4: Example of default workflow UML diagram in Lucidchart

Similar to the phase *Tool Support & GUI*, a list is provided with all the components considered and assessed in this particular phase *Canvas & Model* while evaluating each modeling tool against the given criteria. This list and the descriptions of each component are created from the perspective of how, most commonly, the canvas and the models in web modeling tools are structured, organized and interacted with typically. The descriptions are prepared from the point of view of the assessed web modeling tools and may not apply universally to every other tool.

Generally, most web modeling tools provide an additional menu for directly creating or manipulating diagram elements, for example, for creating connectors between given shapes or adapting font settings of selected shapes and symbols. These actions only apply to the diagram or the selected elements themselves. These options are displayed either **as symbols on the canvas or on the diagram elements**. Additionally, a contextual menu is sometimes provided for further creational and editing options related to the canvas and model.

A categorized description of the assessed components can be found in the listing below:

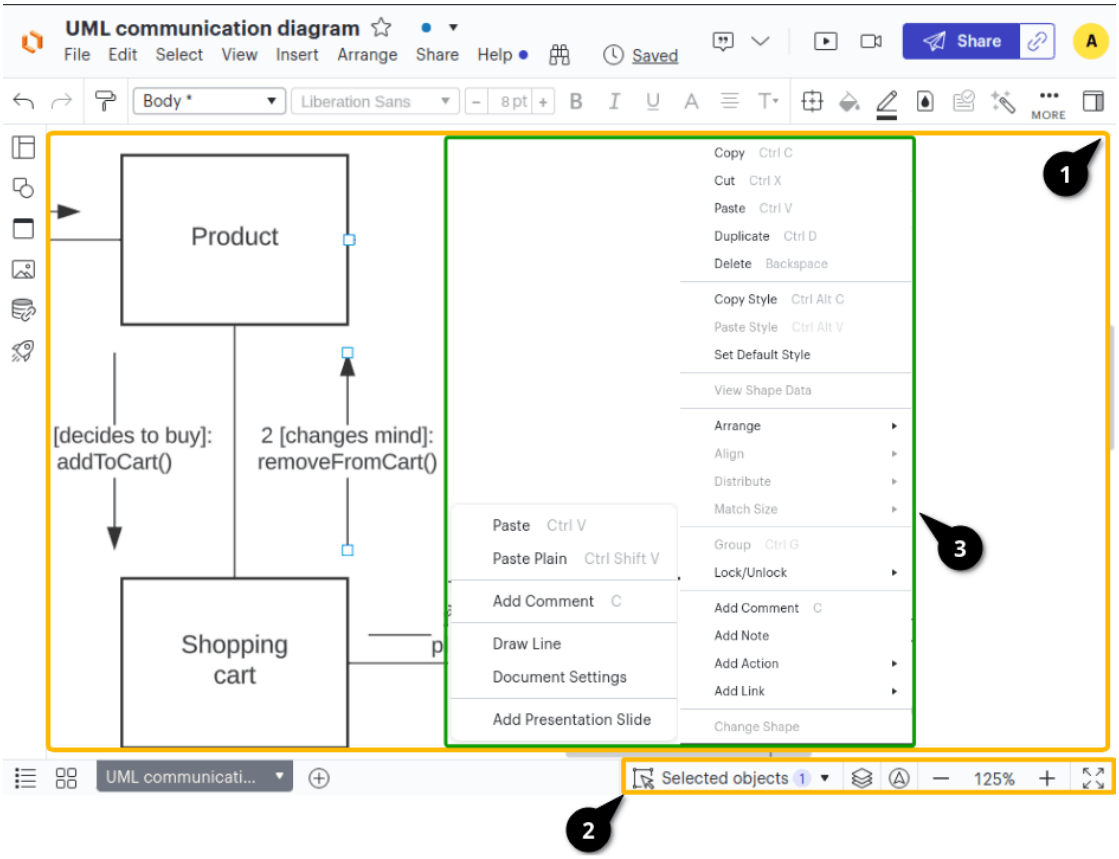

Figure 4.5: Example Snippet of Lucidchart for *Canvas & Model* elements

- **Canvas**: The canvas serves as the container for holding the diagram and its elements. It not only presents the created model but also offers various interaction possibilities, which are assessed too, as described below (example snippet is marked as **1** in Figure 4.5):
  - **Correct Representation** of the diagram and its components is essential and required.
  - **Interaction possibilities** are often offered through *contextual menus*, encompassing various canvas-related actions, such as adding new diagram components, comments, or adjusting settings (example snippet is marked as **3** in Figure 4.5).
  - **View Adaptions**, such as zooming in and out of the canvas to adjust the level of detail or panning, enables users to navigate the canvas and view different areas of the diagram (example snippet is marked as **2** in Figure 4.5).

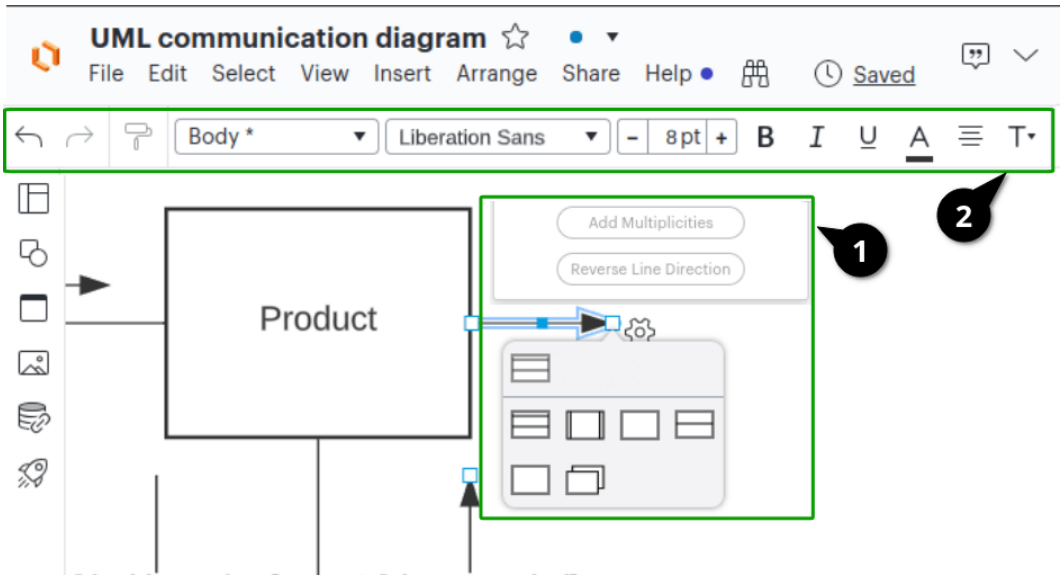

Figure 4.6: Example Snippet of Lucidchart: Canvas Interactions

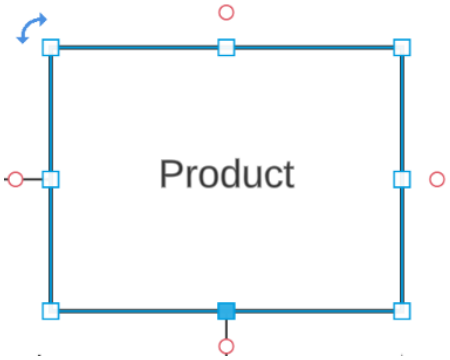

Figure 4.7: Example Snippet of Lucidchart: Canvas Element Adaptations with Rotation Arrow and Resizing Markers

- **Canvas Elements**: All elements that can be added to the canvas container are also considered part of the canvas assessment. These elements include shapes, symbols, connectors, labels, text boxes, and similar components that can be placed on the canvas. The canvas elements also often offer various interaction possibilities, which are assessed too, as described below:
  - **Interactions possibilities**, which include *contextual menus* for creating, deleting, extending, or editing different properties of marked elements (e.g., connectors, labels). In this case, the event handling possibilities include selection, dragging, resizing, and connecting elements (example snippet is marked as **1** in Figure 4.6).

  - **Adaption of the visual representation**, such as modifying element properties, e.g., colors, fonts, border styles, or line thickness(example snippet is marked as **2** in Figure 4.6). This adaption also includes shape adjustments such as resizing, orientation, and repositioning (example snippet can be found in Figure 4.7).

- **Dialogs, Tool-Tips & Notifications:** Some interaction possibilities can trigger dialogs and notifications or provide informational tool-tips if hovered over the canvas or its elements. These are mainly used for guidance, displaying errors, or providing other similar informational content and can be seen in various forms (example snippet can be found in Figure 4.8).

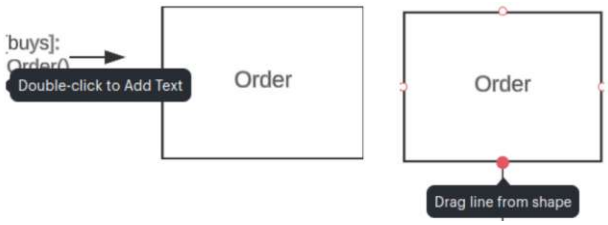

Figure 4.8: Example Snippet of Lucidchart: Tool-Tips

2.) Check evaluation questions: This step aims to gather the information needed for answering the previously defined evaluation questions (cf. Section 4.2.3). The data should be evaluated thoroughly to ensure that it is accurate and comprehensive. Finally, the results should be analyzed to answer the evaluation questions. The result can be found in Section 4.3.6.

3.) Check satisfaction of disability needs: After the information is gathered by applying the evaluation criteria and questions against the evaluation subjects, this step is required to sum up the findings. The main purpose of this task is to determine if a certain tool satisfies most of the disabled user needs for a specific disability type. This question can be answered by analyzing the fulfillment degree of each applied evaluation criterion. This outcome will help to state which tools do or do not support the evaluated disability types to a certain degree.

4.) Assess the level of accessibility satisfaction for each disability type: The results were determined using a scoring system, i.e., satisfactorily fulfilled criteria is worth 1 point, partially fulfilled 0.5, and not fulfilled is worth -1. The assessed web modeling tool

- **satisfies** the needs of a given disability type if the end score is positive.
- **partially satisfies** the needs of a given disability type if the end score is zero.
- **does not satisfy** the needs of a given disability type if the end score is negative.

This step is repeated for the three assessed disability types per the evaluated modeling tool. The assessment result will provide information about which tools are or are not capable of supporting users either with *visual*, *physical*, or *cognitive, learning and neurological* disabilities. The results can be observed in Table 4.17.

5.) Analyze and Interpret Results: In the final stage, the collected results and information will be interpreted using the pre-defined evaluation questions and discussed for future applications. As mentioned in the expected results section (cf. Section 4.2.1), these findings will highlight the accessibility gaps experienced by disabled or impaired users and identify the disability types that are not supported by existing tools. This information will be utilized to plan and design the accessibility prototype for this thesis.

The detailed interpretation of the assessment results according to the evaluation criteria and from the point of view of the assessed tools can be found in Section 4.3 and the overall conclusion in Section 4.4.

4.2.6 Evaluation Criteria

This section describes the fulfillment conditions of each evaluation criteria per disability type. As previously explained, each inspection of the criteria is divided into two phases with a different focus on the evaluated object, namely the *Tool Support & GUI* and *Canvas & Model* (cf. Section 4.2.5).

For better readability, the criteria descriptions provided in the following will be described and mentioned once but are applied to both categories *Tool Support & GUI* for operating and interacting with the tool and its provided features and *Canvas & Model* for operating and interacting with the canvas and the presented model and its elements.

The main difference in both assessment categories, *Tool Support & GUI* (in the following abbreviated as *TG*) and *Canvas & Model* (in the following abbreviated as *CM*), lies in how certain tasks of the evaluation criteria are conducted and which components of the assessed tools are evaluated.

Evaluation Criteria for Visual Disabilities

This section provides the evaluation criteria specifically for visual disabilities and impairments.

V1 - Customizing text and images size: This criterion checks if the possibility to enlarge or reduce text or image sizes according to the user's needs exists. Additionally, even if a resizing option is provided, changing text sizes, images, or other elements should not lead to information loss or misrepresentation (cf. Table 4.2).

V2 - Customizing fonts and colors: The colors and font impact the perception of specific visual impairments. This criterion describes and checks the possibility of customizing fonts and colors (cf. Table 4.3).

○ Not fulfilled ◐ Partially fulfilled ● Satisfactory fulfilled

Table 4.2: Fulfillment conditions for evaluation criteria V1

Condition & Fulfillment	
Every visible text, image, or other significant component of *TG* and *CM* is resizable without any loss of information or rendered inoperable parts. Resizability can be applied seamlessly to accommodate different screen sizes or user preferences. The interface maintains its functional integrity, ensuring that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational during and after resizing.	●
Only some visible texts, images, or other significant components of *TG* and *CM* are resizable without any loss of information or rendered inoperable parts. Some elements can only be resized via external support such as magnifying (browser) extensions or similar. The interface maintains some of its functional integrity, ensuring that the majority of its significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational during and after resizing.	◐
At least one of the visible texts, images, or other significant components of *TG* and *CM* are not resizable, lead to misrepresentation, or render the application inoperable even if used via magnifying (browser) extensions or similar. The interface does not maintain its functional integrity, ensuring that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational during and after resizing.	○

Table 4.3: Fulfillment conditions for evaluation criteria V2

Condition & Fulfillment	
The colors and font of every visible text, element, or tool component of *TG* and *CM* are customizable to the users' needs via designated setting options. The interface ensures that all significant operations (cf. Listing in Section 4.2.5) containing texts and labels are customizable regarding the appearance of text.	●
Only some of the colors and font of the visible text, element, or tool component of *TG* and *CM* are customizable to the users' needs via designated setting options, or achieving this requires external support such as color inverter, highlighting (browser) extensions or similar. The interface ensures that the majority of its significant operations (cf. Listing in Section 4.2.5) containing texts and labels are customizable regarding the appearance of text.	◐
At least one of the colors and font of the visible text, element, or tool component of *TG* and *CM* are not customizable to the users' needs via designated setting options and cannot be achieved via external support such as color inverter, highlighting (browser) extensions or similar. The interface does not ensure that all significant operations (cf. Listing in Section 4.2.5) containing texts and labels are customizable regarding the appearance of text.	○

V3 - Text-to-speech content synthesis: Not only the structure and customizability of visible web elements are essential. Especially for people with visual disabilities that impact both eyes (e.g., blindness), it is necessary to provide a structure that allows text-to-speech synthesis of the provided information or web page, which makes sense and allows these user groups to interact with the web content. This criterion handles whether visible elements are satisfactorily recognizable by text-to-speech applications and can create a practical text-to-speech synthesis of the content (cf. Table 4.4).

Table 4.4: Fulfillment conditions for evaluation criteria V3

Condition & Fulfillment	
The text-to-speech synthesis provided by the applied extension (cf. Section 4.2.6) includes every essential visible text or text alternative of images, symbols, and other non-textual components of *TG* and *CM*. The interface ensures that all significant operations (cf. Listing in Section 4.2.5) can be transformed into meaningful text-to-speech synthesis.	●
The text-to-speech synthesis provided by the applied extension (cf. Section 4.2.6) does not include at least one of the essential visible text or text alternative of images, symbols, and other non-textual components of *TG* and *CM*. The interface cannot ensure that all significant operations (cf. Listing in Section 4.2.5) can be transformed into meaningful text-to-speech synthesis.	○

Evaluation Criteria for Cognitive, Learning and Neurological disabilites

This section provides the evaluation criteria specifically for cognitive, learning and neurological disabilities and impairments.

CLN1-Clearly structured content: This criterion describes the need for a structure that facilitates overview and orientation. A structure should be established that allows users to gain an understanding of the content and quickly navigate their way through it. To achieve this, clear, descriptive, and concise labels or headings, a logical hierarchy, grouping related topics, and implementing search tools and filters should be included. Furthermore, a clear navigation system should be provided that allows users to access the different sections of the content easily. Additionally, images, videos, and other multimedia can be used to aid in understanding the content (cf. Table 4.5).

CLN2-Consistent labeling: This criterion emphasizes the importance of providing corresponding labels for forms, buttons, and other components. Clear and concise labels help users understand the content, avoid misinterpretations, and enable accessibility for text-to-speech readers. The labels should accurately describe the associated content, and their placement should be easily locatable for users (cf. Table 4.6).

CLN3-Predictable interaction: This criterion describes that the outcomes of user interactions should be predictable, i.e., should do what it has indicated. The website's content should be organized and presented to the user so that they can easily anticipate the results of their interactions based on the information and feedback provided. With the provided information, feedback, or similar, the user should be able to make decisions and also be able to understand the cause and effect of their actions based on their decision (cf. Table 4.7).

Table 4.5: Fulfillment conditions for evaluation criteria CLN1

Condition & Fulfillment	
The overall impression of the web application's content, including text, images, or other significant components for *TG* and *CM*, includes a clear and logical hierarchy with well-defined labels or headings. The related topics are appropriately grouped, and search tools and filters are implemented. A clear navigation system is provided, allowing users to access different content sections easily. Images, videos, and other multimedia elements are used effectively to enhance understanding. The interface maintains its functional integrity, ensuring that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational.	●
Some positions of the web application's content, including text, images, or other components significant for *TG* and *CM*, may be misleading, causing minor confusion in navigation or orientation. However, overall, the structure and organization of the content are still effective, and users can generally understand the content and navigate through it efficiently. The interface maintains some of its functional integrity, ensuring that the majority of the significant operations (see Listing in Section 4.2.5) remain accessible, readable, and fully operational.	◐
The overall impression of the web application's content, including text, images, or other significant components for *TG* and *CM*, is confusing, and the orientation is inefficient. The structure lacks a clear hierarchy and coherent grouping of related topics, with labels or headings which are unclear or missing. The overall navigation structure is either absent or poorly designed, making accessing different sections of the content difficult. The interface does not ensure that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational.	○

Table 4.6: Fulfillment conditions for evaluation criteria CLN2

Condition & Fulfillment	
Every visible form, button, and other significant component for *TG* and *CM* has clear, descriptive, and concise labels that accurately describe the associated content and can be easily understood and interpreted according to each element's purpose. The provided labels are positioned in a way that is easy to locate and read for users, making it possible to identify the associated labels quickly without confusion or unnecessary effort. Additionally, the labels are readable and accessible to text-to-speech readers. The interface maintains its functional integrity, ensuring that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational.	●
Many of the visible forms, buttons, and other significant components for *TG* and *CM* have labels that are either incomplete, unclear or not accurately describing the associated content. Furthermore, while most labels are placed appropriately for easy visibility and readability, there may be instances where a few placements could be improved, as some may be slightly harder to find or read. Additionally, the labels generally facilitate text-to-speech readers. There might be some instances where these kinds of tools cannot fully access and read the labels. The interface maintains some of its functional integrity, ensuring that the majority of the significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational.	◐
The overall impression of the content of the web application, including text, images, or other significant components for *TG* and *CM*, have labels that are either incomplete, unclear, or not accurately describing the associated content. Furthermore, the placement of the labels is not accomplished appropriately and lacks easy visibility and readability. Additionally, the labels do not facilitate text-to-speech readers; therefore, these kinds of tools cannot fully access and read the web content. The interface does not ensure that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational.	○

Table 4.7: Fulfillment conditions for evaluation criteria CLN3

Condition & Fulfillment	
The overall impression of the web application's content, including text, images, or other significant components for *TG* and *CM*, is well-organized and presented predictably for the user. The user interactions themselves are clearly indicated, and the overall functionality of the website functions as expected and returns the predicted results. The interface maintains its functional integrity, ensuring that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational regarding their predictability.	●
Many visible forms, buttons, or other significant components for *TG* and *CM* are organized in a partially inconsistent way such that the outcomes are sometimes not predictable for the user. In this particular case, user interaction is sometimes indicated, and there are cases in which the predictability of the functionality or the outcomes is lacking or unclear. This kind of web content organization allows the user to make decisions to some extent based on the available information and feedback. However, on the other hand, it includes gaps in understanding the effect of the actions, and the consequences may not always be fully comprehended. The interface maintains some of its functional integrity, ensuring that the majority of the significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational regarding their predictability.	◐
The overall impression of the web application's content, including text, images, or other significant components for *TG* and *CM*, is insufficiently organized and structured in an inconsistent way such that interactions and outcomes lack predictability and understanding. In this particular case, user interactions are not clearly indicated. Users cannot foresee the results based on the provided information and feedback, which makes them unable to determine what to expect from their interactions or make decisions based on that. The interface does not fully maintain its functional integrity and does not ensure that all significant operations (cf. Listing in Section 4.2.5) are accessible, readable, and fully operational regarding their predictability.	○

CLN4-Different navigating means: This criterion describes that using different navigational structures allows users to use the most appropriate option for them (e.g., a hierarchical menu and search). A website or web application should provide multiple navigation options based on different cases to improve accessibility and user satisfaction (cf. Table 4.8).

CLN5-Text supplemented by illustrations: This criterion describes that textual components should have images, graphs, and similar supplements to improve comprehension. Visual aids support textual components; they help understand the message or information quickly and give more information than the provided textual descriptions (cf. Table 4.9).

Table 4.8: Fulfillment conditions for evaluation criteria CLN4

Condition & Fulfillment	
The web application, including its significant components for *TG* and *CM*, provides a variety of navigational structures, including at least a hierarchical menu and a search, to navigate through web content or to search for specific content. Due to the multiple options, choosing the most suitable and efficient navigation method for specific use cases or user needs is possible. The interface maintains its functional integrity, ensuring that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational regarding the provided navigational aids and search function.	●
The web application, including its significant components for *TG* and *CM*, provides limited navigational structures, such as hierarchical menus and search possibilities. The users of this website or web application can only choose from a limited range of navigation aids. They sometimes have to deal between hierarchal menus or search possibilities, as some components do not support both. In addition to that, the provided navigation possibilities are not suitable for every intended functionality of the website or its content and, therefore, cannot fully satisfy the needs of disabled users. The interface sometimes maintains its functional integrity, ensuring that the majority of the significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational regarding the provided limited number of potential navigational methods.	◐
The web application, including its significant components for *TG* and *CM*, provides either one navigational structure, such as hierarchical menus or search, or none at all. Additionally, the web application does not provide any alternatives, hindering the accessibility and user experience of the provided web content. The interface cannot maintain its functional integrity and does not ensure that the significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational because of the inadequate offer of navigational methods.	○

Table 4.9: Fulfillment conditions for evaluation criteria CLN5

Condition & Fulfillment	
The web application, including its significant components for *TG* and *CM*, contains images, graphs, or similar illustrations that improve the provided functionalities' understanding by supporting the textual components. Additionally, these visual aids are suitable, as they help to understand the information expressed in the text. The interface maintains its functional integrity, ensuring that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational.	●
The web application, including its significant components for *TG* and *CM*, do not contain visual aids, or the provided ones are unsuitable and do not improve the understandability of the textual components. The interface cannot maintain its functional integrity and does not ensure that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational because of the lack of suitable visual aids.	○

Evaluation Criteria for Physical disabilites

This section provides the evaluation criteria specifically for physical disabilities and impairments.

P1-Full Keyboard Support: This criterion describes the case that all possible interactions should be doable with the keyboard only without the need for pointing devices. This criterion was assessed via manual assessment and also with the browser extension Accessibility insights[6], which provides a walkthrough option for manual accessibility checks, based on WCAG 2.1 (cf. Table 4.10).

Table 4.10: Fulfillment conditions for evaluation criteria P1

Condition & Fulfillment	
The overall web application, including its significant components for *TG* and *CM*, can be operated only via the keyboard. The user can interact with all the intended functionalities without using a mouse or similar pointing device. The interface maintains its functional integrity, ensuring that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational if used only via the keyboard.	●
The overall web application, including its significant components for *TG* and *CM*, cannot be operated independently with the keyboard. It is impossible to interact with the intended functionalities only via the keyboard, and a mouse or similar pointing device in combination is needed. The interface cannot maintain its functional integrity, ensuring that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational if used only via the keyboard.	○

P2-Sufficient time limits to react: This criterion describes the case of providing sufficient time limits to the user for reacting. In this case, the users' reaction time should not lead to errors, interruption of the current task, or similar. This criterion was assessed by carrying out the same interactions at different speeds (cf. Table 4.11).

P3-Controls, images and other components with text alternatives: This criterion on alternative texts and ARIA labels is essential for voice and text-to-speech recognition. It ensures that people who use voice recognition or text-to-speech to access or understand websites can do this as quickly as any other user. For assessing this criterion, the browser extension WAVE[7] was used to detect missing aspects automatically (cf. Table 4.12).

[6] Accessibility Insights for Web, `https://accessibilityinsights.io/docs/web/overview/`, (Access: 29.07.2023)

[7] WAVE, `https://wave.webaim.org/`, (Access: 18.08.2023)

Table 4.11: Fulfillment conditions for evaluation criteria P2

Condition & Fulfillment	
The interaction possibilities provided by the web application, including its significant components for *TG* and *CM*, can be operated without being hindered by time limits. All interactions provide sufficient time limits for accomplishing the desired and intended task without resulting in errors or interruptions. The interface maintains its functional integrity, ensuring that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational with users of different working and interacting speeds.	●
Some of the interaction possibilities provided by the web application, including its significant components for *TG* and *CM*, can be operated without being hindered by time limits, or the interface provides appropriate alternatives for satisfactorily carrying out the intended actions. Furthermore, certain interactions can impede the intended functionality or make it more challenging for the user. In some instances, they might be unable to complete their current task. However, all of these functionalities have alternative ways of completing the task in the intended way and successfully, e.g., a different menu entry or the possibility of finishing the current task without getting stuck on insufficient reaction time limits. The interface maintains some of its functional integrity, ensuring that the majority of the significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational with users of different working and interacting speeds.	◐
At least one of the interaction possibilities provided by the web application, including its significant components for *TG* and *CM*, cannot be operated and carried out without being hindered by time limits. Additionally, the website does not provide appropriate alternatives for satisfactorily carrying out the intended actions. The interface cannot maintain some functional integrity and does not ensure that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational with users of different working and interacting speeds.	○

Table 4.12: Fulfillment conditions for evaluation criteria P3

Condition & Fulfillment	
Every visible text, image, or other significant component for *TG* and *CM* has an appropriate text alternative or ARIA label defined, as assessed by the accessibility report of the previously mentioned browser extension WAVE. The interface maintains its functional integrity, ensuring that all significant operations (cf. Listing in Section 4.2.5) remain accessible, readable, and fully operational if used via voice recognition or text-to-speech.	●
Some visible text, image, or other significant components for *TG* and *CM* have appropriate text alternative or ARIA label defined, as assessed by the accessibility report of the previously mentioned browser extension WAVE. However, at most, five web components from the list of significant operations (cf. Listing in Section 4.2.5) do not provide any text alternative or ARIA label.	◐
Many visible text, image, or other significant components for *TG* and *CM* do not have an appropriate text alternative or ARIA label defined, as assessed by the accessibility report of the previously mentioned browser extension WAVE. However, over five web components from the list of significant operations (cf. Listing in Section 4.2.5) do not provide any text alternative or ARIA label.	○

P4-Visual & non-visual orientation or navigational cues: This criterion checks if there exist visual and non-visual orientation or navigational cues which are essential components for navigating and also make the current selection visible to the user. This criterion assesses the existence and quality of three components in particular, which are listed below:

- **Indicators & Focus:** This includes mostly visual web components used to highlight the current focus or to guide users through the interface (e.g., Tooltip, Progress Bar).
- **Orientation:** This describes the satisfaction of the provided web orientation, i.e., how clearly, and effectively the overall web orientation is structured.
- **Keyboard Navigation & Skipping:** This includes if the website provides complete keyboard navigation with efficient iteration and skipping possibilities, such as over-page headers or navigation bars.

The fulfillment conditions for this evaluation criterion can be found in Table 4.13.

Table 4.13: Fulfillment conditions for evaluation criteria P4

Condition & Fulfillment	
Every visible text, image, or other significant component for *TG* and *CM* provides suitable visual and non-visual cues, especially for highlighting the current focus and activity of the user, providing appropriate visual feedback and sufficient keyboard navigation support. To sum up, the three components as described in Listing for P4 in Section 4.2.6 are satisfactorily fulfilled for the significant operations (cf. Listing in Section 4.2.5).	●
Some of the visible text, image, or other significant components for *TG* and *CM* provide suitable visual and non-visual cues, especially for highlighting the current focus and activity of the user or having appropriate visual feedback and sufficient keyboard navigation support. To sum up, two out of three components, as described in Listing for P4 in Section 4.2.6, are satisfactorily fulfilled for the significant operations (cf. Listing in Section 4.2.5).	◐
Many of the visible text, image, or other significant components for *TG* and *CM* do not provide suitable visual and non-visual cues, especially for highlighting the current focus and activity of the user, nor provide appropriate visual feedback and sufficient keyboard navigation support. To sum up, none or only one of the components as described in Listing for P4 in Section 4.2.6 are satisfactorily fulfilled for the significant operations (cf. Listing in Section 4.2.5).	○

P5-Logical navigational mechanisms and page functions: This criterion describes that the page structure should not be misleading or that navigating does not show unexpected behavior. This criterion assesses the existence and quality of three components in particular, which are listed below:

- **Menu Structuring:** This criterion describes the satisfactory level of the provided menu structuring.

- **Keyboard Navigation:** This criterion checks if the website provides complete keyboard navigation with logical and expected behavior, i.e., navigation that works in an expected and natural order.

The fulfillment condition for this evaluation criterion can be found in Table 4.14.

Table 4.14: Fulfillment conditions for evaluation criteria P5

Condition & Fulfillment	
Every visible text, image, or other significant component for *TG* and *CM* is part of a suitable navigation mechanism, which navigates through the website as expected and in a natural order. To sum up, the two components as described in Listing for P5 in Section 4.2.6 are satisfactorily fulfilled for the significant operations (cf. Listing in Section 4.2.5).	●
Some visible text, images, or other significant components for *TG* and *CM* is part of a suitable navigation mechanism, which navigates through the website as expected and in a natural order. To sum up, one of the two components, as described in Listing for P5 in Section 4.2.6, is satisfactorily fulfilled for the significant operations (cf. Listing in Section 4.2.5).	◐
Many visible text, images, or other significant components for *TG* and *CM* are not part of the suitable navigation mechanism, which navigates the website as expected and in a natural order. To sum up, both components, as described in Listing for P5 in Section 4.2.6, are not satisfactorily fulfilled for the significant operations (cf. Listing in Section 4.2.5).	○

P6-Large clickable areas: This specific criterion emphasizes the importance of ensuring that the clickable areas of various interactive elements are large enough, reducing reliance on precise fine motor skills. Additionally, adequate spacing between multiple elements minimizes the risk of unintentional clicks or falling into clickable traps (cf. Table 4.15).

P7-Error Correction Options: This criterion highlights the need for correction options, such as undoing or redoing actions, or other error correction possibilities, such as deleting, renaming, and similar. This feature is essential, as quick, easy, and efficient recovery and error correction improve accessibility and usability. This criterion assesses the existence and quality of at least those error correction options in particular, which are listed below:

- **Undo & Redo Actions:** This criterion emphasizes the importance of providing users with the option to undo or redo their last action for every user interaction.
- **Update textual input:** This criterion examines whether textual inputs enable users to update the content of textual fields easily and quickly.

The fulfillment conditions for this evaluation criterion can be found in Table 4.16.

Table 4.15: Fulfillment conditions for evaluation criteria P6

Condition & Fulfillment	
Every visible text, image, or other significant components for *TG* and *CM* are designed in a way such that clickable traps are avoided, i.e., the size of the element avoids unexpectedly and not intentionally clicking in or out of the element borders. Additionally, for selecting the elements, no exact usage of the mouse or pointing device is needed, as clicking can be done easily. This applies to the elements of the significant operations (cf. Listing in Section 4.2.5).	●
Some of the visible text, image, or other significant components for *TG* and *CM* are designed in a way such that generally clickable traps are avoided, i.e., the size of the element avoids unexpectedly and not intentionally clicking in or out of the element borders. However, some elements lead to clicking traps due to inappropriate element size and lead to not intended interactions or errors. This applies to the elements of the significant operations (cf. Listing in Section 4.2.5).	◐
Many of the visible text, image, or other significant components for *TG* and *CM* are designed in a way such that clickable traps are most likely, i.e., the size of the element lead to unexpected and not intentionally clicking in or out of the element borders and leads to not intended interactions or errors. This is applicable for the elements of the significant operations (cf. Listing in Section 4.2.5).	○

Table 4.16: Fulfillment conditions for evaluation criteria P7

Condition & Fulfillment	
Every visible text, image, or other significant components for *TG* and *CM*, which are editable, also provides at least both error correction options as described in Listing for P7 in Section 4.2.6 for the elements of the significant operations (cf. Listing in Section 4.2.5).	●
Some of the visible text, image, or other significant components for *TG* and *CM*, which are editable, also provide at least both error correction options as described in Listing for P7 in Section 4.2.6. In addition, some elements of the same type only provide one of the previously mentioned error correction options. This applies to the elements of the significant operations (cf. Listing in Section 4.2.5).	◐
Many of the visible text, image, or other significant components for *TG* and *CM*, which are editable, do not provide at least one of the error correction options as described in Listing for P7 in Section 4.2.6. Most elements of the same type do not provide any of the previously mentioned error correction options. This applies to the elements of the significant operations (cf. Listing in Section 4.2.5).	○

Additional support tools or software

Not every evaluation criterion can be checked easily manually, so additional tools were used to get the best results. Automated tools can help assess website accessibility; however, they should not be the sole basis for evaluation. While they can generate a rating, relying exclusively on them may lead to a misleading perception of a website's full accessibility [CH22]. Combining manual and automatic assessment helps gather the best experience possible in this context. The references of the used extensions can be found in **Appendix 3: Additional support tools and extensions**. The overview below

describes how each extension was applied and for which evaluation criteria it was used.

Magnifying Glass (Hover Zoom): This extension can be used on any page as an additional aid to increase the size of elements or text, especially if the web pages do not provide a resizing functionality or it is not satisfying enough. This extension is used in the context of the evaluation criterion **V1** (cf. Section 4.2.6) to check if additional zooming aids can be applied on the web page if needed by the user without any loss of information or unexpected behavior.

OneLine: This tool is a reading aid extension that highlights the first row of the corresponding web pages to help disabled or impaired users by increasing their focus and reading efficiency. This extension is an addition for both the evaluation criteria **V1** and **V2** (cf. Section 4.2.6) in order to check if the content allows using additional tools reasonable and efficiently, especially if changing the appearance according to the customer needs is not possible for the regarding web content.

Read Aloud: This web browser extension is a text-to-speech tool and is applied to check whether the evaluation criterion **V3** (cf. Section 4.2.6) is fulfilled or not. In that particular case, the use of this extension can check if the provided web content is suitably prepared for this kind of tool and if the content can be read aloud to the user in such a way that it makes sense and the web page can be used sufficiently.

WAVE & Accessibility Insights for Web: Both extensions can check automatically and create accessibility reports for known accessibility issues. The results are used as a combination, especially for the evaluation criteria **V3** (cf. Section 4.2.6), **CLN2** and **CLN6** (cf. Section 4.2.6), e.g., to check for contrast errors, missing alternative texts or descriptions.

4.3 Results & Findings

This section describes the results and findings achieved by evaluating the web modeling tools as described previously in this chapter. In the first part, the overall outcomes for each assessed tool will be presented (cf. Table 4.17). Afterward, this section will provide a more detailed overview of the assessment results by categorizing them according to the disability types (cf. Sections 4.3.2 for visual, 4.3.3 for learning, and 4.3.4 for physical disabilities). In addition, Section 4.3.5 highlights the results from the point of view of each evaluated modeling tool.

4.3.1 General Evaluation Results

Table 4.17: Overall evaluation result for the categories Tool Support & GUI (TG) and Canvas & Model (CM) for Visual (V), Cognitive, Learning, Neurological (CLN), and Physical (P) Disabilities.

Tool	Lucidchart		GenMyModel		Gliffy		diagrams.net		Creately		Cacoo		UMLetino		Diagramo		miro		BPMN.io	
	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM
V1	◐	●	●	●	◐	●	◐	●	◐	●	●	●	●	●	◐	●	◐	●	◐	●
V2	○	●	○	○	○	●	●	●	○	●	○	●	○	○	○	●	○	●	○	○
V3	○	○	●	◐	○	◐	○	○	○	○	○	○	●	○	○	○	○	○	○	○
CLN1	●	●	●	●	●	●	●	●	◐	●	●	●	●	●	●	●	●	●	●	●
CLN2	◐	◐	●	◐	◐	◐	◐	●	◐	◐	◐	◐	●	●	○	●	○	◐	◐	◐
CLN3	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
CLN4	●	○	◐	◐	●	○	●	●	●	○	◐	○	○	○	◐	○	◐	○	●	○
CLN5	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
P1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P2	◐	●	●	●	◐	○	◐	◐	◐	◐	◐	●	◐	◐	◐	○	◐	●	◐	●
P3	●	○	○	○	●	●	●	○	●	○	○	○	●	○	○	○	●	○	●	○
P4	◐	◐	◐	◐	◐	◐	◐	●	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐
P5	◐	◐	◐	◐	◐	◐	◐	●	○	◐	◐	◐	●	◐	◐	◐	◐	◐	◐	◐
P6	●	◐	○	○	●	◐	●	◐	◐	◐	○	◐	●	●	●	◐	●	◐	●	●
P7	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

The general results of the complete tool evaluation are presented in Table 4.17. This table shows how each of the assessed tools (cf. Table 4.1) are evaluated against each of the evaluation criteria (V1-V3, CLN1 - CLN5, P1-P7, cf. Section 4.2.6). As mentioned in *Step 2* of the evaluation process description (cf. Section 4.2.5), the assessment for each tool and criteria was conducted in two phases: *Tool Support & GUI* and *Canvas & Model* (cf. Section 4.2.5).

The overall impression of the evaluation is that, in general, the outcomes for each web modeling tool were not remarkably different. The issues or gaps are determined by observing the criteria that are not satisfactorily fulfilled. The following evaluation criteria of the assessment are outstanding: None of the evaluated modeling tools fully meet the *P1 - Full-Keyboard Support* criterion. However, the *P7 - Error Correction Options* criterion is entirely satisfactorily fulfilled, and the *V1 - Customizing text and images size* criterion is almost entirely fulfilled. For detailed results, refer to Sections 4.3.2, 4.3.3, and 4.3.4.

Furthermore, the assessment reveals the following observations:

- **Each tool interprets keyboard support differently:** During the assessment, it was evident that each evaluated tool varies in its level of keyboard support and how it is implemented. However, none of the tools provide full keyboard support, and users are still required to use a mouse or similar pointing device in conjunction with the existing shortcuts. This lack of full keyboard support is particularly unsatisfactory for physically disabled users, as indicated by evaluation criteria P1 (cf. Section 4.2.6).
- **Accessibility vs. Efficiency & Usability:** Most tools are designed to enable the user to use the overall tool efficiently and execute the intended functionalities,

more than providing accessibility solutions or alternatives. Usability aims to make products and services easy for the average user. At the same time, accessibility ensures that individuals with specific needs, such as disabilities, can also use and interact with them effectively. Even though the meanings are different, both approaches do influence each other [TBdR05]. Implementing appropriate solutions for usability and efficiency will most likely improve accessibility to some extent. However, mainly these tools are not providing functionalities or options dedicated explicitly to accessibility and therefore are not accessible to their finest.

- **Look & Feel is rarely adaptable:** The observation has shown that most assessed tools have limited possibilities for adapting the Look & Feel, which includes the (background) colors, font sizes, colors, contrast, and similar. In the case of web modeling tools, most of them provided these adaption possibilities for the created model and its components (e.g., textual labels) but not for the overall interface and the GUI of the tools (e.g. menu header).

4.3.2 Results for the visual disability support

Table 4.18: Evaluation result for the categories Tool Support & GUI (TG) and Canvas & Model (CM) for Visual (V) Disabilities.

Tool	Lucidchart		GenMyModel		Gliffy		diagrams.net		Creately		Cacoo		UMLetino		Diagramo		miro		BPMN.io	
	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM
V1	◐	●	●	●	◐	●	◐	●	◐	●	●	●	●	●	◐	●	◐	●	◐	●
V2	○	●	○	○	○	●	●	●	○	●	○	●	○	○	○	●	○	●	○	○
V3	○	○	●	◐	○	◐	○	○	○	○	○	○	●	○	○	○	○	○	○	○
Result	Partially		Fully		Partially		Fully		Partially		Partially		Partially		Partially		Partially		Not	

The tool evaluation results for users with visual disabilities or impairments can be observed in Table 4.18.

The observations show that most of the evaluated web modeling tools are capable of providing options for **customizing text and image sizes (V1)** (cf. Table 4.2), without any misrepresentation or information loss or the need for any additional support tool, as the resizing or zooming in or out is possible with the default browser zooming in or out functionalities for *Tool Support & GUI*. For *Canvas & Model*, the tool's interface provided the same functionality for resizing in the canvas. For the diagram elements themselves, the tools, which fulfilled this condition, also included settings to adapt the size of single diagram components, such as nodes, edges, labels, and similar.

This evaluation observed that seven tools had limited *Tool Support & GUI* (Lucidchart, Gliffy, diagrams.net, Creately, Diagramo, Miro, BPMN.io). The common issue among these tools was the lack of resizable elements in their GUI, such as menu panels, headers, footers, and similar. As a result, users with visual disabilities might find it challenging to read and comprehend the content these tools display. A representative example of the tool Miro, on behalf of the remaining six tools with the same resizing issue, can be seen in Figure 4.9. The figure clearly illustrates that the zooming process from 100%

to 150% only affects the models themselves and not the other components, such as the menu panel in this particular case.

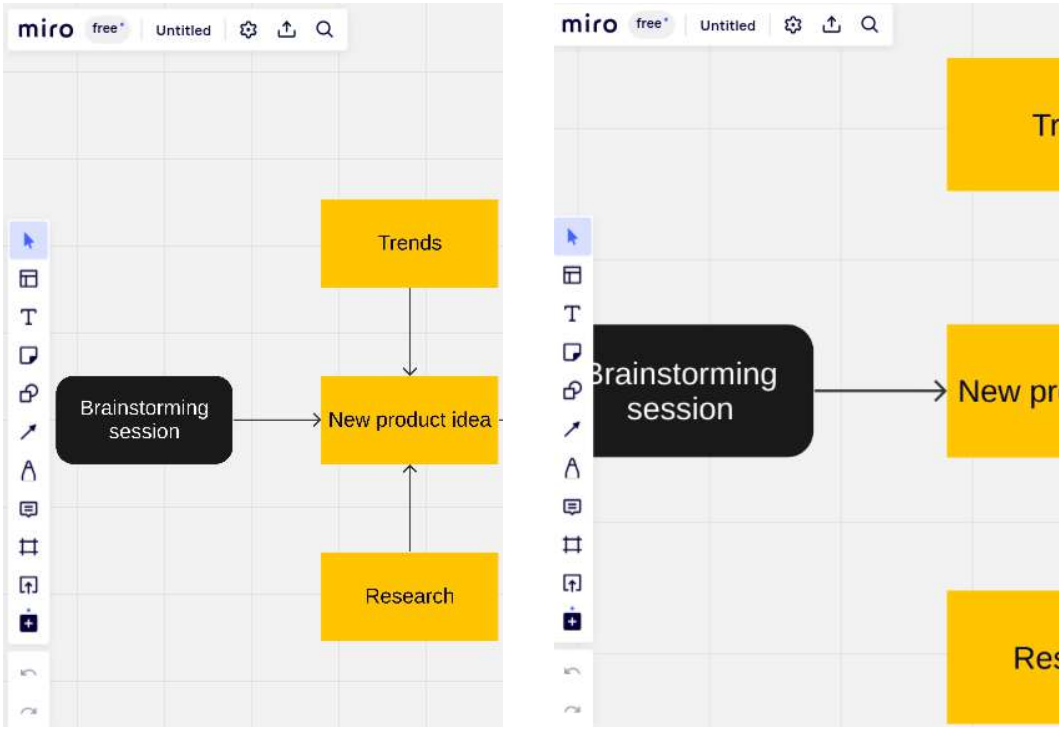

Figure 4.9: Example Snippet of Miro: Zooming from 100% to 150% only resizes the model, not the remaining elements.

Contrary to the first visual evaluation criteria, it is clearly noticeable that the remaining outcomes ended up differently for the assessed tools. The case of **V2**, which provides options to **customize fonts and colors** (cf. Table 4.3), shows a clear pattern in the results. One can observe that most of the tools provide options for adapting font and color-related settings, but this is only limited on *Canvas & Model*, i.e., on the canvas and model components themselves. Except for one web modeling tool, namely, diagrams.net (cf. Figure 4.10), none of them provide the user with additional options to adapt the overall look and feel, including the fonts and colors of the interface components, i.e. for *Tool Support & GUI*.

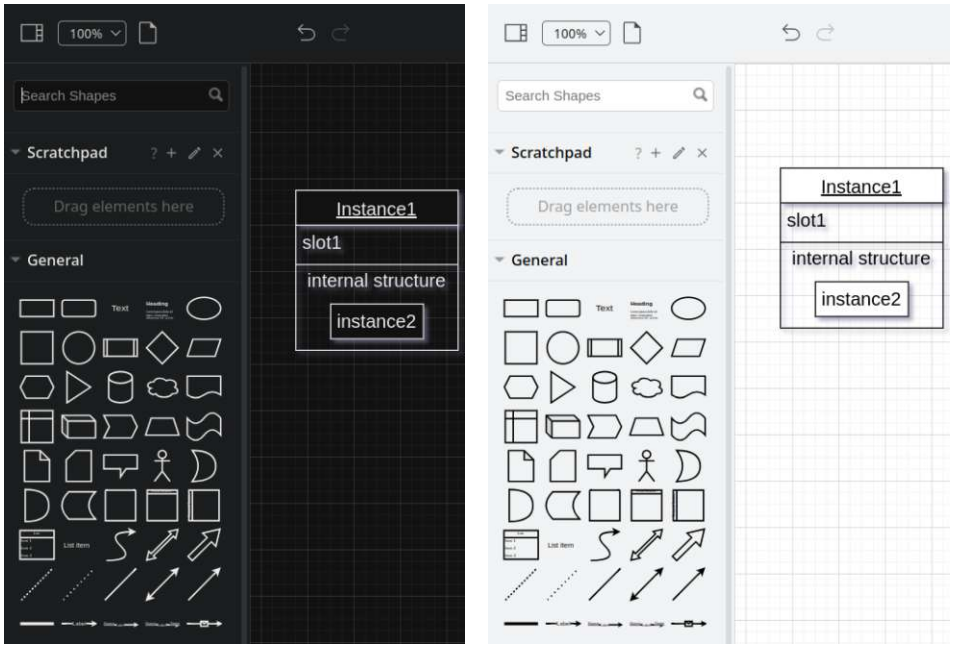

Figure 4.10: Example Snippet of diagrams.net: Changing appearance of the overall tool

However, there is a missed opportunity, particularly in customizing the provided functionalities and menus to cater to the specific needs of visually disabled or impaired users. This approach would offer a more comfortable usage of the GUI in modeling tools for these individuals.

The third and last evaluation criteria **V3 - Creating a text-to-speech synthesis** (cf. Table 4.4) in this disability category has cut poorly in general, as seven out of ten modeling tools did not fulfill this criterion for *Tool Support & GUI* nor for *Canvas & Model*.

Those tools could not generate proper text-to-speech synthesis of the content due to missing text or image alternatives. Out of all the assessed ten tools GenMyModel returned the best possible text-to-speech synthesis as the criterion is fulfilled for *Tool Support & GUI* and partially fulfilled for *Canvas & Model*. It was the only one which recognized most of the canvas elements and diagram components. On the other hand, the tool Gliffy partially fulfilled the criterion for *Canvas & Model* and UMLetino in contrary for *Tool-Support & GUI*.

In general, the outcomes for the *Canvas & Model* are not surprising as it is highly challenging to provide a valuable and logical synthesis for created models, as the semantics of the graphical elements cannot be easily expressed only with text, as researched by Cross et al. [CCD20], who dealt with transforming diagrams into text. The challenge exists due to the different diagram types, syntaxes, and their various semantic rules and meanings.

Conclusion: Overall, the current state of visual disability support in known web modeling tools can be summarized as follows: From the perspective of *Tool Support & GUI*, the tools provide sufficient possibilities for resizing and adapting text, images, and other GUI components. However, when it comes to the customizability of font types, component sizes, and the possibility of creating proper text-to-speech synthesis usable with screen readers, the majority of the tools had their issues and were not able to accomplish these tasks as intended.

In contrast, the outcomes for *Canvas & Model* turned out to be slightly better regarding resizing and customizing specific diagram elements. As previously mentioned, there was no possibility of creating a practical text-to-speech synthesis for the diagram and its components.

In conclusion, it can be said that two of the assessed tools are **fully**, one **not**, and the rest is **partially** capable of supporting people with visual disabilities or impairments according to the pre-defined evaluation criteria. The overall result for each evaluated modeling tool can be observed in the last row of the results table (cf. Table 4.18).

4.3.3 Results for the cognitive, learning, and neurological disability support

Table 4.19: Evaluation result for the categories Tool Support & GUI (TG) and Canvas & Model (CM) for Cognitive, Learning and Neurological (CLN) Disabilities.

Tool	Lucidchart		GenMyModel		Gliffy		diagrams.net		Creately		Cacoo		UMLetino		Diagramo		miro		BPMN.io	
	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM
CLN1	●	●	●	●	●	●	●	●	◐	●	●	●	●	●	●	●	●	●	●	●
CLN2	◐	◐	●	◐	◐	◐	◐	●	◐	◐	◐	◐	●	●	○	●	○	◐	◐	◐
CLN3	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
CLN4	●	○	◐	◐	●	○	●	●	●	○	◐	○	○	○	◐	○	◐	○	●	○
CLN5	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Result	Fully		Fully		Fully		Fully		Fully		Fully		Fully		Fully		Fully		Fully	

The tool evaluation results for users with cognitive, learning, and neurological disabilities or impairments can be observed in Table 4.19.

From the collected data, it can be observed that all tools were fully capable of fulfilling the evaluation criteria **CLN1 - Clearly structured content** (cf. Table 4.5), **CLN3 - Predictable interaction** (cf. Table 4.7) and **CLN5- Text supplemented by illustrations** (cf. Table 4.9) to their finest, for both *Tool Support & GUI* and *Canvas & Model*. The satisfactory fulfillment of these three evaluation criteria shows that most modeling applications nowadays provide their user base with a structured interface, which does not confuse or overwhelm most users. This property also leads to better interaction possibilities, as the actions between the web application and the user are predictable and easy to understand. One significant aspect of this understandable design is using illustrations to supplement the textual components. This is because illustrations, graphs, or images lead to a better understanding at one glance but also help the user to avoid

reading any provided text, mainly if complex words or long text passages are provided (example in Figure 4.11).

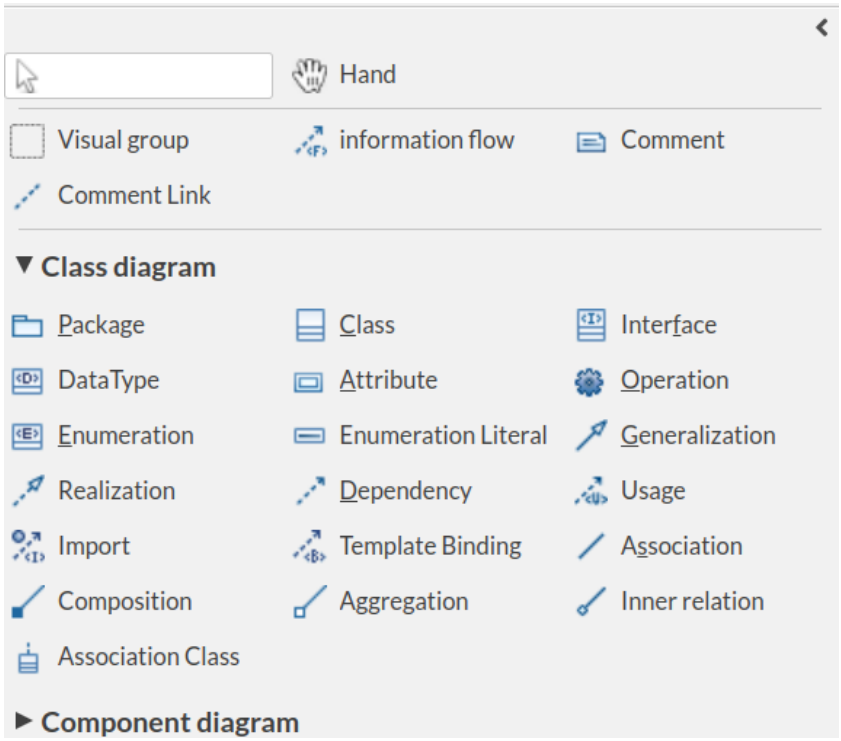

Figure 4.11: Example Snippet of GenMyModel: Text label with visual supplements

The second best criterion is **CLN2 - Consistent Labeling** (cf. Table 4.6). In this case, it is observable that most of the tools are in a condition where they only partially fulfill this criterion. In the case of *Tool Support & GUI*, six out of ten tools have partially satisfied usage of labels, which means that although these tools provide labels to their web components, there exist cases in which labels are missing, unsuitable, or sparely used. Furthermore, there were each two tools, GenMyModel and UMLetino, which satisfactorily fulfilled the usage of consistent labels. In contrast, two tools not fulfilling the criterion, are Diagramo and Miro (cf. Figure 4.12).

From the perspective of *Canvas & Model*, it can be observed that the majority of the tools were not able to entirely fulfill this criterion, as seven out of ten only partially fulfilled the consistent usage of labels. Only, diagrams.net, UMLetino, and Diagramo, were able to entirely fulfill the usage of consistent labels (cf. Figure 4.13).

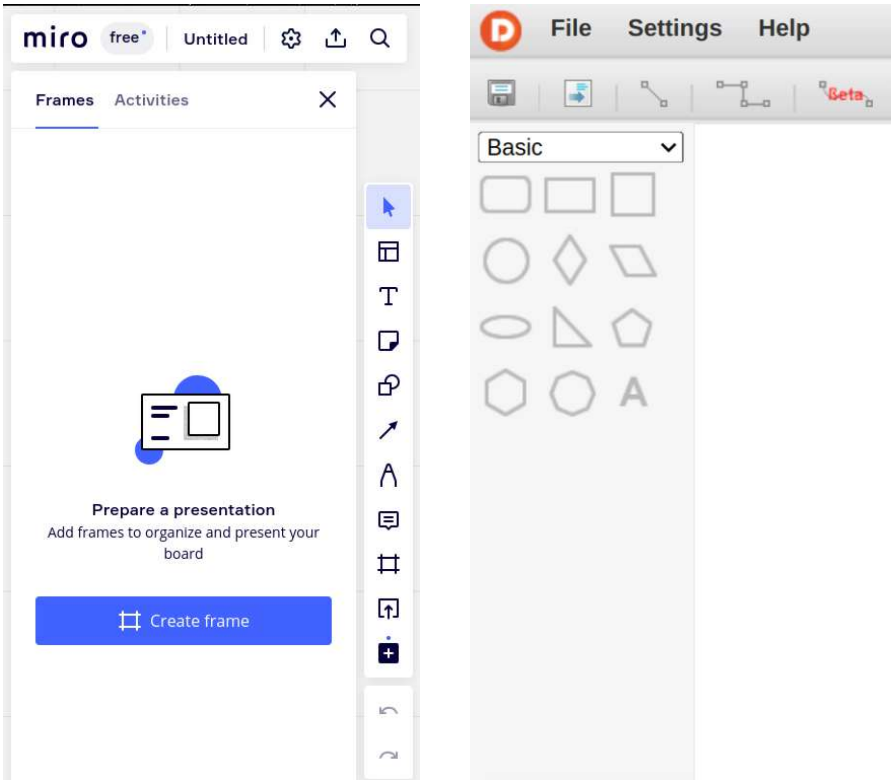

Figure 4.12: Example Snippet of Miro (Left) and Diagramo (Right): Insufficient Labeling

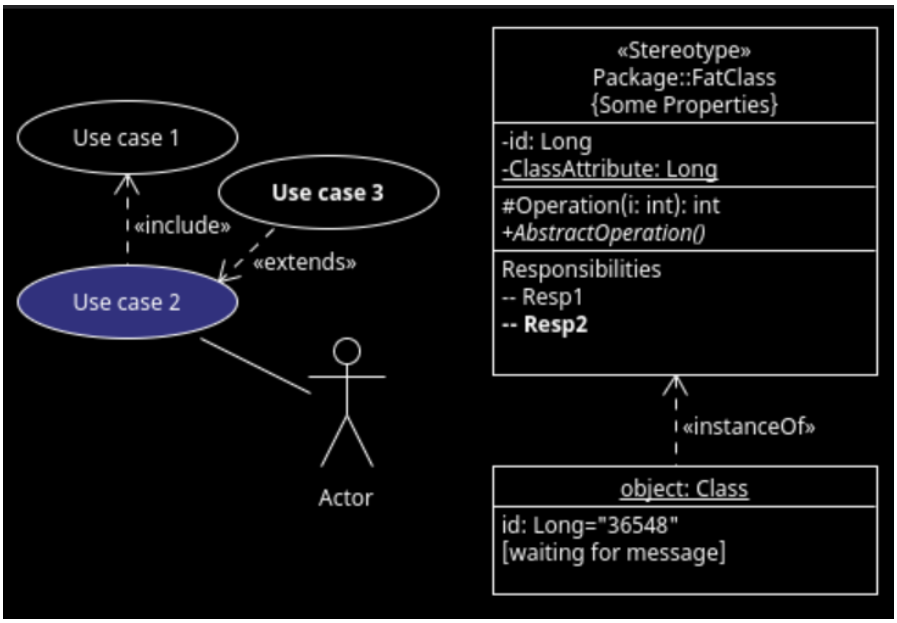

Figure 4.13: Example Snippet of UMLetino: Labelling from the perspective of *Canvas & Model*

The last criterion to be analyzed is **CLN4**, which is about providing **different navigating means** (cf. Table 4.8), which resulted in opposite outcomes compared between *Tool Support & GUI* and *Canvas & Model*.

Observing the evaluation results for *Canvas & Model*, it is evident that none of the tools fulfilled the condition of providing an appropriate navigational mean. However, there is an exception for two modeling tools, one being GenMyModel which fulfills the criterion partially, and the other one being diagrams.net which provides navigational means at a satisfactory level.

Observing the collected information from the perspective of *Tool Support & GUI*, it is evident that half of the tools **fully** and the other half **partially** fulfill the condition of providing a navigational mean for the GUI itself. However, one of the ten tools does not provide a GUI navigational means at all. This result is unsurprising as this tool is a more basic, textual-based diagram editor, namely UMLetino.

With this information, the conclusion can be driven that most of the extended diagram and model editors provide suitable navigation for navigating through the GUI and interface of the web application. However, a clear gap exists for successfully and efficiently iterating through the created model and its components.

Conclusion: Based on the evaluation of web modeling tools, it can be concluded that the support for cognitive, learning, and neurological disabilities is generally satisfactory. Most of these tools offer a clear, predictable, and understandable structure in their graphical user interface and its included components.

Moreover, including suitable illustrations and consistent labels significantly improve the user experience for individuals with CLN disabilities, as through these supplements, the interface of modeling tools becomes less perplexing, overwhelming, and more user-friendly. By contrast, the outcomes for *Canvas & Model* did not show huge differences, except for the case of providing clear, structured, and different navigational mechanisms for the models created by the user, which in comparison is provided and overall better for the tools GUI.

To sum up, aside from a few minor flaws, all the assessed web modeling tools showed that they satisfactorily support the cognitive, learning, and neurological disability type if looked at the results assessed via the pre-defined evaluation criteria. The overall results for each evaluated modeling tool can be observed in the last row of the results table (cf. Table 4.19).

4.3.4 Results for the physical disability support

Table 4.20: Evaluation result for the categories Tool Support & GUI (TG) and Canvas & Model (CM) for Physical (P) Disabilities.

Tool	Lucidchart		GenMyModel		Gliffy		diagrams.net		Creately		Cacoo		UMLetino		Diagramo		miro		BPMN.io	
	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM	TG	CM
P1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
P2	◐	●	●	●	◐	○	◐	◐	◐	◐	◐	●	◐	◐	◐	○	◐	●	●	●
P3	●	○	○	○	●	●	●	○	●	○	○	○	●	○	○	○	●	○	●	○
P4	◐	◐	◐	◐	◐	◐	◐	●	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐
P5	◐	◐	◐	◐	◐	◐	◐	●	○	◐	◐	◐	●	◐	◐	◐	◐	◐	◐	◐
P6	●	◐	○	○	●	◐	●	◐	◐	◐	○	◐	●	●	●	◐	●	◐	●	●
P7	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Result	Fully		Not		Fully		Fully		Not		Not		Fully		Not		Fully		Fully	

The tool evaluation results for users with physical disabilities or impairments can be observed in Table 4.20.

An excellent result can be observed for **P7 - Error Correction Options** (cf. Table 4.16), as every assessed tool provides efficient and easy-to-apply error correction options, e.g., renaming, undoing or redoing, for any possible mistakes that could happen while interacting with the tool.

The remaining evaluation criteria have resulted in similar result patterns. This, in particular, includes **P2 - Sufficient time limits to react** (cf. Table 4.11), **P3 - Controls, images and similiar with text alternatives** (cf. Table 4.12), **P4 - Visual and non-visual orientation or navigation cues** (cf. Table 4.13), **P5 - Logical navigational mechanism and page functions** (cf. Table 4.14) and lastly **P6 - Large clickable areas** (cf. Table 4.15). Mostly the outcomes were in between fully and partially fulfilling the conditions.

Based on the evaluation, it is evident that, except for GenMyModel and BPMN.io, the remaining tools only partially meet the criterion of offering adequate time limits for user response **(P2)** concerning *Tool Support & GUI*. The main problem arises from using Drag & Drop actions, which can be challenging for physically disabled users as it requires precise dragging using a mouse or similar device to move elements on the screen. Therefore, if there is no suitable alternative for these kinds of drag & drop actions, the condition cannot be fulfilled at its finest. However, as the rest of the GUI and overall Tool Support does not have to deal with insufficient reaction times, these criteria are voted as partially fulfilled. As the two explicitly named Tools, GenMyModel, and BPMN.io, provide time-independent alternatives for these kinds of actions, they result in satisfactory fulfillment of the condition.

A comparable outcome can be observed when examining the same criterion for *Canvas & Model*. There is an exception, as Gliffy and Diagramo are the two modeling tools that did not fulfill the condition. In contrast, five of the evaluated fulfilled the criterion entirely, and three partially from the perspective of *Canvas & Model*. The issue with the tools that do not offer adequate reaction limits for the canvas or model is that, particularly

during diagram creation or manipulation, there are no alternatives to dragging actions in general.

In this paragraph, the assessment criteria for including text alternatives for the website's different components will be further analyzed **(P3)**. The results rely on the automatically generated report via the browser extensions (cf. Section 4.2.6). The outcomes show that the majority of the tools (seven out of ten) provide enough text alternatives for *Tool Support & GUI*. The remaining three tools, GenMyModel, Caccoo, and Diagramo, could not provide adequate text alternatives for the components of the overall tool and its GUI.

Observing the same criterion for *Canvas & Model*, it is clear that in that case, only Gliffy could provide text alternatives also for the canvas components and the model.

The results for the evaluation criterion for providing large clickable areas **(P6)** show that from the perspective of *Tool Support & GUI*, most of the GUIs components are designed large enough to prevent clicking mistakes or avoid overly precise handling by the user. Only one tool, Creately, is partially fulfilling as it has some GUI components and elements with probably challenging sizes. Furthermore, during the manual assessment of GenMyModel and Cacoo, it is observed that these tools have smaller component sizes compared to the other tools, resulting in more misclicks (cf. Figure 4.14). The usage of the mouse requires greater precision and detail, which leads to the unsatisfactory fulfillment of this criterion.

From the perspective of *Canvas & Model*, the results show a similar pattern. It can be observed that seven tools offer partial support; two (UMLetino, BPMN.io) fully support the need for larger canvas or model elements, while one (GenMyModel) does not fulfill this requirement at all.

In the case of providing suitable and efficient visual or non-visual orientation and navigation cues **(P4)** (cf. Table 4.13), for both, *Tool Support & GUI* and *Canvas & Model* the tools lead to partial fulfillment. In general, they lost votes due to a common issue: the **Keyboard Navigation & Skipping**. The criterion was not entirely fulfilled due to the missing or inefficiency of the mechanisms to skip and iterate over blocks, such as over-page headers, navigation bars, or similar.

Furthermore, in the case of providing a logical navigational mechanism and page functions **(P5)** (cf. Table 4.14) for *Tool Support & GUI*, most of the assessed modeling tools resulted in partially fulfilling this accessibility need. These tools were not able to fulfill the condition **Keyboard Navigation**, which also overlaps with **P1**. However, as the **Menu structuring** was provided in a clear and useful way, the evaluation of this criterion resulted in partial fulfillment. Throughout the evaluation of various web modeling tools, it was observed that Creately was the only tool that did not meet the criterion for the tool's overall GUI, as shown in Figure 4.15. In this case, the overall menu layout has some issues, as the menu panels do not fully utilize the provided height of the page. As a result, users may need to scroll or click more to access all the content. Moreover, Creately places the menu with frequently accessed functionalities at the bottom of the page. This placement makes it challenging to quickly locate the most commonly used interactions,

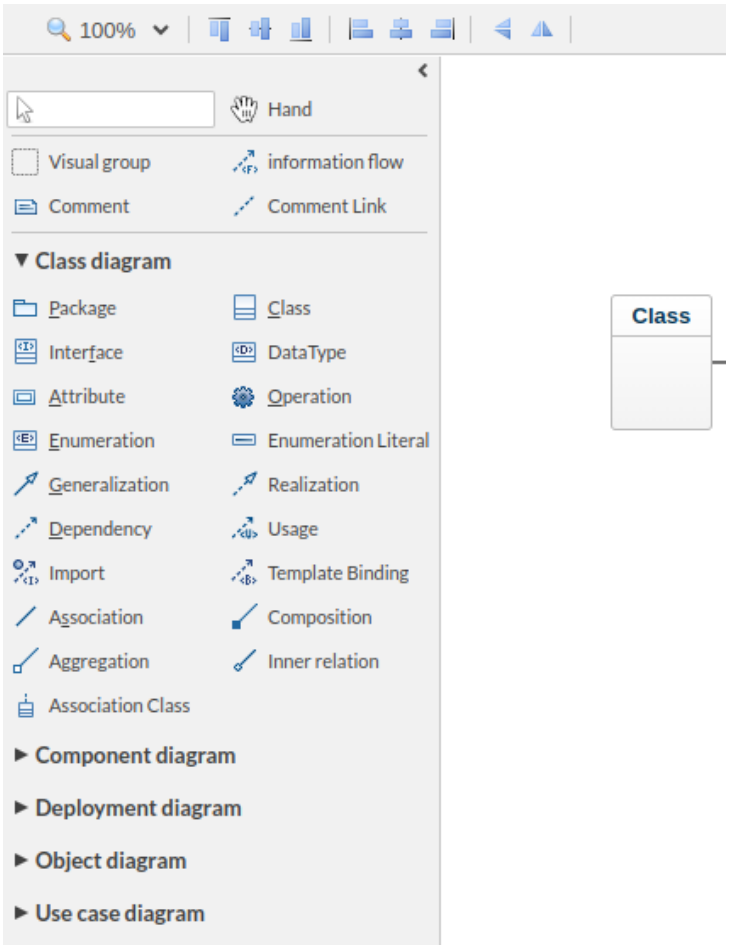

Figure 4.14: Example Snippet of GenMyModel: Insufficient Element Sizes

potentially leading to misunderstandings and an increased chance of unintentional clicks or similar errors. The overall impression is that the elements are closer to each other, looks overfilled, and not adequately grouped.

The same evaluation criteria assessed for *Canvas & Model* lead to nine partially and one entirely fulfilled modeling tool, namely diagrams.net. The modeling tool diagrams.net offers the best logical navigation system and page functions in comparison. The navigation within the model follows a natural order based on the elements, effectively identifying child elements, their relationships, and similarities.

One outstanding result is for **P1 - Full-Keyboard Support** (cf. Table 4.10). Both the *Tool Support & GUI* and *Canvas & Model* components lacked suitable full-keyboard support. While many modeling tools offer keyboard interactions or dedicated shortcuts, these options are often limited to specific features. Therefore, it is impossible to accomplish tasks entirely only via the keyboard.

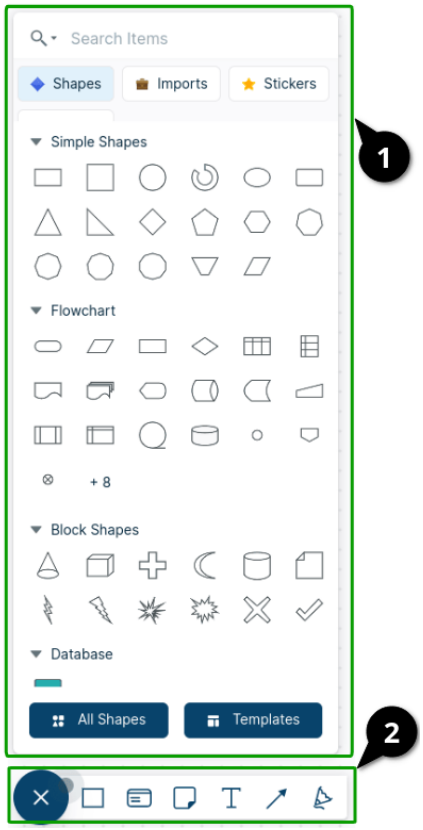

Figure 4.15: Example Snippet of Creately: Menu Panel

Conclusion: In conclusion, the current state of the physical disability support in the evaluated web modeling tools for *Tool Support & GUI* and *Canvas & Model* present a similar result scheme. The overall results generally range from entirely to partially fulfilling the given criteria, including minor flaws in some assessed modeling tools. The outcomes in this particular case are pretty mixed. However, overall it can be said that the majority of the tools do provide efficient and easy-to-use error correction options for any mistakes (P7), have satisfying reaction time limits (P2), and also most of them do provide valuable and adequate visual and non-visual orientation, logical navigation mechanisms (P4), text-alternatives (P3) and sufficient sized GUI components to avoid precise handling (P6). On the contrary, all of the evaluation subjects had in common that they do not provide full-keyboard support (P1), and some of them do lack in providing different navigational mechanisms (P5).

To sum up, aside from four tools, all the remaining assessed web modeling tools show that they adequately support the physical disability type if the overall results are assessed via the pre-defined evaluation criteria. In total, observing the results for all of the criteria together, the tools, GenMyModel, Creately, Cacoo, and Diagramo, did not perform satisfactorily throughout this assessment at different levels. Furthermore, this disability

category had the only evaluation criterion that none of the modeling tools could fulfill, as none of them provided the essential full-keyboard support (P1). This is especially important regarding web modeling tools, as these tools currently rely heavily on the mouse or similar pointing devices.

4.3.5 Results of the evaluated web modeling tools

In this section, the overall results and accessibility states from the perspective of the assessed web modeling tools will be explained in detail.

This section only highlights the strengths and weaknesses compared to the remaining modeling tools. Suppose the majority of the tools resulted in averagely the same strengths and weaknesses for an evaluation criterion. In that case, those criteria will not be mentioned in the following explanations, as those are collectively described in the previous Sections (cf. Sections 4.3.2, 4.3.4, and 4.3.3).

Furthermore, this section highlights any additional accessibility features that the tools offer to enhance overall accessibility, which were not covered in the evaluation criteria.

For better readability, *Tool Support & GUI* will be abbreviated as *TG* and *Canvas & Model* as *CM*.

Lucidchart

According to this tool evaluation, Lucidchart can partially support visual disabilities and fulfills the majority of the criteria for physical, cognitive, learning, and neurological (CLN) disabilities. Of all the evaluated tools, Lucidchart is the only one providing an accessibility report. This report states that Lucidchart is currently conforming the levels A and AA of WCAG 2.1 [8]. In addition to that, this tool also includes additional accessibility features, such as the capability of creating an accessible PDF of the project. An accessible PDF can be created and downloaded for further use, e.g., with screen readers or similar assistive technologies. It allows to set the order of the elements, set customizable text alternatives, and similar. Even though this functionality is helpful, it is necessary that these PDFs and their settings need to be adapted and downloaded beforehand by abled persons and cannot be used in full effect concurrently while using the tool itself. An example of this is that, generally, (cf. V3 in Table 4.18 for Lucidchart) struggles to generate a satisfactory text-to-speech synthesis for the GUI or the Canvas and its components without creating the accessible PDF. Therefore, in this particular evaluation, the criterion mentioned earlier (V3) is considered unfulfilled since it necessitates extra steps and is not directly provided within the tool.

[8] Lucidchart Accessibility, `https://help.lucid.co/hc/de/articles/360049860791#accessibility-compliance-report`, (Access: 25.06.2023)

Table 4.21: Strengths and Weaknesses of Lucidchart compared to other assessed tools

Strengths	Weaknesses
None of the evaluation criteria stood out significantly compared to the others.	None of the evaluation criteria stood out significantly compared to the others.

GenMyModel

This web modeling tool has satisfactorily supported both disability types, visual and CLN, but did not perform well for physical disabilities. GenMyModel did not primarily focus on offering specific accessibility solutions, as it was not designed to cater to the specific needs of individuals with disabilities. Therefore, the tool does not include additional accessibility features or solutions.

Table 4.22: Strengths and Weaknesses of GenMyModel compared to other assessed tools

Strengths	Weaknesses
V1: One of the three tools, which resulted in satisfactorily fulfilling the need for suitable resizing text, images, or other significant components for *TG* and *CM*.	**V2:** One of three tools that are not able to meet the condition of allowing customizability of fonts and colors for *TG* and *CM* at all.
V3: Only tool that provides positive outcomes for being able to create suitable text-to-speech synthesis for *TG* and *CM*, as it satisfactorily fulfills this condition for *TG* and also partially for *CM*.	**P3:** One of three tools which are not fulfilling the criterion of providing suitable text alternatives for the significant components and other elements for *TG* and *CM*.
CLN2: One of the two tools resulting in fully supporting the usage of consistent labels from the perspective of *TG*.	**P6:** The only assessed tool, which does not, either fully or partially, meet the criterion of consisting of large clickable areas for both *TG* and *CM*.
CLN4: One of the two tools which is at least returning a positive outcome for this criterion for *TG* and *CM* as it partially fulfills the condition of providing different navigational mechanisms for both.	
P2: One of the two tools which fully satisfies *TG* and *CM*, the criterion of providing sufficient time limits to react for the user.	

Gliffy

According to the tool assessment, Gliffy fulfills the majority of the evaluation criteria for the disability types CLN and physical but satisfies visual disabilities partially satisfactorily. The overall results indicate that Gliffy performs well, with only some minor flaws, similar to the other evaluated modeling tools on average. Also, in this case, the main focus was not to create a modeling tool that should be accessible specifically; therefore, Gliffy does not provide any additional accessibility features or solutions.

Table 4.23: Strengths and Weaknesses of Gliffy compared to other assessed tools

Strengths	Weaknesses
V3: One of the two tools that provide positive outcomes for creating suitable text-to-speech synthesis for *CM*, as it partially satisfies the criterion.	**P2:** One of two tools which are not fulfilling the criterion of providing sufficient time limits to react for *CM* at all.
P3: Only tool which satisfyingly meets the criterion of providing suitable text alternatives for the significant components and other elements for both *TG* and *CM*.	

diagrams.net

This web modeling tool stands out as one of the top performers among the ten evaluated tools. Upon examining the outcomes for each of the three disability types, it can be observed that diagrams.net successfully provides adequate support for all of them, as it fulfills the majority of the provided evaluation criteria. The tool provides no additional accessibility features than those assessed via the evaluation criteria.

Table 4.24: Strengths and Weaknesses of diagrams.net compared to other assessed tools

Strengths	Weaknesses
V2: Only tool which is satisfactorily fulfilling the need for customizable fonts and colors for *TG* and *CM*.	**P2:** One of three tools, which only partially meets the criterion of providing sufficient reaction time limits for *CM*, whereas the majority can meet the criterion fully.
CLN2: One of three tools that fully support consistent labels for *CM*.	
CLN4: One of the two tools which is at least returning a positive outcome for this criterion for *TG* and *CM* and also the only tool, which is fulfilling the condition of providing different navigational mechanisms at a fully satisfactorily level for *TG* and *CM*.	
P4: Only tool which meets the criterion of having efficient visual and non-visual orientation or navigational cues fully for *CM*.	
P5: Only tool which meets the criterion of providing a logical navigational mechanism and page functions fully for *CM*.	

Creately

According to the tool evaluation, Creately satisfactorily fulfills the majority of the disability support criteria for CLN, partially for visual, and does not fulfill it for physical disabilities. Creately does not provide any additional accessibility features.

Table 4.25: Strengths and Weaknesses of Creately compared to other assessed tools

Strengths	Weaknesses
None of the evaluation criteria stood out significantly compared to the others.	**P2:** One of three tools, which only meets the criterion of providing sufficient reaction time limits specifically for *CM* only partially, whereas the majority is capable of meeting the criterion for *CM* fully.
	P5: Only tool which does not meet the criterion of providing a logical navigational mechanism and page functions fully specifically for *TG*.
	P6: The only assessed tool, which partially meets the criterion of consisting of large clickable areas for *TG*, whereas the majority is satisfactorily fulfilling it for the same case.

Cacoo

This tool satisfactorily supports CLN disabilities, and partially visual disabilities, and does not support physical disabilities according to the pre-defined evaluation criteria. Cacoo does not have any additional accessibility features included.

Table 4.26: Strengths and Weaknesses of Cacoo compared to other assessed tools

Strengths	Weaknesses
V1: One of the three tools, which resulted in satisfactorily fulfilling the need for suitable resizing text, images, or other significant components for *TG* and *CM*.	**P3:** One of three tools that are not fulfilling the criterion of providing suitable text alternatives for the significant components for *TG* and *CM*.
	P6: One of two tools that do not meet the criterion of consisting of large clickable areas at all, specifically for *TG*.

UMLetino

This textual modeling tool is partially fulfilling the support for visual disabilities. Further, UMLetino can fully provide support for CLN and physical disability, as it fulfills most of the criteria. There are no additional or different accessibility functionalities provided by this particular tool. However, as the only textual modeling tool in this assessment, it is evident that its structure has an advantage, especially for visual disabilities support (usage of text-to-speech, speech-to-text) and physical disabilities support (predominantly keyboard usage).

Table 4.27: Strengths and Weaknesses of UMLetino compared to other assessed tools

Strengths	Weaknesses
V1: One of the three tools, which resulted in satisfactorily fulfilling the need for suitable resizing text, images, or other significant components for *TG* and *CM*.	**V2:** One of three tools that are not able to meet the condition of allowing customizability of fonts and colors for *TG* and *CM* at all.
V3: One of the two tools that entirely fulfills the need for suitable text-to-speech synthesis specifically for *TG*.	**CLN4:** Only tool which is not fulfilling the condition of providing different navigational mechanisms for *TG* and *CM* at all.
CLN2: One of three tools that fully support consistent labels specifically for *CM*. One of the two tools that fully support the usage of consistent labels for *TG*. Only tool resulting in fully supporting the usage of consistent labels for both, namely for *TG* and *CM*.	**P2:** One of three tools, which meets the criterion of providing sufficient reaction time limits for *CM* only partially, whereas the majority is capable of meeting this criterion fully for *CM*.
P5: Only tool which does fulfill the criterion of providing a logical navigational mechanism and page functions fully specifically for *TG*.	
P6: One of two tools that do meet the criterion of consisting of large clickable areas for both *TG* and *CM*.	

Diagramo

For this specific tool, the evaluation satisfactorily fulfills the majority of the evaluation criteria for CLN disability support, partially for visual, but not for physical disabilities. Diagramo has no additional accessibility functionalities included.

Table 4.28: Strengths and Weaknesses of Diagramo compared to other assessed tools

Strengths	Weaknesses
CLN2: One of three tools that entirely fulfills the usage of consistent labels specifically for *CM*.	**CLN2:** One of the two tools resulting in not supporting consistent labels for *TG* at all.
	P2: One of two tools that are not fulfilling the criterion of providing sufficient time limits to react specifically for *CM* at all.
	P3: One of three tools that are not fulfilling the criterion of providing suitable text alternatives for the significant components and other elements for *TG* and *CM*.

miro

This modeling tool satisfies most of the evaluation criteria for physical and CLN disabilities, but partially for visual disabilities. Miro does not provide additional accessibility features. However, it is one of the modeling tools which specifically targets the improvement of accessibility and adds new updates and improvements with every new version. These steps include overall improvement to make it more accessible and better usable with

external and additional assistive technologies, improve the keyboard navigation and provide the user with better customizability possibilities[9].

Table 4.29: Strengths and Weaknesses of Miro compared to other assessed tools

Strengths	Weaknesses
None of the evaluation criteria stood out significantly compared to the others.	**CLN2:** One of the two tools resulting in not supporting the usage of consistent labels for *TG* at all.

BPMN.io

The evaluation results for this web modeling tool have resulted in not being able to support visual disabilities at all. However, it can provide support for CLN and physical disabilities at a satisfactory level according to the majority of the assessment criteria. Also, this tool does not include additional accessibility features.

Table 4.30: Strengths and Weaknesses of BPMN.io compared to other assessed tools

Strengths	Weaknesses
P2: One of the two tools which fully satisfies the criterion of providing sufficient time limits to react for *TG* and *CM*.	**V2:** One of three tools that are not able to meet the condition of allowing customizability of fonts and colors for *TG* and *CM* at all.
P6: One of two tools that do meet the criterion of consisting of large clickable areas for *TG* and *CM*.	

4.3.6 Results of the evaluation questions

The previously defined evaluation questions (cf. Section 4.2.3) are answered in the following paragraphs based on the findings:

- **EQ-1: Which accessibility barriers and limitations for people with the assessed disability types[5] do exist *most* frequently that would block or complicate the usage of the tools in the intended way?** It can be observed that overall there is a lack of customizability of font types, sizes, and colors (V2), especially for the provided *Tool Support & GUI*. Furthermore, most modeling tools do not provide a correct and sufficient text-to-speech synthesis (V3). This is unsurprising as, especially for models and their components, whose meaning is contained in the graphical components, it is challenging to transform that graphical information into natural text without the loss of the semantics [CCD20]. Furthermore, the structure of some of the assessed tools had minor flaws in providing clear, structured, and different navigational mechanisms (CLN4), specifically for *Canvas & Model*, which in comparison is provided and slightly better for *Tool Support & GUI*. Additionally, there is a clear gap regarding the full

[9]Miro Accessibility, `https://miro.com/accessibility-statement/improvements/`, (Access: 18.08.2023)

keyboard support (P1), as none of the assessed tools provides this. Lastly, it can be observed that, especially for the *Canvas & Model*, the text alternatives were not always sufficiently provided (P3).

- **EQ-2: Which accessibility barriers and limitations for people with the assessed disability types[5] do exist *least* frequently that would block or complicate the usage of the tools in the intended way?** The assessment shows that customizability regarding resizing different significant components for *Tool Support & GUI* and *Canvas & Model* is possible without any issues, therefore, can be adapted to the user's visual needs (V1). Further, most modeling tools have clearly structured content (CLN1), which is overall not confusing, and the user's interactions and the expected results are predictable (CLN3). For better understandability, most of the tools also provide supplements for text in the form of different illustrations (CLN5). Additionally, the majority of the tools do include efficient error correction options (P7), have satisfying reaction time limits (P2), and also do provide adequate visual and non-visual orientation and logical navigation mechanisms (P4), text-alternatives (P3), and adequately sized GUI components to avoid precise handling (P6).
- **EQ-3: Which accessibility features and functionality do the evaluated tools provide?** The observations show that most tools do not focus on explicitly improving accessibility. However, due to their overall structure and emphasis on enhancing usability and efficiency, they also partially contribute to improving the accessibility state [TBdR05]. The web modeling tool Miro is explicitly targeting to improve specific properties of the tool itself, which also results in better accessibility, such as better keyboard navigation, more customizability, and more in every new release (more details in Section 4.3.5). One of the evaluation subjects, Lucidchart, provides a certificate that confirms WCAG 2.1 at levels A and AA. Additionally, Lucidchart provides an individual accessibility feature for creating accessible PDFs of the content, which is suitable for screen readers and similar assistive technologies (more details in Section 4.3.5).

4.4 Conclusion

The outcomes of the conducted systematic literature review (cf. Chapter 2) and this chapter's tool assessment show a gap in disability-aware conceptual modeling research aside from the apparent increasing importance and relevance. In more detail, the literature review clarified that the research, especially in conceptual modeling, is not fully developed. The gaps in conceptual modeling research specifically exist regarding cognitive, learning, neurological, and physical disability support (cf. Figure 2.9). On the other hand, the tool evaluation described in this chapter had one outstanding evaluation criterion for physical disability support, which was not fulfilled at all from any of the assessed web modeling tools, which is the full-keyboard support (P1). Furthermore, the evaluation of physical disability support revealed that it had the highest number of unsatisfactory

tools compared to the other disability types. On the other hand, the support for visual and CLN disabilities showed sufficient assistance, except for minor shortcomings in some specific tools.

Due to the research gaps and the gaps found with the assessment, this thesis focuses on providing full-keyboard support in web modeling tools. The primary reason for selecting the focus on physical disability and developing a keyboard-only prototype is due to its relevance to web modeling tools. These tools heavily rely on mouse or pointing-device interactions, and incorporating efficient keyboard-only interaction would significantly enhance accessibility. Notably, existing web modeling tools have not adequately addressed this aspect, making it a critical gap to be addressed. A detailed description of the keyboard-only prototype can be found in Chapter 5.

CHAPTER 5

Keyboard-Only Prototype & Evaluation

5.1 Introduction

Ensuring proper support for individuals with physical disabilities is crucial, given the diverse spectrum of people who may encounter challenges when using a mouse or similar pointing device. This category encompasses those users, who are directly impacted by physical impairments and have limited fine motor skills, often including older users. As highlighted by the Web Accessibility Initiative (WAI), a truly accessible website should not only rely on mouse interactions. Instead, it should be designed to ensure the availability and accessibility of all functionalities via keyboard input. This approach serves a dual purpose: aiding individuals reliant on keyboard-based navigation and those who use assistive technologies like speech input, which mimic keyboard actions. This well-thought-out approach works well for many different physical disabilities, including a wide variety of types and levels of severity.[1].

When comparing the results of the systematic literature review (cf. Chapter 2), a research gap in the category of physical disabilities (cf. Figure 2.9) and the absence of full keyboard support for all evaluated modeling tools (cf. **P1** in Table 4.20) guided the development of a prototype in this area. It is essential to clarify that this prototype does not claim to remove all challenges faced by people with physical disabilities. However, it aims to reduce essential barriers and make things more inclusive significantly. This implemented prototype has an essential role as it forms the building blocks for the first contribution in this direction, thus positively influencing further research and implementation in this area.

[1] Introduction to Web Accessibility, https://www.w3.org/WAI/fundamentals/accessibility-intro/, (Access: 18.08.2023)

The keyboard-only prototype extends a Graphical Language Server Platform (GLSP)[2] based workflow diagram editor with new keyboard interactions. GLSP is heavily used in industry and academia to realize web-based modeling tools with advanced visualization and interaction features (cf. [DCLB22]). The newly provided keyboard interactions aim to interact with the web modeling tool to accomplish a basic workflow of creating, editing, and observing a model. Additionally, this prototype is also provided in a generalized form and published as part of the open source project through the Eclipse GLSP Client[3]. This contribution makes these accessibility features ready-to-use and adaptable for different editors and needs.

This chapter is organized as follows: In Section 5.2, a detailed explanation of the theory behind the design, implementation, and use of technologies will be provided. This understanding is essential for comprehending the upcoming sections, such as Section 5.3, where the conception part will be discussed. This section explains how the functionalities are implemented, the design decisions made, and how the modeling editors are enriched through these new interaction possibilities. Following that, in Section 5.4, the evaluation of the prototype will be presented. Finally, this chapter ends with the conclusion in Section 5.5.

5.2 Theory & Background

This section provides the theoretical background essential for the understanding and conception of the implemented prototype. Furthermore, the sections serve as informational guidance to improve understanding of the terms and provide guidelines for **Keyboard Accessibility** (cf. Section 5.2.2) and **Keyboard-Only Controls** (cf. Section 5.2.4)

5.2.1 Understanding Keyboard-Only Interactions: Importance and Implementation

In order to create a suitable prototype, it is essential to clarify what keyboard-only, sometimes referred to as full-keyboard support, means. Enhancing the support of full keyboard usage will improve the Keyboard Accessibility. A keyboard-only website or web application is given if all the provided actions and the navigation can be performed only using the keyboard, without needing a mouse or other pointing device [Gui]. Also, this is one of the minimum requirements of WCAG 2.1 for the lowest level A. More specifically, to fulfill this guideline at level A, it is necessary to not only provide operable content through a keyboard interface, but these interactions should not require specific timings for individual keystrokes. However, it should be noted that keyboard-only support does not mean it is forbidden to also interact with the web content via mouse or other input methods. Moreover, full keyboard support should exist in addition to the remaining input options. Even though the focus of this prototype is on physically disabled or impaired

[2]Eclipse: GLSP, `https://www.eclipse.org/glsp/`, (Access: 23.05.2023)

[3]Eclipse GLSP Client, `https://github.com/eclipse-glsp/glsp-client`, (Access: 02.08.2023)

individuals who find using the keyboard over the mouse easier, this kind of keyboard support also benefits users with other disabilities, such as people who are blind or have low vision, i.e., who cannot use devices requiring eye-hand coordination, like a mouse or find it challenging to recognize the pointer on the screen [4].

5.2.2 Keyboard Accessibility Guideline according to WCAG 2.1

In order to be able to design and define suitable keyboard shortcuts that are accessible and improve keyboard accessibility, it is necessary to gain knowledge about the accessibility guidelines for the keyboard. The success criteria, which is part of the Web Content Accessibility Guideline (WCAG) 2.1, **Guideline 2.1 – Keyboard Accessible** [W3C18a] are shortly summarized in the following Listing to provide an overview for the course of this section.

1. **Keyboard** - *Level A* [4]: As mentioned earlier (cf. Section 5.2.1), this criterion ensures that all content functions can be used with a keyboard, without time limits or complicated key combinations to interact with web content. There are exceptions for functionalities, such as handwriting as input or similar interactions that rely on the user's movement. Additionally, providing full keyboard support should not forbid the usage of pointing devices, and their existence is allowed, as the keyboard-only property is seen as an addition and not a replacement for other input options.

2. **No Keyboard Trap** - *Level A* [5]: This guideline ensures that users are not trapped in keyboard focus on a component within a section or anywhere on a web page. It should be easily possible, to set the focus on a component and take the focus away from the same component without any issues using only the keyboard interface. Those traps are especially a hurdle for keyboard-only users as if the focus is once trapped, there is no other possibility to come out of this trap using only keys. Additionally, moving the focus away from an element should not require the usage of any complex shortcuts than using the *Arrow*, *Tab*, or other standard exiting keys.

3. **Keyboard (No Exception)** - *Level AAA* [6]: This success criterion builds up on the first criterion of this listing, **Keyboard** - *Level A*. All interactive elements and features must be entirely usable with a keyboard without any exceptions. This means there should be no instances where keyboard navigation is limited or restricted due to design choices or technical limitations. The main difference

[4] Keyboard (Level A), `https://www.w3.org/WAI/WCAG21/Understanding/keyboard.html`, (Access: 03.08.2023)

[5] No Keyboard Trap (Level A), `https://www.w3.org/WAI/WCAG21/Understanding/keyboard-no-exception.html`, (Access: 03.08.2023)

[6] Keyboard (No Exception) (Level AAA), `https://www.w3.org/WAI/WCAG21/Understanding/keyboard-no-exception.html`, (Access: 03.08.2023)

between the first and this criterion lies in the scope of their requirements, as the first success criterion **Keyboard** - *Level A* focuses on ensuring operability without specific timings, while the current success criterion emphasizes the complete absence of exceptions to keyboard operability.

4. **Character Key Shortcuts** - *Level A* [7]: This success criterion applies if character key shortcuts are used, i.e., key shortcuts activated through (single) characters. If the shortcut includes characters, numbers, or other symbols, it should be ensured that at least one of the following conditions is fulfilled:

   a) **Turn Off:** The key shortcuts can be turned off to avoid accidental triggering of features or functions when not needed.

   b) **Remap:** The key shortcuts can be remapped to other keys to choose mappings according to user needs or the editor's capabilities, i.e., if the key shortcuts are already occupied.

   c) **Active:** The key shortcut is only active and callable if the according component or element is focused. This property avoids mistakenly calling a feature or function.

   Using single-character shortcuts is easy and efficient for most disabled users, but there is a limitation, especially for users who use speech-input mechanisms. In this case, single characters are used for other commands, like navigation. Therefore, this could introduce the potential for shortcuts to conflict or overlap, leading to unintended interactions or inaccuracies.

5.2.3 Best Practices: Defining Keyboard Shortcuts

The goal of designing keyboard shortcuts is to enhance user productivity and efficiency. Some best practices for designing appropriate keyboard shortcuts according to different sources are as follows [8] [9] [W3C18a]:

1. **Understand User Needs & Context:** It is essential to identify and consider the context of the most frequently used actions or typical workflows of the application while designing keyboard shortcuts to ensure the usability and suitability of the provided key mappings.

[7] Character Key Shortcuts (Level A), https://www.w3.org/WAI/WCAG21/Understanding/character-key-shortcuts.html, (Access: 03.08.2023)

[8] Guidelines for Keyboard User Interface Design, https://learn.microsoft.com/en-us/previous-versions/windows/desktop/dnacc/guidelines-for-keyboard-user-interface-design, (Access: 03.08.2023)

[9] Developing a Keyboard Interface, https://www.w3.org/WAI/ARIA/apg/practices/keyboard-interface/, (Access: 03.08.2023)

2. **Consistency & Standards:** A wide variety of keyboard shortcuts are globally referenced and used and already have an intended meaning, i.e., Copy-Paste as *'CTRL'+'C'* & *'CTRL'+'V'*. Avoid remapping these established conventions and standards for common actions. Furthermore, key combinations already occupied by a specific platform or system it is being used on should also be avoided.

3. **Intiutive Design:** In general, as already stated in **Consistency & Standards**, the design of the critical shortcuts should be logical and intuitive, especially avoiding complex or arbitrary key combinations that may confuse some users. Additionally, using Mnemonic keys would improve the understandability and memorability of the shortcuts, e.g., *'N'* for New or *'S'* for Save.

4. **Accessibility:** One of the essential characteristics to be aware of while designing keyboard shortcuts is to include keyboard accessibility. Not only the existence of key shortcuts is essential but also to provide alternative ways to access functionality (e.g., through menus or toolbars).

5. **Prioritiziation & Overloading:** Assigning shortcuts to the most frequently used actions should be prioritized. Less common actions should have less prominent or no shortcuts depending on the context to avoid overloading with too many combinations. Any actions, which are most commonly triggered together or part of a group-related action should be mapped to similar and consistent key combinations.

6. **Visual Cues & Documentation:** Users should be able to recognize which components or elements on a website or application can be activated via key shortcuts by providing appropriate visual cues. Additionally, providing a keyboard shortcut reference is essential as a guide about all existing combinations. Users should be aware of the existing possibilities.

5.2.4 Exploring Keyboard-Only Requirements & Implementation Insights

The previous sections have clarified what keyboard-only support is and how keyboard shortcuts should be designed and provided in general. The open question now is to detect those requirements to design a keyboard-only application or, in that case, a prototype. This section aims to determine which requirements need to be fulfilled to have a sufficient prototype with keyboard-only functionalities. Especially in this implementation process defining appropriate interactions require more awareness; as per the definition of keyboard-only, it should be possible to accomplish typical workflows of the given website or application only via the keyboard. In this case, this means that at least the most frequently used interactions or the most typical workflow of a modeling editor should be able to carry out solely with the keyboard, and designing this requires more awareness.

Implementation Insights: When creating websites, HTML elements already offer built-in support for keyboard interactions, as HTML itself is keyboard-only ready. Using those browser-provided functionalities and following the pre-defined semantic HTML markup allows proper keyboard interactions to navigate through pages or interact with the existing controls.

However, there is a need for custom functionality, e.g., if there is no equivalent HTML element. In order to create custom functionality, it is essential to optimize the functionality specifically for keyboard interactions. Throughout this customization process, it can quickly happen that the basic needs of accessible keyboard interactions are not fulfilled or disregarded by the developer, leading to a poorly accessible keyboard interface [Gui].

Therefore, the prototype developed in this thesis is based on the built-in functionalities but also on the keyboard accessibility developer guidelines by Web Accessibility In Mind (WebAIM) [Web22b] and Mozilla Developer Network (MDN) [MDN23], to achieve customizability without disregarding keyboard accessibility. The following excerpt of the requirements was considered to fulfill the needs of a keyboard-only web modeling editor. The requirements listed below are partially overlapping with the best practices of defining keyboard shortcuts as mentioned earlier (cf. Section 5.2.3) and also with the Web Content Accessibility Guideline (WCAG) success criteria for keyboard accessibility (cf. Section 5.2.2):

1. **Focus**: Only interactive elements should be focusable, e.g., buttons, links, input fields, and custom interactive elements. This requirement is used to avoid leading users to elements that cannot be used to interact with the given application, mislead or trap the users in an unwanted state.

2. **Navigation**: The application should provide a mechanism to navigate through the application's content. The order in which the interactive elements retrieve the focus from the keyboard matters. This should be logical and intuitive. Additionally, the focus from an element can be moved and is not trapped or locked there (cf. [W3C18b]).

3. **Shortcuts:** The selected shortcuts should allow easy and fast access to menus and functionalities. The shortcuts should be meaningfully designed, especially for frequently used actions within the application. It is essential to ensure that these shortcuts will not conflict with standard keyboard shortcuts used by the operating system or assistive technologies (e.g., *'CTRL'+'C'* for Copying, *'CTRL'+'F'* for Searching). Additionally, there should exist a possibility to discover and learn about available keyboard shortcuts (e.g., help or documentation).

4. **Visibility:** Depending on the type of application, there should exist visual and non-visual orientation cues, page structure, and other navigational aids to help the user with better orientation and avoid misleading interactions. A visible focus should be ensured. This allows users to understand where they are in the application

and which element will receive their keyboard input next. An example of a focus indicator is highlighting or an outline.

5. **Consistency & Predictability:** Any application's interaction and functionality should provide the user with a consistent and predictable experience to avoid unexpected changes in behavior or focus that can confuse or disorient users. Additionally, inconsistent and unpredictable behavior would lead to a higher learning curve and an unsatisfactory user experience.

6. **User Feedback:** In addition to providing keyboard shortcuts, a mechanism should display user feedback or information in real-time (e.g., short notifications about turning functionality on/off) to keep the user informed about their interactions and handling. This notification should be visible and not interfere with application content or navigation.

5.2.5 Eclipse Graphical Language Server Platform

Eclipse Graphical Language Server Platform (abbrev. Eclipse GLSP) is widely utilized in both industry and academia to develop web-based modeling tools that offer advanced visualization and interaction capabilities [BLO23, MB23]. In more detail, GLSP is a client-server framework for building web-based diagram editors and engaging developers to contribute as the framework is extensible and the whole source code is available as open-source and therefore delivering constant technological changes and improvements. GLSP follows a pattern similar to the Language Server Protocol but is adapted explicitly for graphical modeling and diagrams. The developers create modern web-based diagram editors. The server handles tasks like loading and editing diagrams using graphical language rules. Through this, diagram editors can easily be integrated into various tools like VS Code, Eclipse Theia [10], and standalone or web apps.

As previously mentioned, this prototype extends the Workflow Diagram Editor of GLSP. Section 5.2.5 will provide more detailed information about the workflow example language server, which is essential to understand the upcoming feature descriptions and illustrations about the accessibility functionalities. However, the provided accessibility features are not only limited to be used specifically on the Workflow Editor, as the generalization of the functionalities and the nature of GLSP being integrable to various other tools allows to adapt and include the accessibility features also to various other tools, e.g., like Eclipse Theia (cf. Figure 5.5).

A more detailed explanation of the architecture, the provided servers, integrations, and contributions of Eclipse GLSP would exceed the scope of this thesis. For more detailed and newest information, the website [11] or the repository [12] is accessible.

[10] Eclipse Theia Integration, `https://github.com/eclipse-glsp/glsp-theia-integration`, (Access: 06.08.2023)
[11] Eclipse GLSP, `https://eclipse.dev/glsp/`, (Access: 03.08.2023))
[12] Eclipse GLSP Repository, `https://github.com/eclipse-glsp/glsp`, (Access: 03.08.2023)

Workflow Diagram Language Server

The GLSP workflow diagram server is an example implementation of a GLSP server and is most commonly used to demonstrate existing GLSP features. Using the modeling editor, it is possible to create a simple workflow diagram, a flow chart diagram with several node and edge types. The created model can also be compared to a UML activity diagram, which describes the flow of activities necessary to finish a task. The reason for using the workflow diagram editor is, in most cases, that it is a consistent example that is provided by all GLSP components but is also capable of providing integrations to other IDE platforms (e.g., Theia, VSCode, and more). More details about the workflow diagram examples' source code can be found in its code repository [13].

5.3 Conception

This section will present a comprehensive overview of the integrated accessibility features designed to facilitate a keyboard-only modeling experience and enhance inclusivity, particularly for individuals with physical disabilities. The keyboard interactions assigned for these functionalities must be intuitive, easy to understand, and handle. The features provided can be classified into three main categories: **CRUD Features** (cf. Section 5.3.1), **Model Exploration** (cf. Section 5.3.2), and **Model Navigation** (cf. Section 5.3.3). An overview of the provided accessibility features can be observed in Figure 5.1.

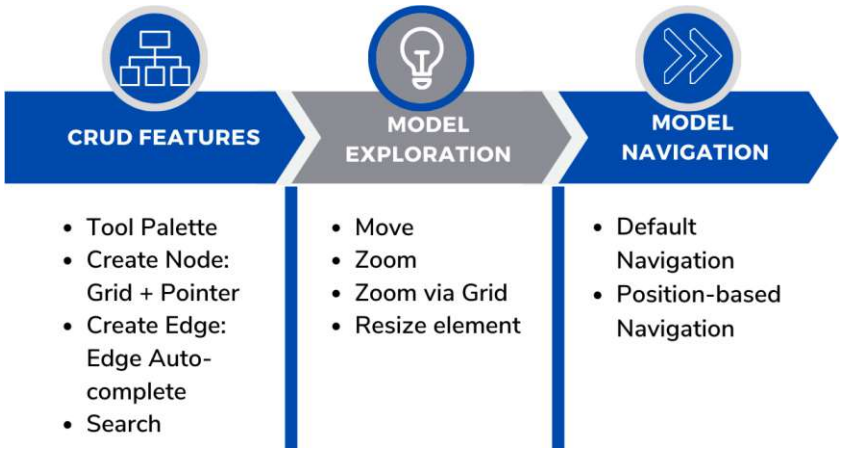

Figure 5.1: Overview of the provided accessibility features with the prototype

In the following sections, each category will be elaborated on in detail to provide a comprehensive understanding of its functionality and benefits.

Lastly, in Section 5.4, an evaluation of the newly introduced accessibility features will be provided. This section briefly compares how the features tackle and enhance the existing

[13] Eclipse GLSP Server - Workflow Diagram Example, `https://github.com/eclipse-glsp/glsp-server#workflow-diagram-example`, (Access: 03.08.2023)

accessibility challenges mentioned earlier in the tool evaluation and how it addresses those issues. This includes an analysis of how and to what extent the feature is fulfilling the Keyboard Accessibility Guidelines (cf. Section 5.2.2), best practices of the definition of keyboard shortcuts (cf. Section 5.2.3), the Keyboard-Only Requirements set earlier in this chapter (cf. Section 5.2.4) and against the average results of the tool assessment regarding physical disability support of modeling tools (cf. Table 4.20).

5.3.1 CRUD Features

This section deals with the CRUD Features (**C**reate, **R**ead, **U**pdate, **D**elete), which make up those four basic operations a software application should be able to perform. Additionally, two basic editor improvements for better user satisfaction, such as visual aids and notifications, are also described in Section 5.3.1.

As in this case, the newly introduced accessibility features are extending the workflow diagram editor by GLSP, some of the fundamental CRUD interactions essential for web modeling editors already exist. However, this prototype intends to enhance the accessibility of the existing CRUD features by providing innovative extensions or other components that support the base operations in terms of accessibility. A modeling editor should be able to accommodate at least two use cases, which are not entirely disjoint but instead have a distinct focus: creating models from scratch and modifying partially existing models [SG15].

The workflow editor by GLSP shares a similar structure and components as the commonly used web modeling tools. In order to create elements, a panel with creatable elements, such as nodes and edges, and other interaction possibilities, such as frequently used actions, is provided. This menu panel is known as the **tool palette** (cf. Figure 5.2). The tool evaluation explained previously in this thesis (cf. Chapter 4) has shown that creating model elements requires the user to either **(i) select the element to be created from the panel, choose and click on the desired position on the canvas** or to use **(ii) Drag & Drop** to move the desired elements from the menu panel on to the canvas.

The issue with solution **(i) Creation via Mouse-Click** is that there is currently no existing option that allows carrying out the whole process of *selecting*, *locating*, and *creating* the element only via key shortcuts and without relying on a mouse or other pointing device throughout the whole creation process. On the other hand, the solution **(ii) Drag & Drop** is already criticized through the tool assessment. The reason for that is that the user most probably needs to handle selecting and dragging the element across the screen precisely (similar to *P6* in Table 4.20) and especially dragging the element to the desired position can be challenging due the extensive movement of the mouse. Additionally, in some cases, it can also be a time issue (cf. *P2* in Table 4.20) if the user is slow and e.g. accidentally releases the object, which can even be significantly more challenging and frustrating for the modeler.

It is evident that in both cases, there can occur challenges for physically disabled users. Obviously, no keyboard-only possibility is provided, as the selection and creation process

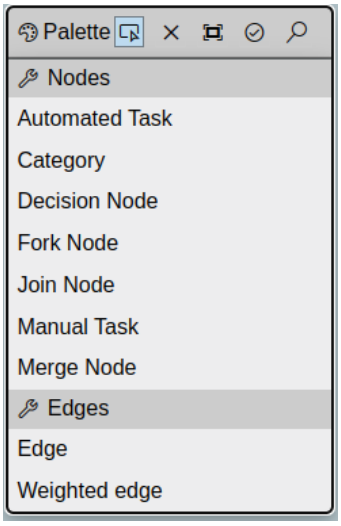

Figure 5.2: Workflow Diagram Editor: Tool Palette

heavily relies on the mouse. Therefore, the goal is to accommodate two main challenges with this prototype's features.

The new functionalities need to address the following issues:

- A mouse-independent way to interact with the tool palette and its content, such that selecting elements or calling frequently used interactions do not need handling via any pointing device.
- A mouse-independent way to position on a desired canvas location to create model elements without any pointing device. Additionally, the user should not lose the freedom of selecting the element's position as desired.

The newly introduced keyboard-only solution for creating a new model consists of two main features, which will be described in the following:

Accessible Tool-Palette

To enhance accessibility, the tool palette and its header menu are now accessible through single-key shortcuts. As illustrated in Figure 5.3, the character keys can be used to activate the entries in the tool palette (marked as **1**), while numeric keys are used for the header menu (marked as **2**). These shortcuts can quickly turn on or off via the key combination *'ALT'+'P'*. Using single-key shortcuts allows easy access to these frequently used actions, eliminating the need for mouse movements.

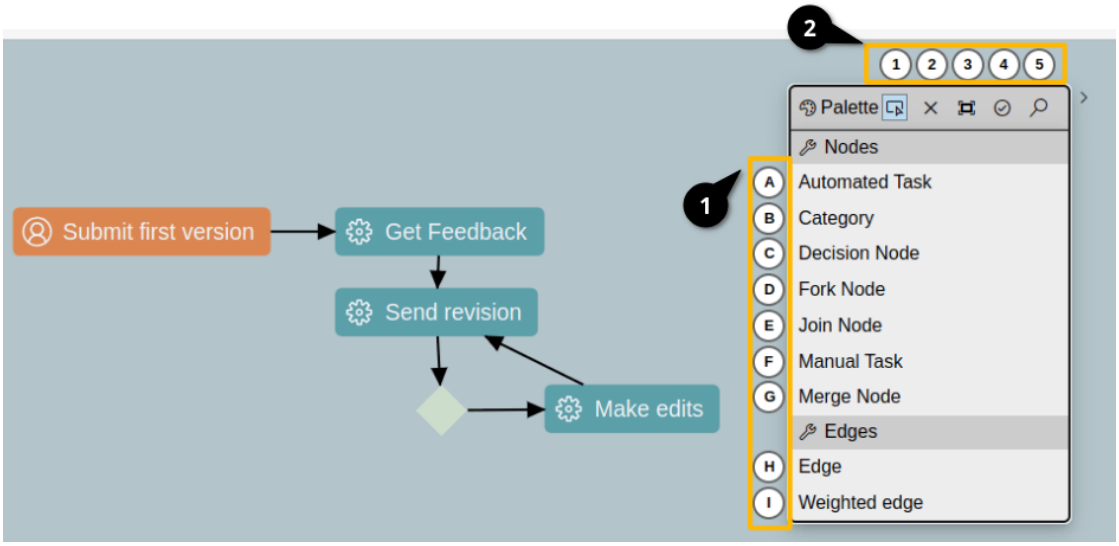

Figure 5.3: Accessible Tool Palette: Palette Entries (1) and Header Menu (2)

Create Node: Grid & Pointer

After selecting a model element to be created, the next step is to find a suitable position on the canvas, move the mouse to the position, and confirm with a mouse click. There is no known keyboard alternative for the whole process in the existing modeling tools.

In order to eliminate the need for pointing devices when positioning precisely on the large canvas area, two new elements were introduced: the **Grid** and **Pointer**.

After selecting a node in the tool palette, a **Grid** turns visible, where the modeler can choose the starting point of the pointer (i.e., a cursor) on the screen using the numeric keys. The **Grid** helps to position over a large area only by using a single numeric key. The canvas is split into **nine** equally sized boxes, and by choosing the corresponding box number, one can break down the location into a smaller area (cf. Figure 5.4).

Even though there is no ready-to-use solution on existing web modeling tools available for providing a Grid, the state-of-art of the selected publications (cf. Chapter 3) has shown that there was research done in using a Grid-based structure in the creation of class diagrams [WSGK20] and also in coding environments [EUHMB22].

Wildhaber et al. [WSGK20] aims to improve the accessibility of UML class diagrams for visually impaired people. In this case, the solution was a 3D printed Grid used as an overlay for mobile devices, where each tile of the grid represents a diagram class. Using custom gestures enables interactions between classes. On the other hand, Ehtesham et al. [EUHMB22] introduced a Grid structure to simplify coding for blind individuals. The Grid Editor enhances coding for blind programmers with a 2D grid. Rows represent code lines, and columns show indent levels, making navigation easier. It visually reflects code complexity and guides with special cells. The design prevents errors, ensures

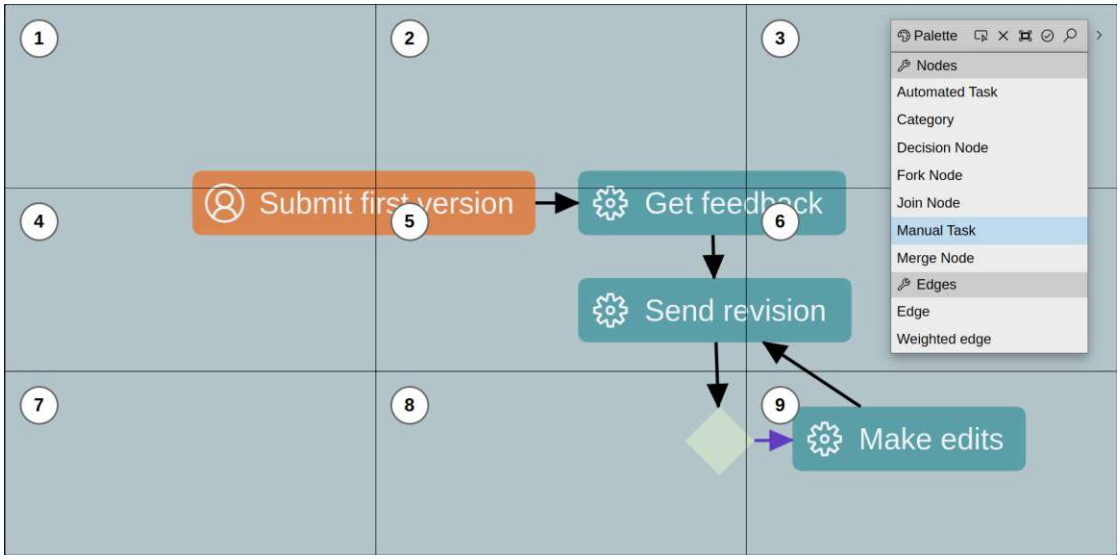

Figure 5.4: Create Node: Grid

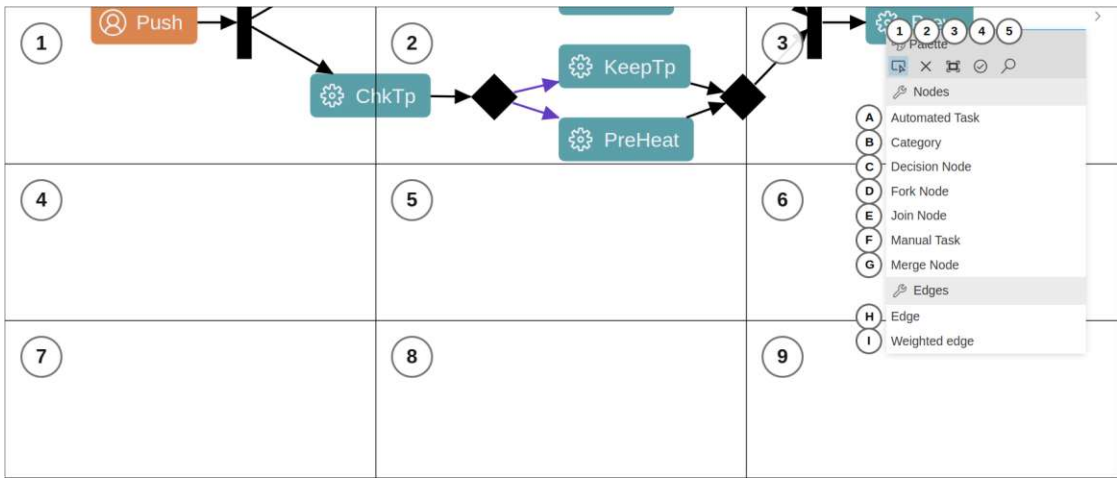

Figure 5.5: Example for Eclipse Theia Integration: Accessible Tool Palette & Grid

focused work, and empowers blind programmers, boosting their coding efficiency and independence. Both solutions show that using a Grid helps to simplify the locating positions on visual surfaces, as each Grid-Box can be easily accessed through its position or number. Additionally, the Grid helps to allow the presentation of complex information in a structured manner.

Therefore, using the Grid in this prototype provides an easy way to locate the cursor on a specific smaller canvas area without the use of a pointing device, only by numeric keys. However, as only positioning the model elements to the respective nine boxes would not be enough and would limit the user in creating multiple components, a second component

for a more fine granular positioning is needed.

The second component is the so-called **Pointer**, displayed after the user selects the desired Grid Box. This pointer is positioned right in the middle of the selected box, and the *arrow* keys can be used to choose the new element's location. By using the **Grid & Pointer** combination, first, the user can delimit the large area to a smaller one with one click, and secondly, allows the user customizability by having a more precise selection with the help of the **Pointer**. By clicking *Enter*, the node creation will be finalized, all without the use of a pointing device (marked as **1** in Figure 5.6).

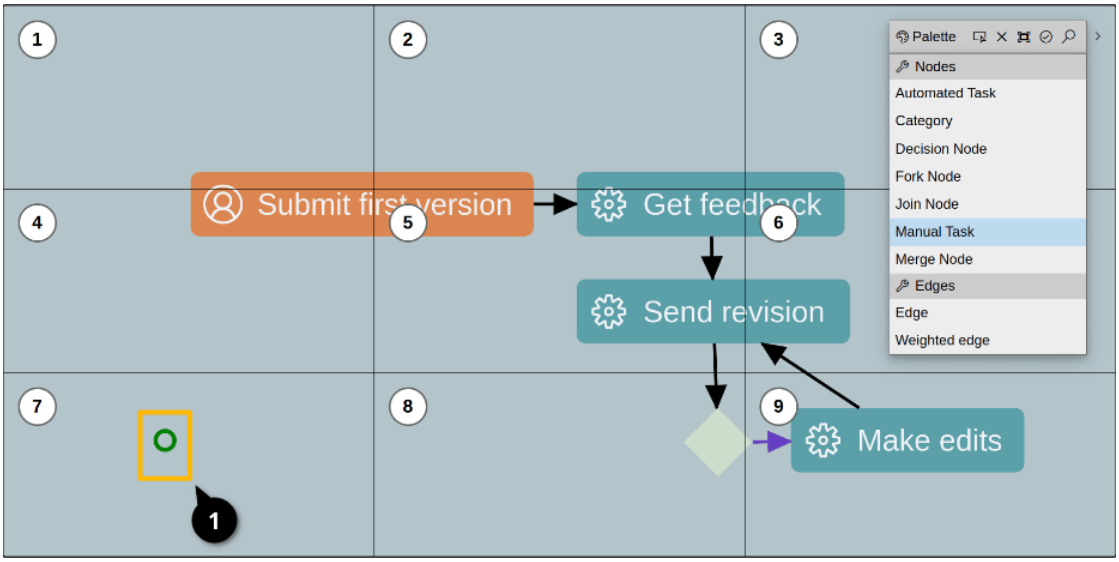

Figure 5.6: Create Node: Pointer (1)

Additionally, the **Pointer** is also capable of providing visual feedback on valid (marked as **1** in Figure 5.7) or invalid actions (marked as **2** in Figure 5.7). This is especially important to guide users in the process of element creation and to avoid any incorrect handlings. As also stated in the tool assessment of the web modeling tools, providing error correction options (cf. *P7* in Table 4.2.6) is necessary, but this **Pointer** proactively avoids unallowed interactions by providing visual feedback on valid or invalid positions, e.g., to avoid overlapping of nodes or edges. This preventive measure is essential for physically disabled individuals, who often face challenges in performing specific actions due to limitations. By preventing invalid interactions, the technology caters to their specific needs and eliminates potential frustrations or barriers that might arise from accidental or unintended actions. This proactive approach enhances usability and inclusivity, empowering physically disabled users to navigate and interact with the system more effectively and fostering a sense of independence and equal participation.

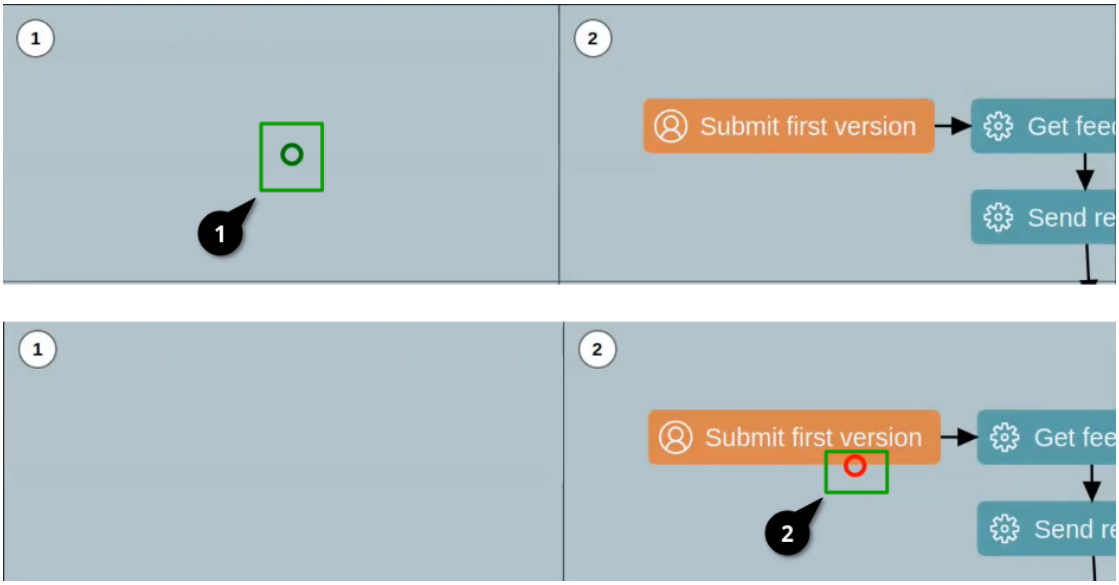

Figure 5.7: Create Node: Pointer for Valid (1) and Invalid (2) Positions

Search in a model

The search functionality is an essential feature in modeling tools; as the model grows, the need for quick searching and finding specific elements will become more necessary and it will help to get a clear understanding and better interaction with the provided model. Most modeling editors provide search possibilities within the model itself but include a typical search, i.e., only searching for free text is possible. Although free text search is one of the key aspects of a search feature, thinking about the use cases and context of modeling tasks, finding elements of a type, group, or similar model-related characteristic is often needed for quick and efficient exploration and interaction with the existing model.

Therefore, the new search feature of this prototype should be able to find text within the model's elements and also search for topics related to the model for a better user experience. The implemented search field is callable via the standard shortcut *'CTRL'+'F'*, and it is possible to search for any free text but also for specific labels, nodes, or edge types of the used model type (marked as **1** in Figure 5.8). Depending on the user input, the matching elements will be shown as a list added to the search field to iterate through the list (marked as **2** in Figure 5.8), which then will be highlighted accordingly in the model (cf. Figure 5.9).

Furthermore, the search field is enriched with the following features:

Auto-Complete: The search functionality also includes a code completion mechanism. The code completion provides keywords and identifiers of elements that are syntactically allowed at the current cursor position. This means that as the user is typing, the editor

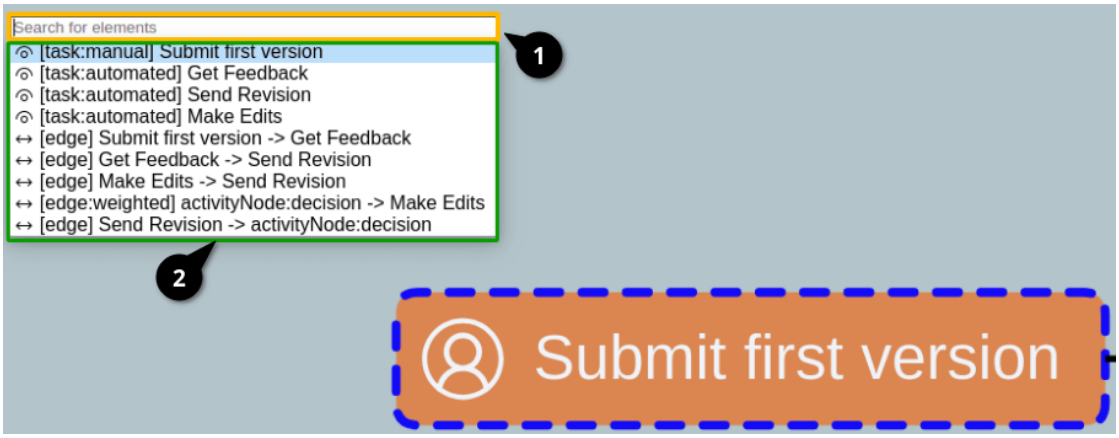

Figure 5.8: Search Functionality: Search Field (1) and List of matching elements (2)

will suggest possible completions based on the context, making it easier for the user to write valid code. Seifermann et al. [SG15] also have envisioned a so-called *modeling-specific search* to allow searching based on specific features of the model, enhancing the search process's value and efficiency. By including the possibility of searching for, e.g., node types, the search feature of this prototype aligns with this vision. Furthermore, an autocomplete feature in web tools can assist physically disabled users by speeding up input, reducing errors and enhancing their overall experience.

Marking of the model elements: Every non-matching element to the search query becomes transparent after the user has provided a user input to the search field. This marking ensures easy identification of the searched elements, especially useful for large and complex models. As observable in Figure 5.9, the matching elements are outstanding compared to the rest of the model and help the user identify the desired information. This enhancement streamlines the navigation and interaction within web tools, offering particular benefits to physically disabled users who might encounter difficulties when dealing with intricate layouts.

Create Edge: Edge Auto-Complete

One further typical action for model creation is the creation of connections or relations between nodes. Depending on the diagram type, various relationships between nodes exist. Most web modeling tools rely on **(i) dragging a connection from the source element to the target element**. Sometimes, alternatives are provided, such as **(ii) selecting the connection and then marking the source and target element via mouse-click**. However, both solutions rely heavily on the mouse or any other pointing device. Moreover, dragging a connection proves to be even more challenging, as demonstrated in the tool evaluation and previously explained for node creation (cf. Section 5.3.1). The challenging issue is here to find a simpler but understandable solution

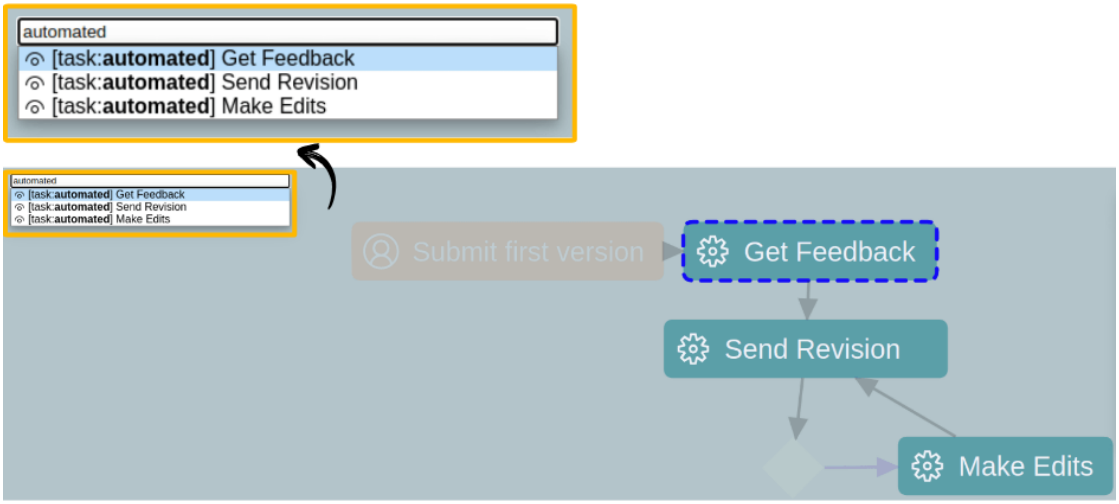

Figure 5.9: Search Functionality: Non-Matching Elements are marked transparently

for the edge creation process and to find suitable key alternatives for selecting source and target elements.

The keyboard alternative for this interaction is designed and implemented in the following way. After selecting the edge action in the tool palette as previously described (cf. Section 5.3.1), a new search field will be displayed. As illustrated in Figure 5.10 the search field prompts the user to search for a **source element** (marked as **1**). After searching and confirming the element, the second search field will be provided to choose and confirm the **target element** (marked as **2**). As a result, this will add the chosen connection between the source and target element without the need for a mouse.

The essential feature here is that this search mechanism for edge creation relies on a **Smart Search**, which means that it only displays the valid source elements for the chosen edge and only the valid target candidates for the selected source element. As a comparison, in Figure 5.10, the standard **Edge** Type was used. The possible *source* or *target* elements were returned accordingly. In contrast, in Figure 5.11, the **Weighted Edge** type was selected on the same diagram, which returned a smaller set of possible elements for the *source* and *target* elements.

The advantage of this **Smart Search** is that it avoids invalid interactions proactively. As already explained for the validation mechanism of the pointer in creating a node (cf. Section 5.3.1), it is helpful to prevent unallowed interactions for physically disabled users because it makes their experience smoother. This way, they will not face unnecessary obstacles or frustrations, ensuring more accessible and independent use of the technology.

Reading, Updating & Deleting a Model

The basic operation *Reading* of a Model is, in that case, covered by the **Search** functionality (cf. Section 5.3.1), as this serves the corresponding model elements to the user for

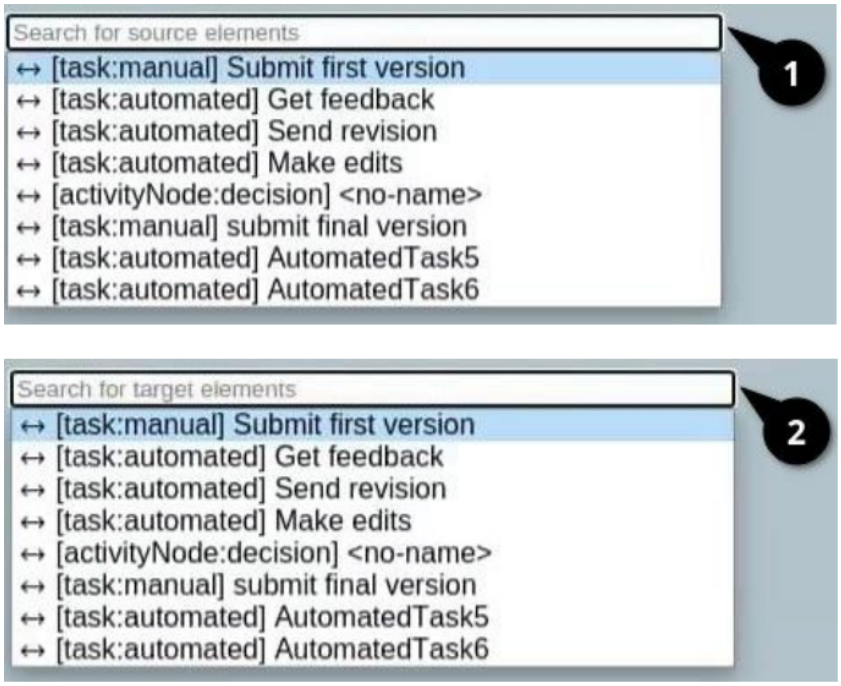

Figure 5.10: Create Edge: Search & Select **Source** Element (1) and **Target** Element (2)

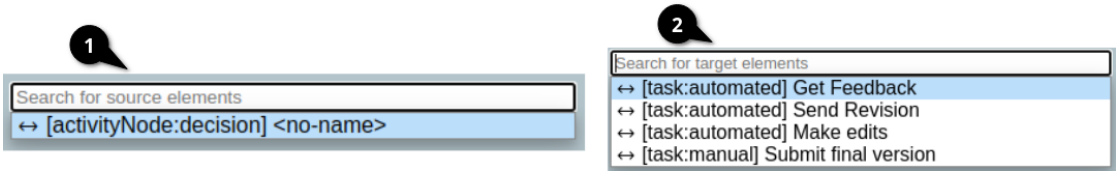

Figure 5.11: Create Weighted Edge: Search & Select **Source** Element (1) and **Target** Element (2)

further handling. The remaining two basic operations of CRUD features, *Update* and *Delete*, are indirectly covered by this prototype's overall functionalities or were already implemented in the framework itself. Updating a model includes operations like adding nodes (cf. Section 5.3.1) or edges (cf. Section 5.3.1), renaming (already implemented and possible via key *'F2'*) or resizing (cf. Section 5.3.2) those model components. Deleting parts of the model is already implemented and possible with the commonly used 'Delete' keys. In order to carry out each of the tasks mentioned above, the initial step is to select the node or edge to be updated or deleted. Therefore, an essential feature here is to select the elements to be modified, and this is possible via keyboard through the Search-Functionality (cf. Section 5.3.1).

General Editor Helpers

Keyboard Shortcut Documentation: As seen in Figure 5.12, a comprehensive menu is provided, which includes all existing key shortcut combinations and their meanings. Providing this list of keyboard shortcuts offers users a quick and convenient reference and is all the more important for keyboard-only users like physically disabled individuals. However, not only the existence of the reference is essential. Almost all assessed modeling tools did provide some key reference, but in some cases finding and activating this reference was not quickly or easily possible, e.g., being located in some drop-down menus, uncommon shortcuts for activation, and similar issues. In order to allow users easy access and a comprehensive key shortcut list, this overlay can be activated at any time using the key combination *'ALT'+'H'*. It can be disabled, with the same shortcut, or with common exiting shortcuts, like *Escape*. However, there is no problem using the modeling editor when the shortcut dialog is open. It does not get in the way of other actions or cover the whole canvas, thanks to its intentionally placed position at the canvas edge. This can be especially helpful for newcomers to the editor who need time to learn the shortcuts. The source code provides an easy registration system for any new shortcut combinations.

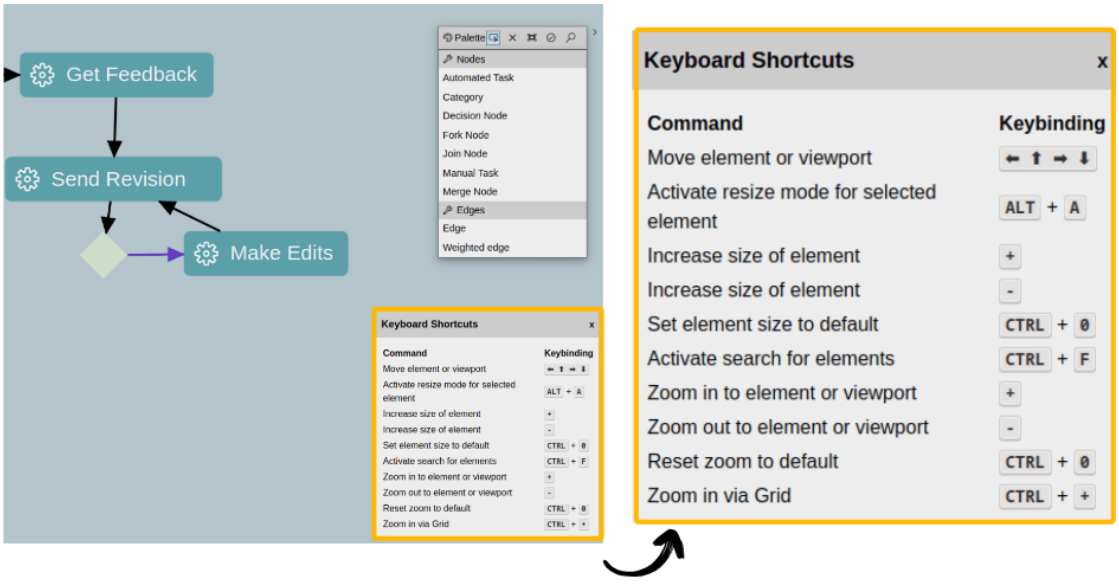

Figure 5.12: Keyboard Shortcut Documentation

User Notifications: Providing visual feedback to users is essential, as it ensures that their actions are triggered or executed correctly and also acts as guidance through the whole modeling process. The assessed web modeling tools in general provide notifications, including tooltips and dialogs. However, for some users, using different notification forms and consistently positioning them in different positions can be confusing. To avoid this issue, minimalistic real-time user feedback is provided, as seen in Figure 5.13. This notification always appears in a fixed position. They display information about, e.g., if

a mode (Resize, Navigation) is activated or deactivated or a hardly visible change has been executed. The provided source code allows other developers to adapt the duration, look and feel, or add additional notifications easily.

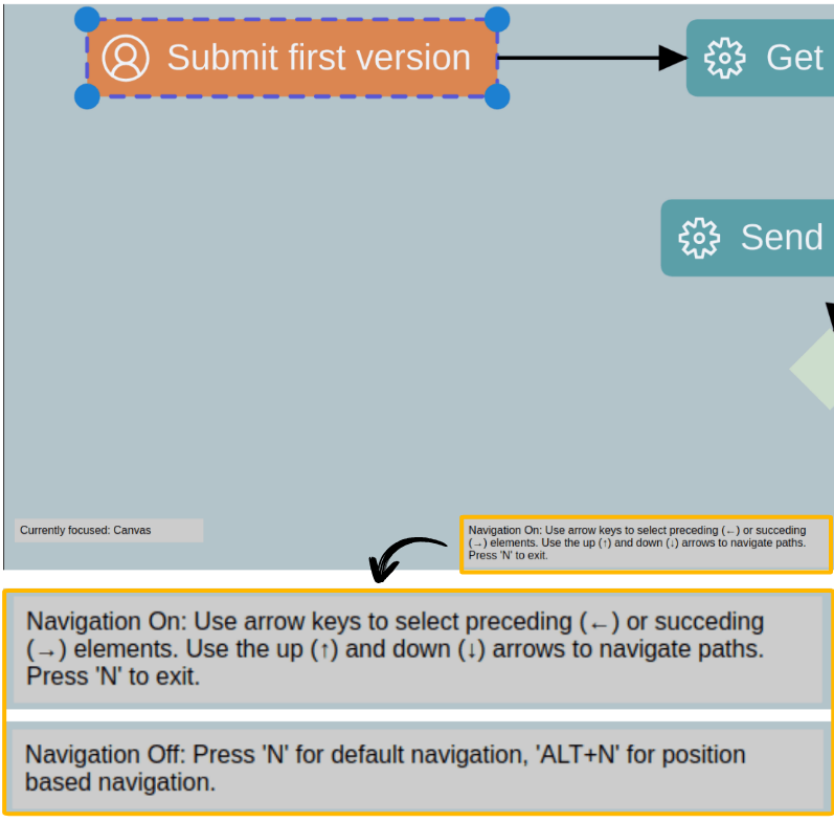

Figure 5.13: Example for User Notification

Visual Indicator for Current Focus: As shown in Figure 5.14, a visual cue indicates the current focus location. Unlike the evaluated modeling tools, which typically use outlined elements to highlight focus, the approach of textual indications offers a distinct advantage. While outlining is helpful in various scenarios, supplying textual information in a fixed canvas spot allows users to confirm the active component directly. This eliminates confusion or the need to search for faint outlines that might be hard to see. The focus is significant as the key shortcuts can result in different actions depending on the focused component.

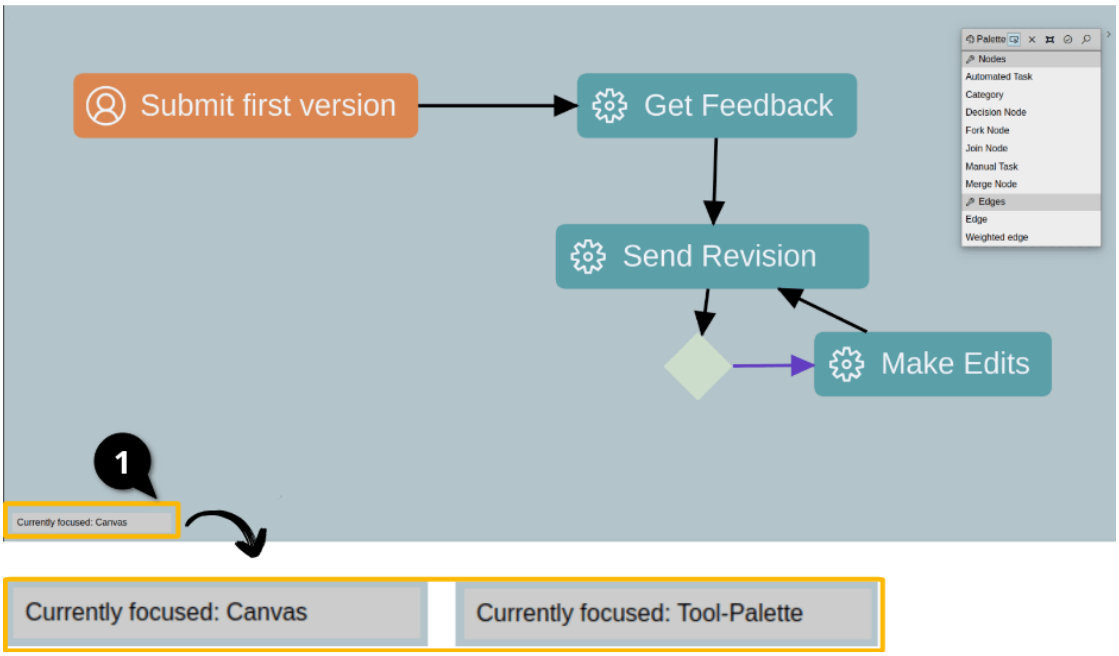

Figure 5.14: Example for Visual Indicator for the Current Focus

5.3.2 Model Exploration

This section introduces diverse techniques for model exploration. The features of Model Exploration facilitate an in-depth examination of element specifics, relationships, and attributes. These tools encompass **zooming**, **movement**, and **interactive modifications**. Users can use these exploration features to enhance their understanding of the model and make well-considered decisions. Typical ways of exploring the model and interacting with the canvas are especially *Movement* (cf. Section 5.3.2), *Zoom* (cf. Sections 5.3.2 and 5.3.2) or modification possibilities, like *Resize* (cf. Section 5.3.2).

Movement

The movement functionality is essential to move parts of the model, such as nodes or edges. This functionality is not a specific feature for physically disabled users, as it is a standard feature for modeling editors; therefore, everyone will benefit from it. The movement allows everyone to participate and allows independence actively. In this case, most modeling editors provide a movement feature. Often there are two alternative ways provided, **(i) move by dragging the element via the mouse** or **(ii) move via arrow keys**. As there is a common way to implement this via *arrow* keys, there is no need for a change or adaption as this is already keyboard-only and suitable for physically disabled users. As moving elements was not possible in the workflow diagram editor, this was implemented, too. The arrow keys make it possible to move selected single or multiple nodes or edges in all directions on the canvas. The elements to be moved can be

selected via the previously mentioned search functionality (cf. Section 5.3.1).

Zoom

The zoom feature is one of the significant model exploration features, as it allows to customize the size of the whole canvas and its content according to the user's needs and improves the visual recognition of the model. As seen in the tool evaluation, almost all modeling editors returned highly satisfactory results for the customizability of text or image sizes (cf. *V1* in Table 4.3.2). Even though Zooming is assigned to visual disabilities in the tool assessment, this functionality is relevant for any user. This is because zooming enables one to examine and work on intricate details closely, making it easier to read, edit, and interact with various elements. Therefore, adding this functionality to the workflow editor was necessary for completeness. Most commonly, zooming can be achieved by using the *respective buttons* on the page, the *mouse wheel*, or the key alternatives *'+'* and *'-'*.

Similar to the **Movement** Functionality (cf. Section 5.3.2), the common key combinations for increasing or decreasing zoom levels are retained. In order to gradually adapt the zoom level of one element, a set of elements, or the canvas itself, the shortcut *'+'* can be used to increase or *'-'* to decrease the zoom level. As for every interaction, it is necessary to provide error correction options (cf. *P7* in Table 4.20). The counteraction for increasing is decreasing, but there should also be a possibility to easily set to default zoom level without repetitively pressing the zooming keys. Therefore, with *'CTRL'+'0'*, the default zoom level and all other zooming activities can be reset.

Zoom via Grid

The Zoom via Grid functionality is an extended version of the previously described **Zoom** Feature (cf. Section 5.3.2). This new action is provided to give the users a more precise zooming experience to the desired direction on the canvas. This means that the Grid will be displayed for zooming only by pressing *CTRL+'+'*. Afterward, the zoom level can be increased by using the respective box number, but now it will not be centered, as the zooming will be carried out in the direction of the desired grid box. This feature gives the user a more refined version of the standard zoom action, as it is advantageous to zoom in on specific parts of the model for further investigation, especially in larger models. This additional functionality aims to allow the modeler more flexibility and independence in the model exploration process.

Resize Element

The resize functionality helps to set the size of the nodes. In most tools, the resizing action is accomplished by dragging the desired edge of the node in another direction. Dragging is a challenging workflow for physically disabled users as precise handling is necessary (cf. *P6* in Table 4.3.4), as explained extensively in the node creation process (cf. Section 5.3.1). In most cases, no keyboard alternative is provided for this kind

of mouse-related modification. Therefore, this prototype has overcome this issue by introducing increasing or decreasing model elements using *'+'* or *'-'* keys. In order to avoid overlapping with the zoom functionality, resizing is only possible after activating the resize mode with the designated shortcut combination *'ALT'+'A'*, or also deactivated with the same shortcut. Again, to provide easy error correction options (cf. *P7* in Table 4.3.4), the key combination *CTRL+0* can be used to reset the default size of the node.

5.3.3 Model Navigation

The last section describes the functionalities used for model navigation. The navigation through a model is essential as it enhances usability, improves productivity, and facilitates efficient exploration and manipulation of complex models. Significantly while the user modifies an existing model, the person likely spends more time reading and understanding. Navigation possibilities heavily support this process [SG15]. In addition, the tool evaluation has also presented that for most of the tools, the provided navigation is not sufficient, if they even have one (cf. *P4* & *P5* in Table 4.3.4).

This prototype implements two different algorithms for navigating through the model, one is the **Default Navigation** (cf. Section 5.3.3), and the second one is the **Position-based Navigation** (cf. Section 5.3.3). Both navigational features have their strengths or weaknesses depending on the model, the context, and the current task the user is working on. Providing two different navigational means focused on different techniques covers the standard ways of iterating through models.

Default Navigation

The **Default Navigation** mode can be activated via the single key shortcut *'N'* after a node has been selected as the starting point. The main idea of this navigation algorithm is to enable navigating through the nodes and edges of the model using the *arrow* keys, depending on the direction of the given relations, i.e., the next element is chosen based on the outgoing or incoming relation depending on which way the user is iterating. This navigation can be used to follow the natural order of the diagram based on the created relations. An example can be seen in Figure 5.15. In this case, if the starting point is the element **'Submit first version'** (marked as **1**), then the right arrow would lead to the next element based on the existing relation, which is the **Edge** (marked as **2**) between both nodes and repeating the right key press would lead to the target node which in that case is **'Get Feedback'** (marked as **3**) and the arrow keys *left, up, down* do not lead to any changes as there are no relations in those directions.

Position-based Navigation

The **Position-based Navigation** mode can be activated via the matching shortcut *'ALT'+'N'*. This second option enables one to navigate based on the x and y coordinates of the elements on the canvas, disregarding their relationships, i.e., the next element to

be selected will be the nearest depending on the coordinates and the selected direction. An example can be seen in Figure 5.16. In this case, the starting point is the element **'Submit first version'** (marked as **1**), then the right arrow would be the right nearest element **'Get Feedback'** (marked as **2**) and down arrow from the same starting point would be the nearest element in the corresponding direction which is the element **'Send Revision'** (marked as **3**).

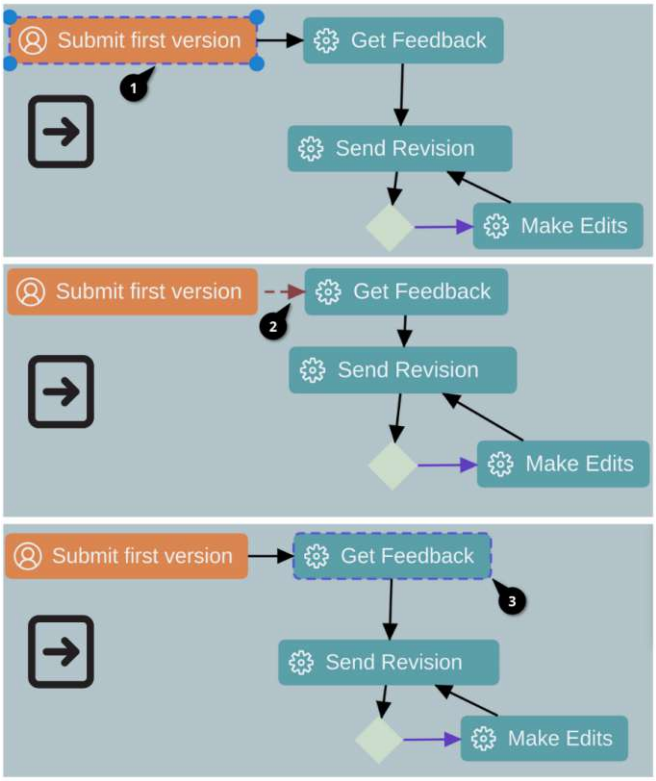

Figure 5.15: Example for Default Navigation

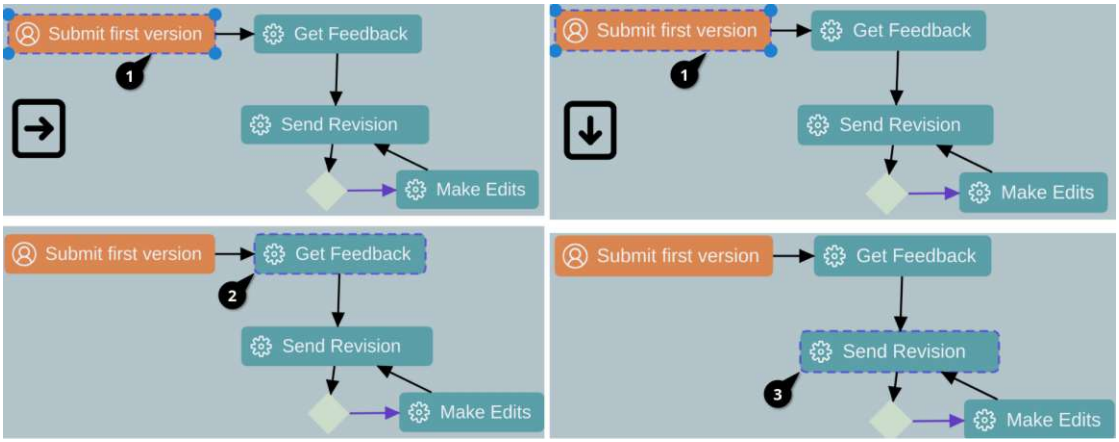

Figure 5.16: Example for Position-based Navigation

5.4 Evaluation

This section will provide an evaluation of the implemented accessibility features for physically disabled individuals in using web modeling editors. The evaluation consists of two parts: Firstly, the new functionalities will be analyzed against the *Keyboard Accessibility Guideline* (cf. Section 5.2.2). Afterward, the section will provide a brief explanation about if and how the features satisfy the *Best Practices for defining keyboard shortcuts* (cf. Section 5.2.3) and the *Keyboard-Only Requirements* (cf. Section 5.2.4). Secondly and lastly, the features will be evaluated against the physical disability support results from the tool assessment (cf. Section 4.3.4).

5.4.1 Evaluation against Guidelines & Requirements

Keyboard Accessibility Guideline (cf. Section 5.2.2): The set of implemented features overall satisfies the success criteria of the *Keyboard Accessibility Guideline* except for some minor flaws regarding criterion **Character Key Shortcuts**. The satisfaction of the remaining criteria is given, as every provided feature is fully operable only via the keyboard for the essential workflow of a modeling editor, without exception, and there are no keyboard traps as the user can easily use the assigned shortcut combinations and move on with the next task or navigate using common means like *arrows* or *tab keys*. However, there are some issues and vital information to note about the previously mentioned success criterion **Character Key Shortcuts**:

- **Issue with Single-Key Shortcuts:** Based on the provided definitions by the Web Content Accessibility Guideline (WCAG) 2.1, using single character keys is efficient for many keyboard users. However, it can be a challenge for speech control users, e.g., overlapping with navigational commands that are often single keys. Nevertheless, providing a universal solution with this kind of prototype is not possible. Therefore, the decision was to use those single-character and numeric keys but provide a generalized version of the implementation where the shortcuts can be replaced and adapted as intended and needed.

- **Using Character Keys:** Furthermore, when using character key shortcuts Web Content Accessibility Guideline (WCAG) 2.1 recommends fulfilling at least one of the three conditions: *(i) provide a mechanism to turn off the shortcut*, *(ii) allow remapping of the keys*, and *(iii) activate shortcuts only when the corresponding components have focus*. In this case, most shortcuts are activatable if the user needs them and, therefore, can be disabled (e.g., Resize, Navigation). For those key shortcuts which are automatically displayed if a specific action is triggered, there also exists a combination for turning on or off (e.g., Grid, Tool Palette), which fulfills **(i)**. Furthermore, the shortcuts can only be activated if the component has focus and can also be disabled via *Escape* or the *Activation Shortcut* if there is no use to it, fulfilling **(iii)**. Currently, the remapping functionality in **(ii)** is given on a

programmatic level but can also be provided to the editor users if further extended to the desired modeling editors.

Best Practice: Defining Keyboard Shortcuts (cf. Section 5.2.3): The six best practice criteria for defining suitable keyboard shortcuts as explained in Section 5.2.3 were fulfilled for the provided keyboard-only prototype. The most critical tasks typical for modeling editors were prioritized and assigned to easy-to-understand keyboard shortcuts (cf. 1, 3, 5) by following existing common standards for shortcuts to avoid any double mapping of keys (cf. 2). Additionally, as far as possible the usage complex, multiple shortcuts have been avoided which in further improves the overall handling via the keyboard, as mostly single key shortcuts were included (cf. 3, 4). To support users in their modeling task and in the interaction with the overall editor itself, textual user feedback, highlighting elements according to their context (e.g., Navigation, Search), and a listing of all of the existing shortcuts is always ready to be used (cf. 6).

Keyboard-Only Requirements: The keyboard-only prototype does fulfill the most common requirements of such a keyboard interface as described in Section 5.2.4. Throughout the design and implementation of the accessibility functionalities, it was ensured that each shortcut combination is only accessible if the corresponding element is **focused** (cf. 1) to avoid any misleadings or accidentally triggering unwanted actions. As presented in Section 5.3.3, two different **navigational** (cf. 2) algorithms to iterate through the model are also provided. As described in the evaluation against the best practices of shortcut definitions in this Section, the shortcuts are designed intuitively, using easy-to-follow key combinations with at most two keys to avoid complexity. Furthermore, existing common standards and guidelines are followed, ensuring **consistency** by avoiding overlapping with system defaults. Additionally, **user notifications** are provided in real-time to inform the modeler about the triggered actions, enabled or disabled interaction modes, and similar. Lastly, visual aids are added to highlight invalid actions, such as the creation of overlapping nodes with the pointer (cf. Section 5.3.1) or for filtering and highlighting searched elements, such as marking non-matching elements of a search task (cf. Section 5.3.1), or highlighting the current element focus while navigating through a given model.

5.4.2 Evaluation against the Results of Physical Disability Support

This section will compare how the newly introduced set of accessibility features perform compared to the average results per evaluation criteria from the tool evaluation regarding the physical disability support (cf. Table 5.1).

In Table 5.1, the first two columns labeled **Average Result** represent the results from the assessment per criteria on average. The remaining two columns labeled **New Features** present the results of the new accessibility features according to each criterion, followed by a brief explanation for each criterion.

Table 5.1: Evaluation result for the categories Tool Support & GUI (TG) and Canvas & Model (CM) for Physical (P) Disabilities **Before and After the new accessibility features**.

Tool	Average Result		New Features	
	TG	CM	TG	CM
P1	○	○	●	●
P2	◐	●	●	●
P3	●	○	-	-
P4	◐	◐	●	●
P5	◐	◐	●	●
P6	●	◐	●	●
P7	●	●	●	●

The evaluation criteria *P1 - Full Keyboard Support* was not fulfilled by any of the modeling tools. This criterion is entirely fulfilled by providing this keyboard-only prototype with new accessibility features.

Furthermore, for *P2 - Sufficient time limits to react*, the challenge was the *Drag & Drop* functionality, which was criticized due to the need for precise and swift handling (detailed explanation in Section 5.3.1). Providing keyboard shortcuts for this task, e.g., creating nodes or relations, also positively influenced this criterion, as it counteracts the earlier criticism.

The assessment criterion *P3 - Controls, images and similar with text alternatives* was not compared in this case, as the prototype was not intended to improve the text alternatives of elements and therefore does not include any additional mechanism.

Observing the criterion *P4 - Visual and non-visual orientation or navigation cues*, it is also evident that the average results were not satisfactory in fulfilling the criterion due to the missing *Keyboard Navigation* possibility. On the one hand, the prototype has implemented a textual indicator (cf. Section 5.3.1) displaying the currently focused component on the canvas, and on the other hand, a keyboard navigation possibility is added to navigate through the tool itself (*TG*) and two navigation mechanism for the model itself (*CM*) (cf. Section 5.3.3).

In the case of *P5 - Logical navigational mechanism and page functions*, again, the one issue hindering the tools from entirely fulfilling the criterion was the missing *Keyboard Navigation*. As this is now explicitly added for *Canvas & Model (CM)* and the Workflow Diagram Editor is already capable of navigating for *Tool Support & GUI (TG)*, the criterion can be marked as fulfilled.

From the point of view of the *Canvas & Model (CM)*, the criterion *P6 - Large clickable areas* was a challenge due to small components or features which rely on precise handling with the mouse or other pointing devices. Providing a keyboard-only alternative mechanism for the key tasks of a modeling editor influenced the outcomes for this criterion

positively, as there is no need for precise handling anymore due to the modeling tasks being achievable via key shortcuts.

The last criterion, which is *P7 - Error Correction Options*, has already provided the best results. Therefore, throughout the implementation of the new features, the goal was to keep this up by providing error correction options if applicable. Furthermore, the prototype is even enriched with proactive ways in order to avoid invalid or accidentally triggering interactions by providing mechanisms that guide the modeler and prevent invalid actions (cf. Pointer in Section 5.3.1, Smart Search in Section 5.3.1).

5.5 Conclusion

This section provides an insight into the design and implementation process of the keyboard-only prototype, intending to provide the first building block of including accessibility features to existing web modeling tools without replacing existing features or the need for any external tool. The idea is to show that it is possible to enhance accessibility by adding innovative components as support or extending existing common interaction possibilities. The implementation extends the workflow diagram editor by Eclipse GLSP. However, it is also provided as an easy plug-in solution to enable and engage other tool developers to make their tool accessible for physically impaired modelers.

Of course, this prototype is only a tiny portion of the possibility of achieving accessible web modeling tools, and it is impossible to support every single person with unique physical disabilities through this set of interactions. However, this lays the foundation for future research and technological advancements, e.g., by engaging to conduct empirical tests on this prototype in the future and gaining more knowledge to improve the accessibility state further.

Solving these challenges effectively needs teamwork. This means bringing together experts in different fields like computer science (conceptual modeling, software engineering, human-computer interaction), social science, and maybe even medicine. This thesis can influence and engage different fields by showing the gaps and some functional possibilities as a starting point.

CHAPTER 6

Conclusion

6.1 Summary

The goal of this thesis was to investigate and present the state of web accessibility in conceptual modeling research and currently well-established web modeling tools and to combine those findings to provide a valuable contribution with a keyboard-only prototype toward a more inclusive modeling experience in nowadays web modeling tools.

In order to achieve this, the thesis has been divided into three main stages that build upon each other. (i) The initial stage focused on acquiring a comprehensive understanding of research in conceptual modeling and web accessibility, (ii) the second stage was based on investigating the accessibility state and support offered by existing web modeling tools, and (iii) the last stage is the prototype part of this thesis, for combining the results of the investigations and gained knowledge and providing the first building blocks towards a more inclusive modeling experience in web-based modeling editors specifically for physically disabled users.

The comprehensive systematic literature review is the first stage and provides the foundation for the overall work. The literature research has led to the discovery of valuable papers and a better understanding of web accessibility research. Additionally, it has revealed a significant research gap concerning accessibility in conceptual modeling. The systematic literature review shows that web accessibility research has increased significantly. However, conceptual modeling research is still scarce and only focused on visual disabilities and theoretical contributions. The relevant papers cover a wide range of topics, highlighting the need for considering accessibility early in the software development process, evaluating accessibility from a user perspective, exploring automation possibilities, and addressing the gaps in research and solutions for various disabilities in conceptual modeling. However, still, there are apparent gaps in conceptual modeling research, especially for *Physical*, *Auditory*, *Speech*, and *Learning* disabilities. Significantly, this

field is a rapidly growing and essential research area where new contributions would be beneficial and valuable in the community of disabled modelers.

The identified research gap leads to further investigation, provided in this thesis's second part. The accessibility state and disability support offered by existing web modeling tools have been assessed for *Visual*, *Physical*, and *Learning* disabilities. While most of the assessed tools resulted in shortcomings in terms of customization options to accommodate user needs or to provide satisfactory text-to-speech synthesis, there was one criterion that none of the assessed modeling tools fulfilled: full keyboard support. This issue is particularly significant for individuals with *Physical* disabilities who heavily depend on keyboard interaction. In summary, one criterion, namely full keyboard support, was not met satisfactorily in the evaluation process.

The research gap from the first stage and the particular findings of the tool assessment revealed insufficient full-keyboard support for physical disabilities. Further, it showed limited support regarding the evaluation criteria for *Visual* and *Learning* disabilities in the assessed modeling tools. Both stages' findings served as a motivation for the prototype phase of this work, which is the third stage of this thesis. The findings of the first two stages resulted in the development of a keyboard-only prototype extension for the workflow diagram editor by Eclipse GLSP. This prototype was developed, specifically catering to keyboard-only interactions, aiming to address the lack of keyboard support identified during the evaluation, providing a solution for physically disabled individuals and allowing modeling interactions only with the keyboard for a more inclusive modeling environment. The goal is to simplify the usage of heavily mouse-based web modeling tools and engage physically disabled users in modeling and using web modeling tools. Of course, it is worth mentioning that not every interaction that relies on a mouse or other pointing device can be entirely replaced by a keyboard shortcut in the same way. However, this prototype has shown that the fundamental interactions needed to create, manipulate, explore, or navigate a model can be keyboard-only by adding a new mechanism or structure to the tool.

The keyboard-only prototype is not only successfully extending the workflow diagram editor by GLSP. However, it is also successfully generalized and integrated into the open-source project, ready to be used or integrated into other platforms (e.g., Eclipse Theia, VSCode) to include a fundamental set of accessibility features in various platforms and to engage the extensions and improvement of accessibility features.

6.2 Future Work

This thesis has provided a fully functional keyboard-only prototype improving the inclusivity of physically disabled users in modeling editors. Additionally, the accessibility features of the prototype are already included in the open-source project of the Eclipse GLSP Client, which is ready-to-use, integrateable, and extendable into further platforms.

Nevertheless, this contribution is only the first step into a more inclusive modeling

environment, and further research and implementation of advanced technologies will be beneficial to enhance the accessibility state of web modeling editors. This thesis has provided valuable findings, insights, and concrete design and structure ideas through the prototype, which can further influence future research and technological improvements in this area.

Some future work would be conducting empirical studies with a group of disabled users in order to analyze how the prototype and its features are improving the modeling process or to explore which further advancements and improvements could be helpful.

Furthermore, a more advanced tool evaluation could be conducted in the future, including a wide variety of web tools that focus on other diagram types and compare a larger set of tools.

Furthermore, the findings of this thesis are planned to share with more scientific communities. The main findings of this work have been accepted at the International Conference on Conceptual Modeling (ER 2023) [SMB23]. Including this topic, the findings, and the keyboard-only prototype in these scientific conferences help reach the target users and will be provided to a broader audience, influencing further works and improvements in accessibility research regarding conceptual modeling.

CHAPTER 7

Appendix

Appendix 1: Selected Publications

Appendix 1: Selected publications

#	Title	Authors	Reference	Year	Source
1	Presenting UML Software Engineering Diagrams to Blind People	A. King and P. Blenkhorn and D. Crombie and Sijo Dijkstra and G. Evans and John Wood	King, A., Blenkhorn, P., Crombie, D., Dijkstra, S., Evans, G., & Wood, J. (2004, July). Presenting UML software engineering diagrams to blind people. In International Conference on Computers for Handicapped Persons (pp. 522-529). Springer, Berlin, Heidelberg.	2004	ConnectedPapers
2	Comparing Accessibility Evaluation and Usability Evaluation in Hag\'{a}Qu\'{e}	Tanaka, Eduardo Hideki and Bim, Silvia Amelia and da Rocha, Heloisa Vieira	Tanaka, E. H., Bim, S. A., & da Rocha, H. V. (2005, October). Comparing accessibility evaluation and usability evaluation in HagáQuê. In Proceedings of the 2005 Latin American conference on Human-computer interaction (pp. 139-147).	2005	SearchQuery
3	Adding Semantics in Web-Based Digital Libraries to Support Information Seeking of Blind People	Salampasis, Michail and Kouroupetroglou, Chr. and Tektonidis, Dimitris	Salampasis, M., Kouroupetroglou, C., & Tektonidis, D. (2005, August). Adding semantics in web-based digital libraries to support information seeking of blind people. In Proceedings of the 5th WSEAS international conference on Simulation, modelling and optimization (pp. 164-171).	2005	SearchQuery
4	Web Compliance Management: Barrier-Free Websites Just by Simply Pressing the Button? Accessibility and the Use of Content-Management-Systems	Schulz, Martina and Pieper, Michael	Schulz, M., & Pieper, M. (2007). Web compliance management: barrier-free websites just by simply pressing the button? accessibility and the use of content-management-systems. In Universal Access in Ambient Intelligence Environments (pp. 419-426). Springer, Berlin, Heidelberg.	2006	SearchQuery
5	New environment for visually disabled students to access scientific information by combining speech interface and tactile graphics	Komada, Toshihiko and Yamaguchi, Katsuhito and Kawane, Fukashi and Suzuki, Masakazu	Komada, T., Yamaguchi, K., Kawane, F., & Suzuki, M. (2006, July). New environment for visually disabled students to access scientific information by combining speech interface and tactile graphics. In International Conference on Computers for Handicapped Persons (pp. 1183-1190). Springer, Berlin, Heidelberg.	2006	SearchQuery
6	Editing a Test Suite for Accessibility of Interactive Web Sites	Weber, Gerhard and Weimann, Kurt	Weber, G., & Weimann, K. (2007, July). Editing a test suite for accessibility of interactive web sites. In International Conference on Universal Access in Human-Computer Interaction (pp. 193-201). Springer, Berlin, Heidelberg.	2007	SearchQuery
7	Easing Web Guidelines Specification	Leporini, Barbara and Patern\`{o}, Fabio and Scorcia, Antonio	Leporini, B., Paternò, F., & Scorcia, A. (2007, July). Easing web guidelines specification. In International Conference on Web Engineering (pp. 254-268). Springer, Berlin, Heidelberg.	2007	SearchQuery
8	Constructing relational diagrams in audio: the multiple perspective hierarchical approach	Oussama Metatla and N. Bryan-Kinns and T. Stockman	Metatla, O., Bryan-Kinns, N., & Stockman, T. (2008, October). Constructing relational diagrams in audio: the multiple perspective hierarchical approach. In Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility (pp. 97-104).	2008	ConnectedPapers
9	Modelling perception using image processing algorithms	Biswas, Pradipta and Robinson, Peter	Biswas, P., & Robinson, P. P. (2009). Modelling perception using image processing algorithms.	2009	SearchQuery
10	What's next? A visual editor for correcting reading order	Sato, Daisuke and Kobayashi, Masatomo and Takagi, Hironobu and Asakawa, Chieko	Sato, D., Kobayashi, M., Takagi, H., & Asakawa, C. (2009, August). What's next? a visual editor for correcting reading order. In IFIP Conference on Human-Computer Interaction (pp. 364-377). Springer, Berlin, Heidelberg.	2009	SearchQuery
11	An integrative accessibility engineering approach using multidimensional classifications of barriers in the web	Ruth-Janneck, Diana	Ruth-Janneck, D. (2011, March). An integrative accessibility engineering approach using multidimensional classifications of barriers in the web. In Proceedings of the International Cross-Disciplinary Conference on Web Accessibility (pp. 1-7).	2011	SearchQuery
12	Accessibility Evaluation Improvement Using Case Based Reasoning	Avila, Cecilia and Guevara, Juan Carlos and Fabregat, Ramon and Baldiris, Silvia	Avila, C., Baldiris, S., Fabregat, R., & Guevara, J. C. (2012, October). Accessibility evaluation improvement using case based reasoning. In 2012 Frontiers in Education Conference Proceedings (pp. 1-6). IEEE.	2012	SearchQuery
13	Viable haptic UML for blind people	Loitsch, Claudia and Weber, Gerhard	Loitsch, C., & Weber, G. (2012, July). Viable haptic UML for blind people. In International Conference on Computers for Handicapped Persons (pp. 509-516). Springer, Berlin, Heidelberg.	2012	SearchQuery

Appendix 1: Selected Publications

14	A Web Content Accessibility Evaluation Process for Learning Objects in the Context of a Virtual Learning Environment	Avila, Cecilia and Baldiris, Silvia and Fabregat, Ramon and Guevara, Juan Carlos	Avila, C., Baldiris, S., Fabregat, R., & Guevara, J. C. (2012, September). A web content accessibility evaluation process for learning objects in the context of a virtual learning environment. In International Conference on Web-Based Learning (pp. 181-190). Springer, Berlin, Heidelberg.	2014	SearchQuery
15	Collaborative editing: Collaboration, awareness and accessibility issues for the blind	Buzzi, Maria Claudia and Buzzi, Marina and Leporini, Barbara and Mori, Giulio and Penichet, Victor M.R	Buzzi, M. C., Buzzi, M., Leporini, B., Mori, G., & Penichet, V. M. (2014, October). Collaborative editing: collaboration, awareness and accessibility issues for the blind. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 567-573). Springer, Berlin, Heidelberg.	2014	SearchQuery
16	Can we work together? on the inclusion of blind people in UML model-based tasks	Luque, L. and Veriscimo, E.S. and Pereira, G.C. and Filgueiras, L.V.L.	Luque, L., Veriscimo, E. D. S., Pereira, G. D. C., & Filgueiras, L. V. L. (2014). Can we work together? on the inclusion of blind people in uml model-based tasks. In Inclusive Designing (pp. 223-233). Springer, Cham.	2014	SearchQuery
17	Empathic Communication of Accessibility Barriers in Web 2.0 Editing	Pascual, Afra and Ribera, Mireia and Granollers, Toni	Pascual, A., Ribera, M., & Granollers, T. (2015, May). Empathic communication of accessibility barriers in web 2.0 editing. In Proceedings of the 12th International Web for All Conference (pp. 1-8).	2015	SearchQuery
18	Towards collaboration on accessible UML models	Seifermann, Stephan and Groenda, Henning	Seifermann, S., & Groenda, H. (2015). Towards Collaboration on Accessible UML Models. Mensch und Computer 2015–Workshopband.	2015	SearchQuery
19	Survey on textual notations for the Unified Modeling Language	Stephan Seifermann and Henning Groenda	Seifermann, S., & Groenda, H. (2016, February). Survey on textual notations for the unified modeling language. In 2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD) (pp. 28-39). IEEE.	2016	ConnectedPapers
20	Architectural metamodel for requirements of images accessibility in online editors	Villarroel-Ramos, Jorge and Sanchez-Gordon, Sandra and Lujan-Mora, Sergio	Villarroel-Ramos, J., Sanchez-Gordon, S., & Luján-Mora, S. (2018, November). Architectural metamodel for requirements of images accessibility in online editors. In 2018 International Conference on Information Systems and Computer Science (INCISCOS) (pp. 312-319). IEEE.	2018	SearchQuery
21	Method for accessibility assessment of online content editors	Acosta, Tania and Acosta-Vargas, Patricia and Salvador-Ullauri, Luis and Luján-Mora, Sergio	Acosta, T., Acosta-Vargas, P., Salvador-Ullauri, L., & Luján-Mora, S. (2018, January). Method for accessibility assessment of online content editors. In International Conference on Information Technology & Systems (pp. 538-551). Springer, Cham.	2018	SearchQuery
22	Guest Editor's Introduction: Reimagining Disability and Accessibility in Technical and Professional Communication	Zdenek, Sean	Zdenek, S. (2019). Guest editor's introduction: Reimagining disability and accessibility in technical and professional communication. Communication Design Quarterly Review, 6(4), 4-11.	2019	SearchQuery
23	Overcoming Accessibility Barriers for People with Severe Vision Impairment in Web-based Learning Environments: A Literature Review	M. Nascimento and A. Brandão and L. O. Brandão and Francisco Oliveira	do Nascimento, M. D., Brandão, A. A., de Oliveira Brandão, L., & de MB Oliveira, F. C. (2019, October). Overcoming Accessibility Barriers for People with Severe Vision Impairment in Web-based Learning Environments: A Literature Review. In 2019 IEEE Frontiers in Education Conference (FIE) (pp. 1-8). IEEE.	2019	SearchQuery
24	Approaches for diagrams accessibility for blind people: a systematic review	M. Torres and Regina Barwaldt	Torres, M. J. R., & Barwaldt, R. (2019, October). Approaches for diagrams accessibility for blind people: a systematic review. In 2019 IEEE Frontiers in Education Conference (FIE) (pp. 1-7). IEEE.	2019	ConnectedPapers
25	An Auditory Interface to Workspace Awareness Elements Accessible for the Blind in Diagrams' Collaborative Modeling	M. Torres and Regina Barwaldt and Paulo Cesar Ramos Pinho and L. Topin and T. F. Otero	Torres, M. J. R., Barwaldt, R., Pinho, P. C. R., de Topin, L. O. H., & Otero, T. F. (2020, October). An auditory interface to workspace awareness elements accessible for the blind in diagrams' collaborative modeling. In 2020 IEEE Frontiers in Education Conference (FIE) (pp. 1-7). IEEE.	2020	ConnectedPapers

Appendix 1: Selected Publications

26	Transforming Diagrams' Semantics to Text for Visually Impaired	Charles D. Cross and D. Cetinkaya and H. Dogan	Cross, C., Cetinkaya, D., & Dogan, H. (2020, July). Transforming Diagrams' Semantics to Text for Visually Impaired. In International Conference on Human-Computer Interaction (pp. 339-350). Springer, Cham.	2020	ConnectedPapers
27	An Accessible Evaluation Tool to Detect Easy-To-Read Barriers	Morato, Jorge and Campillo, Adrian and Sanchez-Cuadrado, Sonia and Iglesias, Ana and Berrios, Olga	Morato, J., Campillo, A., Sanchez-Cuadrado, S., Iglesias, A., & Berrios, O. (2020, December). An Accessible Evaluation Tool to Detect Easy-To-Read Barriers. In 9th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion (pp. 55-60).	2020	SearchQuery
28	Self-directed creation and editing of uml class diagrams on mobile devices for visually impaired people	Wildhaber, Fabian and Salloum, Nadim and Gygli, Marcel and Kennel, Andrea	Wildhaber, F., Salloum, N., Gygli, M., & Kennel, A. (2020, August). Self-Directed Creation and Editing of UML Class Diagrams on Mobile Devices for Visually Impaired People. In 2020 IEEE Tenth International Model-Driven Requirements Engineering (MoDRE) (pp. 49-57). IEEE.	2020	SearchQuery
29	The diversity crisis in software development	Albusays, Khaled and Bjorn, Pernille and Dabbish, Laura and Ford, Denae and Murphy-Hill, Emerson and Serebrenik, Alexander and Storey, Margaret-Anne	Albusays, K., Bjorn, P., Dabbish, L., Ford, D., Murphy-Hill, E., Serebrenik, A., & Storey, M. A. (2021). The diversity crisis in software development. IEEE Software, 38(2), 19-25.	2021	SearchQuery
30	An Editing Process for Blind or Visually Impaired Editors	Baker, Matthew and Nightingale, E.M. and Bills, Suzy	Baker, M. J., Nightingale, E. M., & Bills, S. (2021). An Editing Process for Blind or Visually Impaired Editors. IEEE Transactions on Professional Communication, 64(3), 275-287.	2021	SearchQuery
31	Towards a Modelling Workbench with flexible Interaction Models for Model Editors operating through Voice and Gestures	de Carvalho, João Fonseca and Amaral, Vasco	de Carvalho, J. F., & Amaral, V. (2021, July). Towards a Modelling Workbench with flexible Interaction Models for Model Editors operating through Voice and Gestures. In 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC) (pp. 1026-1031). IEEE.	2021	SearchQuery
32	Gosling: A Grammar-Based Toolkit for Scalable and Interactive Genomics Data Visualization	L'Yi, Sehi and Wang, Qianwen and Lekschas, Fritz and Gehlenborg, Nils	L'Yi, S., Wang, Q., Lekschas, F., & Gehlenborg, N. (2021). Gosling: A grammar-based toolkit for scalable and interactive genomics data visualization. IEEE Transactions on Visualization and Computer Graphics, 28(1), 140-150.	2022	SearchQuery
33	Enabling Convenient Online Collaborative Writing for Low Vision Screen Magnifier Users	Lee, Hae-Na and Prakash, Yash and Sunkara, Mohan and Ramakrishnan, I.V. and Ashok, Vikas	Lee, H. N., Prakash, Y., Sunkara, M., Ramakrishnan, I. V., & Ashok, V. (2022, June). Enabling Convenient Online Collaborative Writing for Low Vision Screen Magnifier Users. In Proceedings of the 33rd ACM Conference on Hypertext and Social Media (pp. 143-153).	2022	SearchQuery
34	Improving accessibility of CMS-based websites using automated methods	Csontos, Balázs and Heckl, István	Csontos, B., & Heckl, I. (2020). Improving accessibility of CMS-based websites using automated methods. Universal Access in the Information Society, 1-15.	2022	SearchQuery
35	Inclusive Multimodal Voice Interaction for Code Navigation	Paudyal B., Creed C., Williams I., Frutos-Pascual M.	Paudyal, B., Creed, C., Williams, I., & Frutos-Pascual, M. (2022, November). Inclusive Multimodal Voice Interaction for Code Navigation. In Proceedings of the 2022 International Conference on Multimodal Interaction (pp. 509-519).	2022	SearchQuery
36	Grid-Coding: An Accessible, Efficient, and Structured Coding Paradigm for Blind and Low-Vision Programmers	Ehtesham-Ul-Haque M., Monsur S.M., Billah S.M.	Ehtesham-Ul-Haque, M., Monsur, S. M., & Billah, S. M. (2022, October). Grid-Coding: An Accessible, Efficient, and Structured Coding Paradigm for Blind and Low-Vision Programmers. In Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology (pp. 1-21).	2022	SearchQuery
37	Source code editor using voice commands to support people with motor disabilities	Muñoz J.G.S., Bringas J.A.S., Encinas I.D., León M.A.C., Verdugo A.I.D.C.	Muñoz, J. G. S., Bringas, J. A. S., Encinas, I. D., León, M. A. C., & Verdugo, A. I. D. C. (2023). Source code editor using voice commands to support people with motor disabilities. Universal Access in the Information Society, 1-18.	2023	SearchQuery

Appendix 2: Commercial Software and Tools

Appendix 2: Commercial Software and Tools				
#	**Title**	**Company**	**Plans and Versions**	**Reference (Accessed: 18.08.2023)**
1	Lucidchart	Lucid Software	Free Plan	https://www.lucidchart.com/
2	GenMyModel	Axellience	Free Online GenMyModel Environment	https://www.genmymodel.com/
3	Gliffy	Perforce	Free Online Trial	https://www.gliffy.com/
4	diagrams.net	JGraph Ltd	Free Online Version	https://app.diagrams.net/
5	Creately	Cinergix Pty. Ltd.	Free Plan	https://creately.com/
6	Cacoo	Nulab Inc.	Free Plan	https://nulab.com/cacoo/
7	UMLetino	Open-Source Community	Free - no other plans exists	https://www.umletino.com/umletino.html
8	Diagramo	Diagramo Ltd.	Free - no other plans exists	http://diagramo.com/
9	miro	RealtimeBoard Inc.	Free Plan	https://miro.com/
10	BPMN.io.	Camunda Services GmbH	Free Online Version	https://bpmn.io/

Appendix 3: Additional support tools and extensions

Appendix 3: Additional support tools and extensions				
#	Title	Type	Area of operation	Reference (Accessed: 18.08.2023)
1	Magnifying Glass (Hover Zoom)	Google Chrome Extension	Magniyfing glass for zooming	https://mybrowseraddon.com/magnifying-glass.html
2	OneLine	Google Chrome Extension	Reading aid for highlighting rows of web pages	https://useoneline.com/
3	Read Aloud	Google Chrome Extension	Text-to-Speech Voice Reader	https://readaloud.app/
4	WAVE by WebAIM (Web Accessibility Evaluation Tool)	Browser Extension / Standalone API	Creates report for common accessibility errors	https://wave.webaim.org/
5	Accessibility Insights for Web	Browser and Desktop Extension	Checking for Accessibility Compliance	https://accessibilityinsights.io/docs/web/overview/

List of Figures

2.1 The steps of the systematic literature review applied in this thesis 8
2.2 Document collection and major filtering steps 15
2.3 Retrieved documents categorized by publication year 16
2.4 Initial categorization of the 94 papers after abstract review (cf. Table 2.1) 16
2.5 Trend of the 37 selected publications 17
2.6 Selected publications categorized as defined in Table 2.1 18
2.7 Publications grouped by countries of the contributed authors 19
2.8 Number of publications based on publication source and subject area. . . 20
2.9 Overview of the disability types combined with the contribution type. . . 20

4.1 Example Snippet of Lucidchart for *Tool Support & GUI* elements 52
4.2 Snippet from Lucidchart: Template Gallery 53
4.3 Example Snippet of Lucidchart: Dialog and Notification 54
4.4 Example of default workflow UML diagram in Lucidchart 55
4.5 Example Snippet of Lucidchart for *Canvas & Model* elements 56
4.6 Example Snippet of Lucidchart: Canvas Interactions 57
4.7 Example Snippet of Lucidchart: Canvas Element Adaptations with Rotation Arrow and Resizing Markers 57
4.8 Example Snippet of Lucidchart: Tool-Tips 58
4.9 Example Snippet of Miro: Zooming from 100% to 150% only resizes the model, not the remaining elements. 73
4.10 Example Snippet of diagrams.net: Changing appearance of the overall tool 74
4.11 Example Snippet of GenMyModel: Text label with visual supplements . . 76
4.12 Example Snippet of Miro (Left) and Diagramo (Right): Insufficient Labeling 77
4.13 Example Snippet of UMLetino: Labelling from the perspective of *Canvas & Model* 77
4.14 Example Snippet of GenMyModel: Insufficient Element Sizes 81
4.15 Example Snippet of Creately: Menu Panel 82

5.1 Overview of the provided accessibility features with the prototype 98
5.2 Workflow Diagram Editor: Tool Palette 100
5.3 Accessible Tool Palette: Palette Entries (1) and Header Menu (2) 101
5.4 Create Node: Grid 102
5.5 Example for Eclipse Theia Integration: Accessible Tool Palette & Grid . 102

5.6 Create Node: Pointer (1) 103
5.7 Create Node: Pointer for Valid (1) and Invalid (2) Positions 104
5.8 Search Functionality: Search Field (1) and List of matching elements (2) . 105
5.9 Search Functionality: Non-Matching Elements are marked transparently . 106
5.10 Create Edge: Search & Select **Source** Element (1) and **Target** Element (2) 107
5.11 Create Weighted Edge: Search & Select **Source** Element (1) and **Target** Element (2) 107
5.12 Keyboard Shortcut Documentation 108
5.13 Example for User Notification 109
5.14 Example for Visual Indicator for the Current Focus 110
5.15 Example for Default Navigation 113
5.16 Example for Position-based Navigation 113

List of Tables

2.1 Categories used for the initial mapping 12
2.2 Article Metadata used for classification 12
2.3 Classification of disability types by the Web Accessibility Initiative (WAI) 13
2.4 Classification types for the proposed solutions 14
2.5 Classification for the focus and type of the publications 14

4.1 Evaluated Web Modeling Tools 48
4.2 Fulfillment conditions for evaluation criteria V1 60
4.3 Fulfillment conditions for evaluation criteria V2 60
4.4 Fulfillment conditions for evaluation criteria V3 61
4.5 Fulfillment conditions for evaluation criteria CLN1 62
4.6 Fulfillment conditions for evaluation criteria CLN2 62
4.7 Fulfillment conditions for evaluation criteria CLN3 63
4.8 Fulfillment conditions for evaluation criteria CLN4 64
4.9 Fulfillment conditions for evaluation criteria CLN5 64
4.10 Fulfillment conditions for evaluation criteria P1 65
4.11 Fulfillment conditions for evaluation criteria P2 66
4.12 Fulfillment conditions for evaluation criteria P3 66
4.13 Fulfillment conditions for evaluation criteria P4 67
4.14 Fulfillment conditions for evaluation criteria P5 68
4.15 Fulfillment conditions for evaluation criteria P6 69
4.16 Fulfillment conditions for evaluation criteria P7 69
4.17 Overall evaluation result for the categories Tool Support & GUI (TG) and Canvas & Model (CM) for Visual (V), Cognitive, Learning, Neurological (CLN), and Physical (P) Disabilities. 71
4.18 Evaluation result for the categories Tool Support & GUI (TG) and Canvas & Model (CM) for Visual (V) Disabilities. 72
4.19 Evaluation result for the categories Tool Support & GUI (TG) and Canvas & Model (CM) for Cognitive, Learning and Neurological (CLN) Disabilities. 75
4.20 Evaluation result for the categories Tool Support & GUI (TG) and Canvas & Model (CM) for Physical (P) Disabilities. 79
4.21 Strengths and Weaknesses of Lucidchart compared to other assessed tools 84
4.22 Strengths and Weaknesses of GenMyModel compared to other assessed tools 84
4.23 Strengths and Weaknesses of Gliffy compared to other assessed tools . . . 85

4.24 Strengths and Weaknesses of diagrams.net compared to other assessed tools 85
4.25 Strengths and Weaknesses of Creately compared to other assessed tools . 86
4.26 Strengths and Weaknesses of Cacoo compared to other assessed tools . . . 86
4.27 Strengths and Weaknesses of UMLetino compared to other assessed tools 87
4.28 Strengths and Weaknesses of Diagramo compared to other assessed tools . 87
4.29 Strengths and Weaknesses of Miro compared to other assessed tools . . . 88
4.30 Strengths and Weaknesses of BPMN.io compared to other assessed tools . 88

5.1 Evaluation result for the categories Tool Support & GUI (TG) and Canvas & Model (CM) for Physical (P) Disabilities **Before and After the new accessibility features**. 116

Glossary

Keyboard Accessibility According to Web Accessibility In Mind (WebAIM), Keyboard Accessibility is the design and development of websites and applications so that users can navigate, interact and operate with all provided functionalities only using a keyboard and without relying on a mouse or other pointing device[1].. 92

mnemonic Mnemonic refers to a memory aid or device that helps people remember information more easily. [2].. 95

Mozilla Developer Network (MDN) Mozilla Developer Network (abbrev: MDN) is an online resource for web developers, offering documentation, guides, and tutorials on web technologies like HTML, CSS, and JavaScript [3].. 96

Tacticle Graphics Tactile Graphics deliver mostly non-textual information using raised surfaces for visually impaired persons to touch [4]. . 28, 32, 33

Web Accessibility In this thesis, the term *Accessibility* refers to *Web Accessibility*, which is defined by the Web Accessibility Initiative (WAI) as follows [5]:

> *"Web accessibility means that websites, tools, and technologies are designed and developed so that people with disabilities can use them"*

. 25

Web Accessibility In Mind (WebAIM) Web Accessibility in Mind (abbrev. WebAIM) is a nonprofit organization that educates and supports web creators in making digital content accessible to people with disabilities by offering resources,

[1] Keyboard Accessibility, https://webaim.org/techniques/keyboard/, (Access: 03.08.2023)

[2] Cambridge Dictionary, https://dictionary.cambridge.org/dictionary/english/mnemonic, (Access: 06.08.2023)

[3] MDN, https://developer.mozilla.org/en-US/, (Access: 03.08.2023)

[4] Tacticle Graphics, https://www.washington.edu/doit/what-are-tactile-graphics, (Accessed: 23.07.2023)

[5] Introduction to Web Accessibility, https://www.w3.org/WAI/fundamentals/accessibility-intro/ (Accessed: 28.09.2022)

training, tools, and more. WebAIM aims to promote inclusive design and a more accessible web experience for everyone, regardless of their abilities [6]. . 2, 96, 133

Web Accessibility Initiative (WAI) Web Accessibility Initiative (abbrev. WAI) is led by the World Wide Web Consortium (W3C) to promote accessibility on the web. WAI provides guidelines, resources, and strategies to help make websites and web content accessible to individuals with disabilities, ensuring equal access to information and services online.[7] . 4, 9, 10, 13, 40, 50, 91, 131, 133

Web Content Accessibility Guideline (WCAG) Web Content Accessibility Guideline (abbrev. WCAG) [8] is a set of guidelines produced by the World Wide Web Consortium (W3C) for ensuring web content accessibility. WCAG standards address digital content and websites' perceived ability, operability, understandability, and robustness. It is the goal of these guidelines to make web content accessible to people with disabilities, such as those who are visual, auditory, physically disabled, cognitively challenged, and neurotypical.. 9, 45, 93, 96, 114

World Health Organization (WHO) The World Health Organization (abbrev. WHO) [9] is a specialized agency of the United Nations responsible for coordinating and promoting international public health. It provides leadership on global health issues, sets health standards, and offers technical assistance and support to countries in managing health challenges... 1

World Wide Web Consortium (W3C) W3C, the World Wide Web Consortium, is an international organization that sets standards and guidelines for web technologies, ensuring interoperability and accessibility. It promotes collaboration among industry, academia, and government to advance the development of the World Wide Web. [10]. 1, 134

[6] WebAIM, https://webaim.org/s, (Accessed: 03.08.2023)
[7] WAI, https://www.w3.org/WAI/ (Access: 17.07.2023)
[8] WCAG 2 Overview, https://www.w3.org/WAI/standards-guidelines/wcag/ (Accessed: 17.07.2023)
[9] WHO, https://www.who.int/about (Accessed: 18.07.2023)
[10] W3C, https://www.w3.org/ (Access: 17.07.2023)

Bibliography

[AAVSULM18] Tania Acosta, Patricia Acosta-Vargas, Luis Salvador-Ullauri, and Sergio Luján-Mora. Method for accessibility assessment of online content editors. In *Proceedings of the International Conference on Information Technology & Systems (ICITS 2018)*, pages 538–551. Springer, 2018.

[ABD+21] Khaled Albusays, Pernille Bjorn, Laura Dabbish, Denae Ford, Emerson Murphy-Hill, Alexander Serebrenik, and Margaret-Anne Storey. The diversity crisis in software development. *IEEE Software*, 38(2):19–25, 2021.

[ABFG12] Cecilia Avila, Silvia Baldiris, Ramon Fabregat, and Juan Carlos Guevara. Accessibility evaluation improvement using case based reasoning. In *2012 Frontiers in Education Conference Proceedings*, pages 1–6. IEEE, 2012.

[ABFG14] Cecilia Avila, Silvia Baldiris, Ramon Fabregat, and Juan Carlos Guevara. A web content accessibility evaluation process for learning objects in the context of a virtual learning environment. In *New Horizons in Web Based Learning: ICWL 2011 International Workshops, KMEL, ELSM, and SPeL, Hong Kong, December 8-10, 2011, ICWL 2012 International Workshops, KMEL, SciLearn, and CCSTED, Sinaia, Romania, September 2-4, 2012. Revised Selected Papers 11*, pages 181–190. Springer, 2014.

[BBL+14] Maria Claudia Buzzi, Marina Buzzi, Barbara Leporini, Giulio Mori, and Victor MR Penichet. Collaborative editing: collaboration, awareness and accessibility issues for the blind. In *On the Move to Meaningful Internet Systems: OTM 2014 Workshops: Confederated International Workshops: OTM Academy, OTM Industry Case Studies Program, C&TC, EI2N, INBAST, ISDE, META4eS, MSC and OnToContent 2014, Amantea, Italy, October 27-31, 2014. Proceedings*, pages 567–573. Springer, 2014.

[BLO23] Dominik Bork, Philip Langer, and Tobias Ortmayr. A vision for flexibile glsp-based web modeling tools, 2023.

[BNB21] Matthew J Baker, EM Nightingale, and Suzy Bills. An editing process for blind or visually impaired editors. *IEEE Transactions on Professional Communication*, 64(3):275–287, 2021.

[BR09] Pradipta Biswas and Peter Robinson. Modelling perception using image processing algorithms. *People and Computers XXIII Celebrating People and Technology*, pages 494–503, 2009.

[CCD20] Charlie Cross, Deniz Cetinkaya, and Huseyin Dogan. Transforming diagrams' semantics to text for visually impaired. In *International Conference on Human-Computer Interaction*, pages 339–350. Springer, 2020.

[CH22] Balázs Csontos and István Heckl. Improving accessibility of cms-based websites using automated methods. *Universal Access in the Information Society*, 21(2):491–505, 2022.

[CLB22] Giuliano De Carlo, Philip Langer, and Dominik Bork. Rethinking model representation - A taxonomy of advanced information visualization in conceptual modeling. In *Conceptual Modeling - 41st International Conference*, volume 13607, pages 35–51. Springer, 2022.

[dCA21] João Fonseca de Carvalho and Vasco Amaral. Towards a modelling workbench with flexible interaction models for model editors operating through voice and gestures. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1026–1031. IEEE, 2021.

[DCLB22] Giuliano De Carlo, Philip Langer, and Dominik Bork. Advanced visualization and interaction in glsp-based web modeling: realizing semantic zoom and off-screen elements. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*, pages 221–231, 2022.

[dNBdOBdMO19] Marcos D do Nascimento, Anarosa AF Brandão, Leônidas de Oliveira Brandão, and Francisco C de MB Oliveira. Overcoming accessibility barriers for people with severe vision impairment in web-based learning environments: A literature review. In *2019 IEEE Frontiers in Education Conference (FIE)*, pages 1–8. IEEE, 2019.

[EUHMB22] Md Ehtesham-Ul-Haque, Syed Mostofa Monsur, and Syed Masum Billah. Grid-coding: An accessible, efficient, and structured coding paradigm for blind and low-vision programmers. In *Proceedings of*

the 35th Annual ACM Symposium on User Interface Software and Technology, pages 1–21, 2022.

[Gui] Accessibility Developer Guide:. Controlling a computer with a keyboard only. `https://www.accessibility-developer-guide.com/knowledge/keyboard-only/controlling-a-computer/`. Access: 03.08.2023.

[KBB+09] Barbara Kitchenham, O Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, 51(1):7–15, 2009.

[KBC+04] Alasdair King, Paul Blenkhorn, David Crombie, Sijo Dijkstra, Gareth Evans, and John Wood. Presenting uml software engineering diagrams to blind people. In *Computers Helping People with Special Needs: 9th International Conference, ICCHP 2004, Paris, France, July 7-9, 2004. Proceedings 9*, pages 522–529. Springer, 2004.

[KYKS06] Toshihiko Komada, Katsuhito Yamaguchi, Fukashi Kawane, and Masakazu Suzuki. New environment for visually disabled students to access scientific information by combining speech interface and tactile graphics. In *Computers Helping People with Special Needs: 10th International Conference, ICCHP 2006, Linz, Austria, July 11-13, 2006. Proceedings 10*, pages 1183–1190. Springer, 2006.

[LPS07] Barbara Leporini, Fabio Paternò, and Antonio Scorcia. Easing web guidelines specification. In *Web Engineering: 7th International Conference, ICWE 2007 Como, Italy, July 16-20, 2007 Proceedings 7*, pages 254–268. Springer, 2007.

[LPS+22] Hae-Na Lee, Yash Prakash, Mohan Sunkara, IV Ramakrishnan, and Vikas Ashok. Enabling convenient online collaborative writing for low vision screen magnifier users. In *Proceedings of the 33rd ACM Conference on Hypertext and Social Media*, pages 143–153, 2022.

[LVPF14] L. Luque, E. S. Veriscimo, G. C. Pereira, and L. V. L. Filgueiras. Can we work together? on the inclusion of blind people in uml model-based tasks. In P. M. Langdon, J. Lazar, A. Heylighen, and H. Dong, editors, *Inclusive Designing*, pages 223–233, Cham, 2014. Springer International Publishing.

[LW12] Claudia Loitsch and Gerhard Weber. Viable haptic uml for blind people. In *Computers Helping People with Special Needs: 13th International Conference, ICCHP 2012, Linz, Austria, July 11-13, 2012, Proceedings, Part II 13*, pages 509–516. Springer, 2012.

[LWLG21] Sehi LYi, Qianwen Wang, Fritz Lekschas, and Nils Gehlenborg. Gosling: A grammar-based toolkit for scalable and interactive genomics data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):140–150, 2021.

[MB23] Haydar Metin and Dominik Bork. On developing and operating glsp-based web modeling tools: Lessons learned from bigUML. In *Proceedings of the 26th International Conference on Model Driven Engineering Languages and Systems, MODELS 2023*. IEEE, 2023.

[MBE+23] Jonathan Giovanni Soto Muñoz, Jesús Andrés Sandoval Bringas, Israel Duran Encinas, Mónica Adriana Carreño León, and Arturo Ivan De Casso Verdugo. Source code editor using voice commands to support people with motor disabilities. *Universal Access in the Information Society*, pages 1–18, 2023.

[MBKS08] Oussama Metatla, Nick Bryan-Kinns, and Tony Stockman. Constructing relational diagrams in audio: the multiple perspective hierarchical approach. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, pages 97–104, 2008.

[MCSC+20] Jorge Morato, Adrian Campillo, Sonia Sanchez-Cuadrado, Ana Iglesias, and Olga Berrios. An accessible evaluation tool to detect easy-to-read barriers. In *Proceedings of the 9th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion*, pages 55–60, 2020.

[MDN23] MDN. Keyboard. `https://developer.mozilla.org/en-US/docs/Web/Accessibility/Understanding_WCAG/Keyboard`, 2023. Access: 03.08.2023.

[PCWFP22] Bharat Paudyal, Chris Creed, Ian Williams, and Maite Frutos-Pascual. Inclusive multimodal voice interaction for code navigation. In *Proceedings of the 2022 International Conference on Multimodal Interaction*, pages 509–519, 2022.

[PFdMF21] Débora Maria Barroso Paiva, André Pimenta Freire, and Renata Pontin de Mattos Fortes. Accessibility and software engineering processes: A systematic literature review. *Journal of Systems and Software*, 171:110819, 2021.

[PJ15] Gregor Polančič and Gregor Jošt. The impact of the representatives of three types of process modeling tools on modeler's perceptions and performance. *Journal of Software: Evolution and Process*, 28:n/a–n/a, 01 2015.

[PRG15] Afra Pascual, Mireia Ribera, and Toni Granollers. Empathic communication of accessibility barriers in web 2.0 editing. In *Proceedings of the 12th International Web for All Conference*, pages 1–8, 2015.

[RJ11] Diana Ruth-Janneck. An integrative accessibility engineering approach using multidimensional classifications of barriers in the web. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, pages 1–7, 2011.

[SG15] Stephan Seifermann and Henning Groenda. Towards collaboration on accessible uml models. In *Mensch & Computer Workshopband*, pages 411–417, 2015.

[SG16] Stephan Seifermann and Henning Groenda. Survey on textual notations for the unified modeling language. In *2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 28–39. IEEE, 2016.

[SKTA09] Daisuke Sato, Masatomo Kobayashi, Hironobu Takagi, and Chieko Asakawa. What's next? a visual editor for correcting reading order. In *Human-Computer Interaction–INTERACT 2009: 12th IFIP TC 13 International Conference, Uppsala, Sweden, August 24-28, 2009, Proceedings, Part I 12*, pages 364–377. Springer, 2009.

[SMB23] Aylin Sarioglu, Haydar Metin, and Dominik Bork. How inclusive is conceptual modeling? a systematic review of literature and tools for disability-aware conceptual modeling. In *Proceedings of the 42nd International Conference on Conceptual Modeling (ER 2023)*. Springer, 2023.

[SP07] Martina Schulz and Michael Pieper. Web compliance management: barrier-free websites just by simply pressing the button? accessibility and the use of content-management-systems. *LECTURE NOTES IN COMPUTER SCIENCE*, 4397:419, 2007.

[SSO+20] Sheila Rizzato Stopa, Célia Landmann Szwarcwald, Max Moura de Oliveira, Ellen de Cassia Dutra Pozzetti Gouvea, Maria Lúcia França Pontes Vieira, Marcos Paulo Soares de Freitas, Luciana Monteiro Vasconcelos Sardinha, and Eduardo Marques Macário. National health survey 2019: history, methods and perspectives. *Epidemiologia e Serviços de Saúde*, 29:e2020315, 2020.

[TB19] Márcio Josué Ramos Torres and Regina Barwaldt. Approaches for diagrams accessibility for blind people: a systematic review. In *2019 IEEE Frontiers in Education Conference (FIE)*, pages 1–7. IEEE, 2019.

[TBdR05] Eduardo Hideki Tanaka, Sílvia Amélia Bim, and Heloísa Vieira da Rocha. Comparing accessibility evaluation and usability evaluation in hagáquê. In *Proceedings of the 2005 Latin American conference on Human-computer interaction*, pages 139–147, 2005.

[TBP+20] Márcio Josué Ramos Torres, Regina Barwaldt, Paulo Cesar Ramos Pinho, Luiz Oscar Homann de Topin, and Tiago Fossati Otero. An auditory interface to workspace awareness elements accessible for the blind in diagrams' collaborative modeling. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–7. IEEE, 2020.

[VRSGLM18] Jorge Villarroel-Ramos, Sandra Sanchez-Gordon, and Sergio Luján-Mora. Architectural metamodel for requirements of images accessibility in online editors. In *2018 International Conference on Information Systems and Computer Science (INCISCOS)*, pages 312–319. IEEE, 2018.

[W3C18a] W3C. How to meet wcag (quick reference) - keyboard accessible. `https://www.w3.org/WAI/WCAG21/quickref/#keyboard-accessible`, 2018. Access: 03.08.2023.

[W3C18b] W3C. Web Content Accessibility Guidelines (WCAG) 2.1. `https://www.w3.org/TR/WCAG21/`, 2018. Access: 18.09.2022.

[Web17a] WebAIM. Diverse Abilities and Barriers. `https://www.w3.org/WAI/people-use-web/abilities-barriers/`, 2017. Access: 20.04.2023.

[Web17b] WebAIM. Diverse Abilities and Barriers - Cognitive, Learning & Neurological. `https://www.w3.org/WAI/people-use-web/abilities-barriers/#cognitive`, 2017. Access: 15.06.2023.

[Web17c] WebAIM. Diverse Abilities and Barriers - Physical. `https://www.w3.org/WAI/people-use-web/abilities-barriers/#physical`, 2017. Access: 15.06.2023.

[Web18] WebAIM. Web Content Accessibility Guidelines (WCAG) 2.1: Keyboard Accessible. `https://www.w3.org/TR/WCAG21/#keyboard-accessible`, 2018. Access: 16.06.2023.

[Web21] WebAIM. WebAIM's WCAG 2 Checklist. `https://webaim.org/standards/wcag/checklist`, 2021. Access: 18.09.2022.

[Web22a] WebAIM. The 2022 report on the accessibility of the top 1,000,000 home pages. `https://webaim.org/projects/million/2022`, 2022. Access: 18.09.2022.

[Web22b] WebAIM. Keyboard accessibility. https://webaim.org/techniques/keyboard/, 2022. Access: 03.08.2023.

[Web23] WebAIM. The 2023 report on the accessibility of the top 1,000,000 home pages. https://webaim.org/projects/million/, 2023. Access: 29.04.2023.

[WSGK20] Fabian Wildhaber, Nadim Salloum, Marcel Gygli, and Andrea Kennel. Self-directed creation and editing of uml class diagrams on mobile devices for visually impaired people. In *2020 IEEE Tenth International Model-Driven Requirements Engineering (MoDRE)*, pages 49–57. IEEE, 2020.

[WW07] Gerhard Weber and Kurt Weimann. Editing a test suite for accessibility of interactive web sites. In *Universal Access in Human-Computer Interaction. Applications and Services: 4th International Conference on Universal Access in Human-Computer Interaction, UAHCI 2007 Held as Part of HCI International 2007 Beijing, China, July 22-27, 2007 Proceedings, Part III 4*, pages 193–201. Springer, 2007.

[Zde19] Sean Zdenek. Guest editor's introduction: Reimagining disability and accessibility in technical and professional communication. *Communication Design Quarterly Review*, 6(4):4–11, 2019.