Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Development of a mobile web browser user interface for the blind

Bc. Jan Charvát

Supervisor: Doc. Ing. Daniel Novák, PhD.
Field of study: Cybernetics and Robotics
Subfield: Cybernetics and Robotics
April 2019

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Charvát** Jméno: **Jan** Osobní číslo: **420279**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Kybernetika a robotika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Vývoj uživatelského rozhraní pro webový prohlížeč pro nevidomé

Název diplomové práce anglicky:

Development of a mobile web browser user interface for the blind

Pokyny pro vypracování:

1. Seznamte se s existujícím mobilním systémem pro nevidomé uživatele
2. Seznamte se s existujícími opensource mobilními HTML renderovacími jádry. Především prozkoumejte možnosti použití jádra Gecko a jádra Blink a vyberte vhodného kandidáta pro integraci do existujícího systému.
3. Navrhněte a implementujte uživatelské rozhranní uzpůsobené pro nevidomé uživatele využívající zvolené renderovací jádro.
4. Výsledný koncept otestujte na 5 nevidomých nebo slabozrakých uživatelích

Seznam doporučené literatury:

[1] Petr Svobodník. “Zpřístupnění mobilních telefonů se systémem Android pro nevidomé uživatele”. Diploma thesis. Czech Technical University in Prague, 2013.
[2] Mozilla Foundation. Gecko repository on Github. https://github.com/jostw/gecko/tree/master/mobile/android
[3] Chromium contributors. Chromium/Blink repository. https://chromium.googlesource.com/chromium/blink
[4] Ritwika Ghose, T. Dasgupta and A. Basu. Architecture of a web browser for visually handicapped people. 2010 IEEE Students Technology Symposium (TechSym), Kharagpur, 2010, pp. 325-329.
[5] J. Cofino, A. Barreto, F. Abyarjoo and F. R. Ortega. Sonifying HTML tables for audio-spatially enhanced non-visual navigation. 2013 Proceedings of IEEE Southeastcon, Jacksonville, FL, 2013, pp. 1-5.

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Daniel Novák, Ph.D., Analýza a interpretace biomedicínských dat FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **26.02.2019** Termín odevzdání diplomové práce: **28.05.2019**

Platnost zadání diplomové práce:
do konce letního semestru 2019/2020

doc. Ing. Daniel Novák, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta

Acknowledgements

Foremost, I would like to thank my supervisor Doc. Ing. Daniel Novák, PhD. for offering an interesting topic of investigation and his support. Furthermore, I would like to thank Czech Blind United (SONS) for valuable comments and willingness to help with user testing.

I would also like to thank my parents for their continued moral and financial support throughout my studies.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date ....

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne ....

....

signature

Abstract

This thesis deals with the development of a mobile web browser for visually impaired people (i.e., blind and low vision). The user interface was developed concerning the specific needs of the target user group. The resulting accessible mobile web browser was integrated into the existing Core system developed for visually impaired people. Integrated web browser was tested by visually impaired users in cooperation with Czech Blind United (SONS).

Keywords: mobile web browser, web accessibility

Supervisor: Doc. Ing. Daniel Novák, PhD.
Department of Cybernetics,
Karlovo náměstí,
Praha 2

Abstrakt

Tato práce se zabývá vývojem mobilního prohlížeče pro lidi se zrakovým postižením (tj. slepce a slabozraké). Uživatelské rozhraní prohlížeče bylo navrhnuto s ohledem na specifické potřeby cílové uživatelské skupiny. Výsledný přístupný prohlížeč byl integrován do existujícího systému, který byl vyvinut pro zrakově postižené uživatele. Integrovaný prohlížeč byl testován zrakově postiženými uživateli za spolupráce se Sjednocenou organizací nevidomých a slabozrakých (SONS).

Klíčová slova: mobilní webový prohlížeč, přístupnost webu

Překlad názvu: Vývoj uživatelského rozhranní pro webový prohlížeč pro nevidomé

Contents

Figures

Tables

Chapter 1

Introduction

According to the World Health Organization (WHO) estimates, 1.3 billion people live with some form of vision impairment, 217 million of them have moderate to severe vision impairment (VI), and 36 million people are blind [35]. The leading causes of VI are uncorrected refractive errors, cataract, age-related macular degeneration, glaucoma, diabetic retinopathy, corneal opacity, and trachoma. An experience of VI varies upon many different factors (e.g., availability of prevention and treatment or accessibility of public space or information space). Vision loss is the most serious sensory disability, causing approximately 90% deprivation of entire multi-sense perception for an individual [25]. VI has a significant impact on an individual's quality of life (QoL), including the ability to work and to develop personal relationships. Almost half of the visually impaired people (VIP) fell "moderately" or "completely" cut off from people and things around them [25].

The Assistive technology (AT) is commonly considered to be technology designed for people with some form of impairment or seniors, and by definition, it is a broad topic involving technologies, equipment, devices, apparatus, services, systems, processes and environmental modifications [2]. AT has a very relevant social impact, allowing impaired people to live more independently and perform activities of daily life (ADL) effectively and safely, which leads to increasing overall QoL. Regrettably, due to our ever-increasing aging and blind populations, AT has the potential to impact our QoL in the future broadly. Research on assistive technology for the visually impaired has traditionally focused on *mobility*, *navigation*, *object recognition* and *printed information access.*

Over the last decade, there has been the expansion of smart mobile technology, and it became an integral part of our everyday life. According to [30] the smartphones dominates over the cell phones in advanced economies, but also most of the emerging ones. It's estimated that in developed countries 76% of adults owns a smartphone and in developing countries 45%. What is more interesting for this thesis, 90% of adults from developed countries and 60% from developing countries use the internet (see Figure 1.1). A significant part of them also tends to use social media. It reveals how fundamental the internet became over the years.

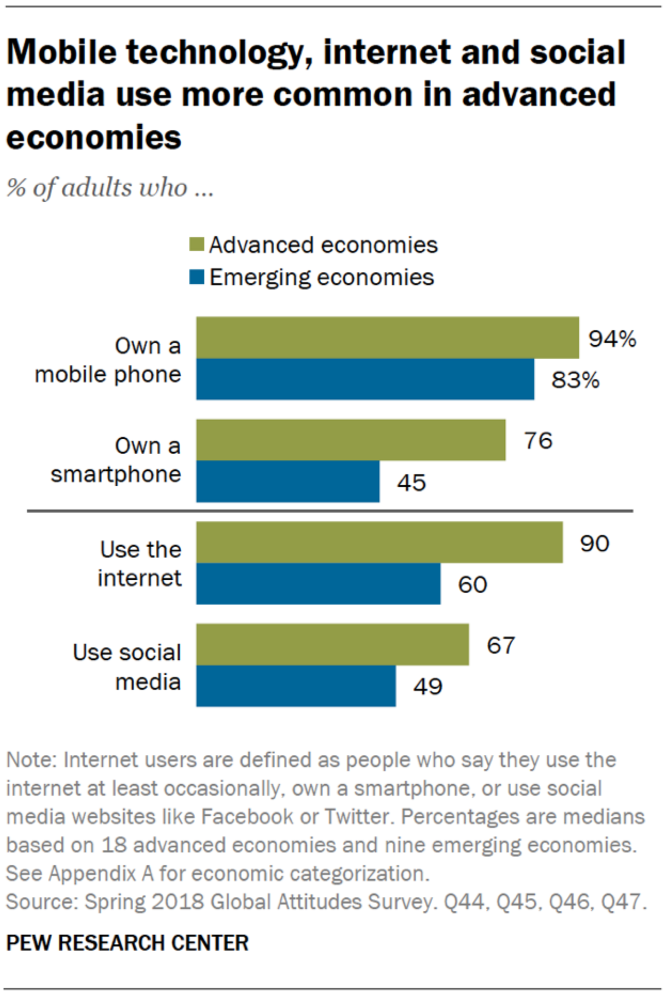


Figure 1.1: Mobile technology, internet and social media use more common in advanced economies.

The rise of available, omnipresent and ever more powerful smartphones brought new opportunities in the field of AT [25]. One of the advantages of modern smart-phone used as a universal accessibility tool is the fact that most people tend to carry it with themselves most of the day, so it is more convenient than any other specialized accessibility devices. Multipurpose, customizable and affordable mobile devices, became the platform for delivering a large number of diverse ATs in the form of single-purpose applications or more complex software systems modifying the device behavior for special needs. However, mainstream mobile devices are typically visually demanding, which makes them not accessible enough for individuals with VI. This need led to the exploration of usage of sensory modalities other than vision became the most powerful tool in making these devices accessible. The most explored sensor modalities are

1. speech recognition,
2. non-speech auditory feedback,
3. haptic feedback,
4. multimodal input,
5. text-to-speech (TTS),
6. gesture recognition.

At present, most of the mainstream smart-phones offer various accessibility tools making these devices accessible for people having various types of impairment. Moreover, the developers of mobile applications are encouraged to follow the correct accessibility design to attract more users with disabilities [17].

1.1 Web accessibility

Accessing the internet is essential for most of us and is widely used for shopping or communication among many others. Internet presents the most significant source of written information in human history, however, without special tools or browsers, this useful source of information is often inaccessible to the people with disabilities, in our case to the people who are VI.

The well-working web accessibility solutions are built on following pillars: a) web design w.r.t accessibility (responsibility of web developers and designers), b) usage of accessibility tools like screen readers or browsers (responsibility of accessibility tools developers). The correct design of a web page with accessibility in mind can prevent some common tiring problems, e.g., search button with missing or improper description.

Examples of some recommended best practices for accessible web design according to [34] are

1. Don't use completely inaccessible elements like Flash.
2. Do not use popup windows. They break the back button, confuses the user and causes lost focus-related problems.
3. Put alt tag on images used for links and empty alt tag for any other image, so that the users are not forced to iterate over a large number of image descriptions.
4. Use label tag for input fields.

According to the survey on preferences of screen reader users [33] based on responses from 1792 participants, the approximately 85% of respondents answered that 'Better (more accessible) web sites' would have a more significant impact on accessibility than 'Better assistive technology.'

Chapter 2

The Core system and other related work

This work focuses on design and implementation of a mobile web browser for the blind and VIP and following integration into the phone powered by the Android-based *Core system* [31] developed for the VIP. To my best knowledge, there is no specific mobile web browser for the blind; the existing solutions are regular mobile web browsers made accessible through multipurpose accessibility tools (e.g., TalkBack on Android OS, or VoiceOver on iOS). These accessibility tools will be discussed in the following Sections for comparison. Some of the existing desktop web browsers for the VIP will be discussed in Section 2.1, because many problems related to web accessibility affects both desktop and mobile web browsers.

The application called *Core system* was developed as a Master thesis of Petr Svobodník from Czech Technical University [31] to make Android-powered devices more accessible through UI with clear, simplistic design philosophy customized for needs of people who have vision loss. The existing *Core system* is a target of a proposed mobile web browser integration, and principles of this specialized UI will directly influence the design of a suggested mobile web browser, for that reason the *Core system* is described in detail in Section 2.4.

2.1 Desktop browsers for visually impaired

Several attempts have been taken to build a web browser for the VIP. One of them was **Home Page Reader**. Home Page Reader was a commercial web browser developed for VIP by the company IBM and currently is not offered for purchase anymore. Home Page Reader used extraction and filtering of text and TTS output with different gender voices for reading a text (male) and links (female). The product did not support any speech input method, hierarchical representation of the pages, text-to-Braille or mouse gestures recognition. Compared to IBM's Home Page Reader, **eGuideDog** [36] is a open-source browser for VIP and features integrated TTS, an advanced text extraction algorithm and hierarchical page representation. The project also offers many useful accessibility tools for both users and developers. Another browser for VIP is **WebbIE** [27], which features text extraction, image processing (removes or replaces images with alternative description) and plain text representation of the page for ease of use with screen readers. WebbIE is a closed-source solution but is offered free of charge under the GNU Public License. Similarly to eGuideDog, the project offers more accessibility tools than just a browser. **Firefox** browser [9], which is the most often used browser among the screen reader users according to [33] (see Figures 2.1 and 2.2), offers many accessibility tools packed as add-ons [7], which serve various purposes from web page simplification for screen reading to generating document-maps based on a headings structure. **MozBraille** [8] is one of the Firefox extensions, which transforms it into the standalone accessible web browser designed for VIP. MozBraille features TTS, text to Braille conversion, digital magnifier and Braille terminal for blind developers. Another open-source web browsers for the blind, **ShrutiDrishti** [29], also combined TTS output with text to Braille conversion and provided a user-friendly environment and digital magnifier as well. Another web browser for VIP featuring text to Braille conversion is **BrailleSurf** [3].

We can conclude that TTS is the essential feature of the browser for the blind and was integrated into all examined browsers. Some form of text extraction and filtering was included in all implementations as well. However, very few browsers implement text to Braille conversion or automatic speech recognizer. Another significant limitation of most of the presented browsers is reading the page content in a sequential pattern.

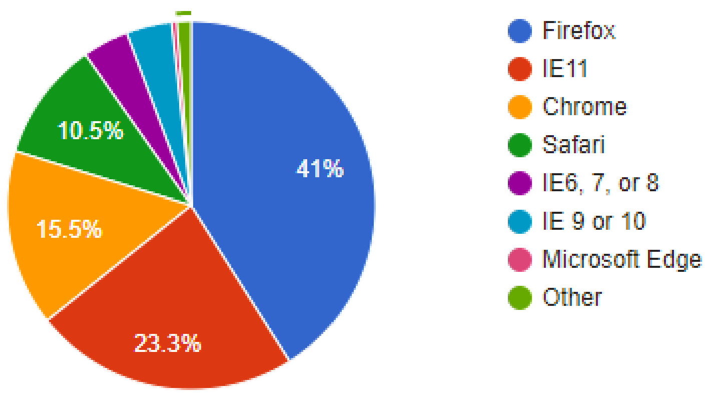


Figure 2.1: According to the survey on preferences of screen readers users [33] is Firefox the most used browser among the screen readers users.

2.2 The TalkBack

The TalkBack service is open-source [20] accessibility solution, which comes pre-installed as part of the Google Android Accessibility Suite [21]. Technically

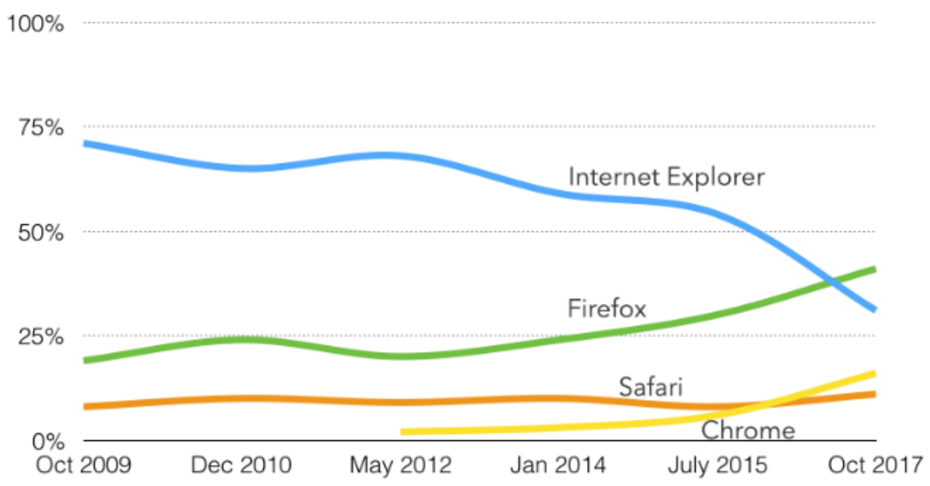


Figure 2.2: According to the survey on preferences of screen readers users [33], Internet Explorer is not the most used browser among the screen readers users for the first time. The graph also reveals that the popularity of Firefox and Chrome is stably increasing.

TalkBack is implemented as Accessibility Service [18] and is considered to be the standard way of making Android touch screen phones accessible.

TalkBack uses an excessive number of gestures for navigation, and some users find it too complex and hard to learn. Most important gestures are enumerated in Table 2.1.

Action	Gesture
Move to next item on screen	Swipe right
Move to previous item on screen	Swipe left
Cycle through navigation settings	Swipe up or down
Select focused item	Double-tap
Move to first item on screen	Up then down
Move to last item on screen	Down then up
Scroll forward	Right then left
Scroll back	Left then right
Move slider up	Right then left
Move slider down	Left then right
Home button	Up then left
Back button	Down then left
Overview button	Left then up
Notifications	Right then down
Screen search	Left then down
Open local context menu	Up then right
Open global context menu	Down then right

Table 2.1: TalkBack gestures.

The main reason behind focusing on the TalkBack Service is one of its modules that makes Android-based web browsers accessible [23]. From our point of view, the most interesting part of the TalkBack web module (see Figure 2.3) is specific navigation based on the following options:

1. **Headings.** Navigate by headings (level 1-6).
2. **Links.** Navigate by different kinds of links, such as visited, unvisited, or active.
3. **Controls.** Navigate by other elements, such as form fields, buttons, or menus.
4. **Characters, words, or lines.** Explore one character, word, or line at a time.
5. **Landmarks.** Navigate by ARIA landmarks, such as "main" or "navigation." (Available only in the local context menu.)
6. **Special content.** Explore content such as tables. (Available only in the local context menu.)
7. **Default.** Explore every element on the page in order.

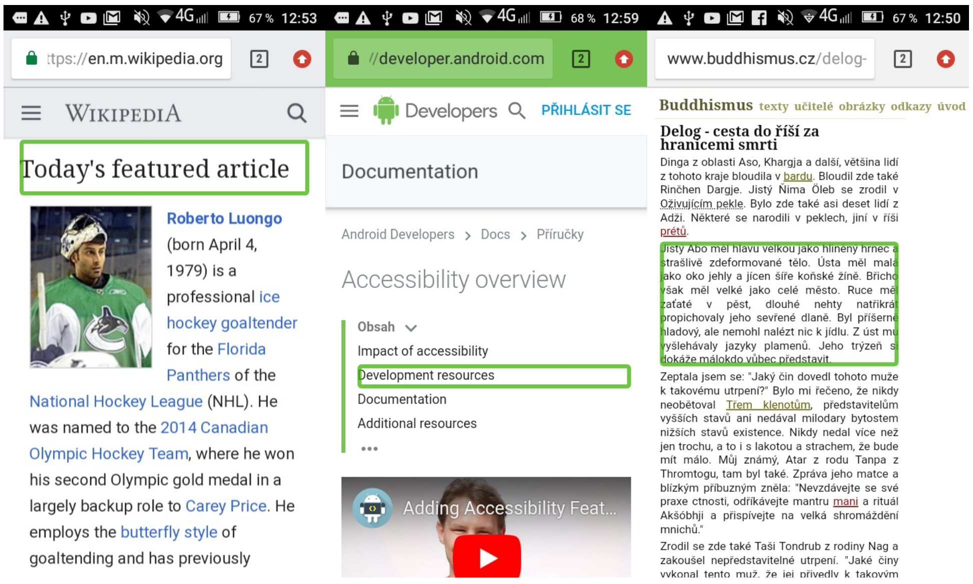
These navigation options represent a reasonable and effective way of navigation in a web page structure. These navigation options can be cycled using down and up gestures. Linear navigation through page content items is done by left and right swipe gestures.

2.3 Other accessibility solutions

As said in the Introduction, the smart-phones became the universal platform for delivering various accessibility tools. Some of them are single-purpose utility applications; others are far more complex. This Section discusses some other solutions relevant to this topic.

2.3.1 The VoiceOver

VoiceOver is a gesture-based screen reader included in iOS [26] and servers similar purpose like TalkBack service in Android systems. VoiceOver makes both system or third-party applications accessible. Gesture-based control is somewhat analogical to the gesture control of TalkBack described in Section 2.2. The text input is handled in a standard way; each character on the keyboard is read aloud when the user touches it. Some other input methods like handwriting or speech input are supported as well. VoiceOver incorporates the so-called Rotor, which enables cycling trough navigation



Today's featured article

Roberto Luongo (born April 4, 1979) is a professional ice hockey goaltender for the Florida Panthers of the National Hockey League (NHL). He was named to the 2014 Canadian Olympic Hockey Team, where he won his second Olympic gold medal in a largely backup role to Carey Price. He employs the butterfly style of goaltending and has previously

Documentation

Android Developers › Docs › Příručky

Accessibility overview

Obsah
Impact of accessibility
Development resources
Documentation
Additional resources

Adding Accessibility Feat...

Buddhismus texty učitelé obrázky odkazy úvod

Delog - cesta do říší za hranicemi smrti

Dinga z oblasti Aso, Khargja a další, většina lidí z tohoto kraje bloudila v bardu. Bloudil zde také Rinčhen Dargje. Jistý Ňima Öleb se zrodil v Oživujícím pekle. Bylo zde také asi deset lidí z Adži. Některé se narodili v peklech, jiní v říši pretů.

Jistý Abo měl hlavu velkou jako hliněný hrnec a strašlivě zdeformované tělo. Ústa měl malá jako oko jehly a jícen šíře koňské žíně. Břicho však měl velké jako celé město. Ruce měl zaťaté v pěst, dlouhé nehty natřikrát propichovaly jeho sevřené dlaně. Byl příšerně hladový, ale nemohl nalézt nic k jídlu. Z úst mu vyšlehávaly jazyky plamenů. Jeho trýzeň si dokáže málokdo vůbec představit.

Zeptala jsem se: "Jaký čin dovedl tohoto muže k takovému utrpení?" Bylo mi řečeno, že nikdy neobětoval Třem klenotům, představitelům vyšších stavů ani nedával milodary bytostem nižších stavů existence. Nikdy nedal více než jen trochu, a to i s lakotou a strachem, že bude mít málo. Můj známý, Atar z rodu Tanpa z Thromtogu, tam byl také. Zpráva jeho matce a blízkým příbuzným zněla: "Nevzdávejte se své praxe ctnosti, odříkávejte mantru mani a rituál Akšóbhji a přispívejte na velká shromáždění mnichů."

Zrodil se zde také Taši Tondrub z rodiny Nag a zakoušel nepředstavitelné utrpení. "Jaké činy vykonal tento muž, že jej přivedly k takovým

Figure 2.3: Several web pages opened in Chrome browser and explored using TalkBack service. The green bounding rectangle is a virtual cursor (a.k.a accessibility focus), which marks a current selection, whose content and type is read by TTS.

modes (including headings, links, or images) similarly like TalkBack and other smart-phone focused accessibility solutions when reading web pages or PDF documents. Service also features Image recognition tool (triggered by triple tap) capable of image description to the user, even in the case when the image was not annotated. Another useful feature of VoiceOver is support for external Braille keyboards and displays, which is not implemented even in the most of desktop browsers for blind.

2.3.2 The Corvus

Corvus [5] is a commercial gesture-based 'accessibility kit' for Android developed by Slovak company Stopka and consists of two main parts

1. screen reader integrated into Android similarly to TalkBack, but having own gestures,
2. special user interface tailored for VI users similarly to the *Core system*.

Corvus includes some accessibility features like a light detector or QR code scanner intended for object tagging.

2.4 The Core system

The *Core system* refers to the application, whose development started as a Master thesis of Petr Svobodník [31]. This section describes the UI design philosophy of the *Core system*, because it has a direct impact on browser design and integration into the system. The *Core system* is built using a very minimalistic UI without the use of graphical elements. The core of the *Core system* UI is a textual and contrast (white on black by default) screen, which usually displays short labels in big font. See Figure 2.4 for an example of described UI screens used by the *Core system*.

The *Core system* is capable of running on smartphones with or without a touchscreen. The target hardware phone used in the thesis is a smartphone equipped with tactile buttons. For a more detailed technical description of a target device see Subsection 4.4.2.

The main menu of the *Core system* has a structure of a directed graph (almost tree with a few exceptions). See Figure 2.5 for representative subset of the menu graph. The navigation in a menu structure can be achieved by using the following gestures (on a touchscreen) or keys in case of our target device

1. pressing left d-pad button or tapping in the left half of the screen navigates the user to the previous item in the list,
2. pressing right d-pad button or tapping in the right half of the screen navigates the user to the next item in the list,
3. pressing center d-pad button or long pressing anywhere on the screen navigates the user to the next level of the menu tree,
4. pressing back button or long pressing anywhere on the screen with two fingers navigates the user to the previous level of the menu tree.

For the quick navigation in menus can be used the numeric keyboard shortcuts as well (e.g., key three navigates the user to the third menu item). After the transition to another menu item, the user is informed about the text and position of the item using auditory speech feedback. The *Core system* allows repeated speaking or spelling of the item's text. The text input is implemented using various types of keyboards (text, numeric, time, date) with support for dictation input. The control style of the Browser application should be consistent with the *Core system* as much as possible.

The *Core system* further offers some vision based accessibility tools like QR code scanner supposed to tag object of daily living or color classifier.

The most significant advantage of *Core system* is UI developed for VIP, but this approach limits the possibility of supporting a third-party application because users are expected to use applications built-in the *Core system*.

The browser will not be designed as a stand-alone application, but an integral part of the *Core system*. The other parts of the browser (e.g., menu or dialogs) should be consistent with the *Core system* and use its standard UI components.

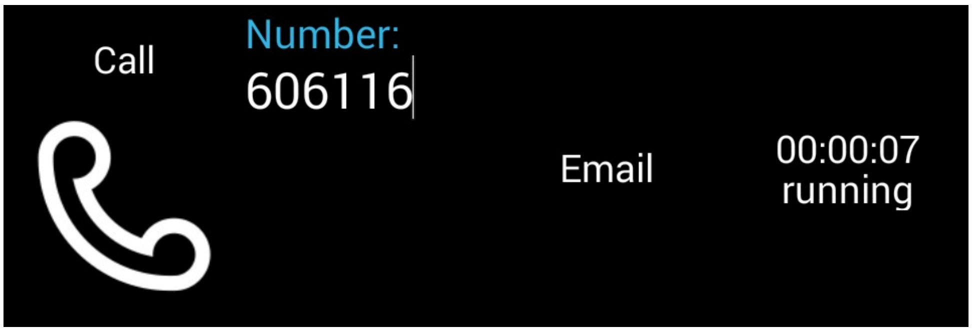


Figure 2.4: Examples of UI components used by the *Core system*. From left to right: a) Top level iconic menu item, b) Numeric keyboard for entering a number to dial, c) Lower level menu item and d) Running Stopwatch activity.

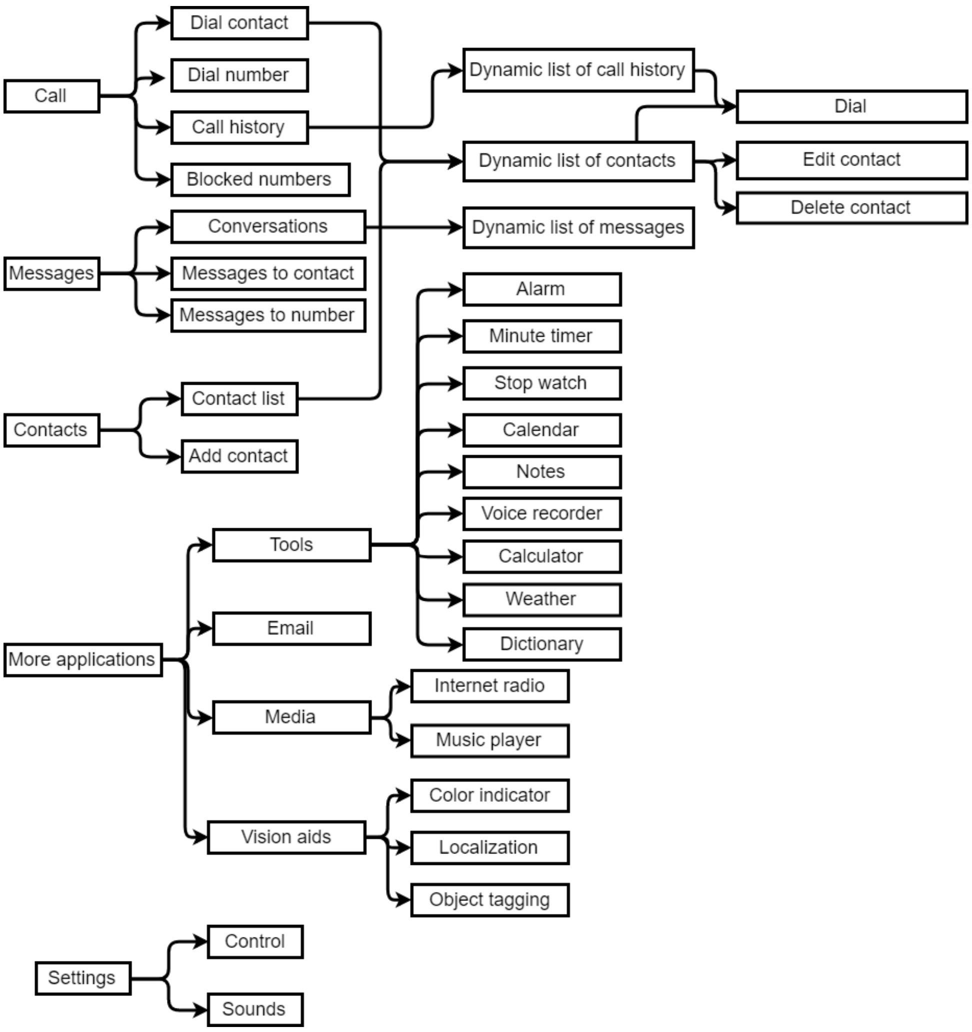


Figure 2.5: Subset of the *Core system* menu graph.

2.5 Conclusion

In this chapter, we discussed existing work related to the thesis topic. Existing desktop web browsers for VIP were analyzed in Section 2.1. TalkBack service was discussed in Section 2.2, because it represents a standard way of making Android-based smart-phones accessible and its web module exhibits an effective way of navigation in the page content. Other relevant accessibility solutions were described in Section 2.3 as well. One of them, VoiceOver presents a very similar style of navigation in structured documents and other exciting features like automatic image annotation. The target of a proposed mobile web browser integration called *Core system* was examined in detail in Section 2.4, because its UI design philosophy, principles, and properties will directly influence the design of a browser documented in the following Chapter.

Chapter 3

Web browser design

This chapter deals with the design of a web browser user interface (UI). Initial considerations related to a web browser and the World Wide Web (WWW) are discussed in Section 3.1. Some open-source browser engines available on the Android platform are examined to choose a suitable candidate for browser implementation and integration into the *Core system* in Section 3.2. Requirements on the application are formulated in Section 3.3 and later becomes the subject of the input UI (Section 3.4) and the output UI (Section 3.5) designs, which are documented separately.

3.1 Introduction

The World Wide Web is a pervasive and diverse global scale network formed by various technologies and standards (both hardware and software) at all ISO/OSI levels. However, the multiplatform World Wide Web is kept together by its core standards like Hypertext Mark-up Language (HTML), Cascading Style Sheets (CCS) or JavaScript at the highest level of ISO/OSI levels, or TPC/IP and UDP protocols from the transport layer. The proposed mobile web browser is built on top of the protocol stack. Thus the most relevant is a tree structure of an HTML page because it is subject to information extraction and filtering algorithm and navigation algorithm as well. Additional semantic information provided by meaningful mark-ups of text is used to give the blind user context.

See Figure 3.1 for the list of the most often used HTML elements. The graph is based on an analysis of 8 million samples of web pages [6]. HTML tags for link and headings often used for faster navigation to the area of interest are present among the most often used tags. Link tag occurred in 86.5% of pages and heading of level 1 on more than 55%. The tags form and input, which often serves for navigation purposes as well, are also present in the list.

There are different types of pages depending on the purpose and content, but exact classification can be possibly done in various ways. One of the

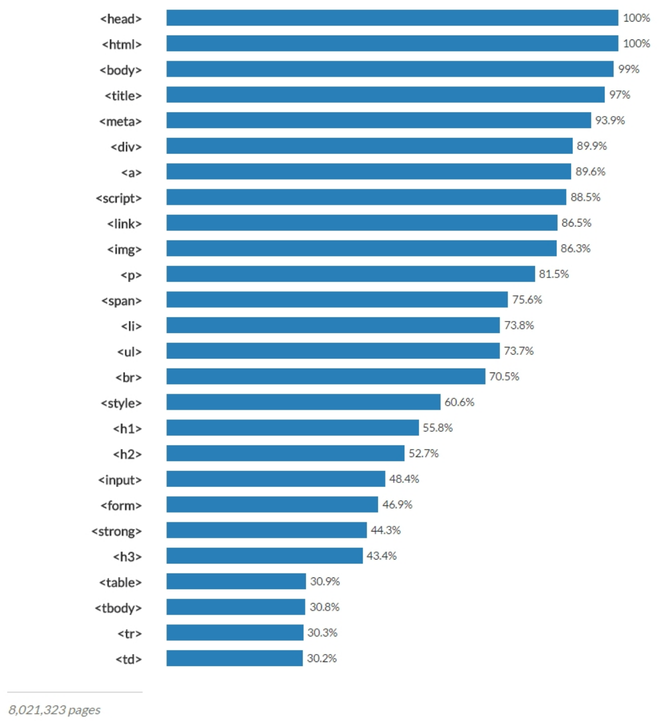


Figure 3.1: The 26 most often used HTML elements ordered by frequency according to [6].

possible classifications of the web pages might look like follows

1. single article page,
2. multiple article page,
3. e-mail,
4. search engine,
5. portal,
6. blog,
7. form,
8. social network,
9. forum,
10. online shop or auction web site.

There is a need for some classification of web pages to evaluate the ease of use of a proposed mobile web browser on a sample from each category.

3.2 Web browser engines

The web browser engine is the software capable of transforming an HTML document to its visual form. The engine draws structured text from a document, and formats it based on provided style declarations (most often CCS). Rendering engines are usually written in C++ and JavaScript. This section discusses some open-source candidate browser engines for the proposed browser implementation.

3.2.1 WebKit

WebKit [1] is a web browser engine written in C++ and initially used only by macOS. However, currently is multiplatform and available for Linux, Windows, Apple iOS, Android, and other. WebKit is used by Safari, AppStore and many other applications on macOS, iOS and Linux. Google originally incorporated WebKit for its Chrome, but starting from version 28 it was replaced with Blink engine, which is a fork of WebKit's component WebCore.

3.2.2 Blink

Blink [4] is a web browser engine written in C++ and used in default Android browser Chrome, Opera (from version 15), Microsoft Edge, other Chromium-based browsers and other projects. Blink is developed as a part of the Chromium Project with contributions from Google, Opera, Intel, or Samsung, among many others. Blink is available on many platforms, e.g., MS Windows, Linux, macOS, or Android. The Android's built-in WebView [24] component (capable of displaying web content) is currently based on Blink, which replaced WebKit (similarly to Chrome).

3.2.3 Gecko and GeckoView

Gecko [28] is a web browser engine developed by Mozilla [14]. It is a well-known engine used in Firefox browser [9], Thunderbird email client, and many other applications for displaying web pages. Gecko is free (under the Mozilla Public License) and open-source software written in C++ and JavaScript. Mozilla officially supports its use on Android, Linux, macOS, and Windows. Gecko offers a programming API that makes it suitable for a wide variety of internet-based applications.

GeckoView [13] is a reusable Android library that wraps the native Gecko engine. Firefox Reality [11], Firefox Focus [10], and other Android applications are powered by GeckoView, which serves a similar purpose like Android's built-in WebView, which was never intended for building web browsers and lacks some of the features of fully-developed browsers. Moreover, it is impossible to know precisely which engine (and what version) will power a WebView on client devices.

Compared to WebView, GeckoView is a standalone library bundled with the client application, and developers can be sure the tested code is the code that will run. Moreover, GeckoView exposes the entire power of the Web, including being suitable for building browsers, and like Firefox, it offers excellent support for modern Web standards. GeckoView is bundled with configurable content blocking feature and does not require any third party plugins while providing safety for the users. Furthermore, Gecko emphasizes accessibility [15] and strives to make its software accessible.

One of Mozilla's accessibility concepts used for page content interpretation is Gecko accessible role, for the complete list of these roles see [12]. Gecko roles provided to the client application offer very relevant additional contextual information about page elements.

The main reason for choosing Gecko (i.e., GeckoView) over the other open-source rendering engine Blink [4] suggested in the assignment is Gecko's support of accessibility features.

It is worth mentioning that GeckoView library is still in development, and although some stable version exists, this application uses possibly problematic nightly builds for the benefit of the latest API.

3.3 Requirements

In the beginning, the required functionality of a proposed web browser should be defined. The web browser should support the following standard features

1. enter a web address or search text using some search engine,
2. navigate forward or backward in the page history,
3. reload the page,
4. find text in the page,
5. bookmarks management (list, open or delete bookmark),
6. history management (list and clear history, open or delete page in history),
7. content blocking settings (categories of content to block, e.g., ads or social trackers),
8. open private session,
9. downloaded files management,
10. homepage setting.

These features are standard and well known from most of the web browsers. Web browser for blind people has also satisfy some specific additional requirements related to impairment. These mandatory requirements are set subsequently

1. provide suitable (most often auditory or vibration) output in reaction to each user action,
2. repeat speech text for current virtual cursor selection,
3. repeat title of the current page,
4. implement suitable text extraction and filtering algorithm,
5. implement a simple and effective mechanism of navigation in a web page content,
6. provide TTS related settings,
7. support earcons (i.e., sounds that announce events),
8. implement TTS output concerning overwhelming with auditory feedback,
9. support voice input method.

3.4 Input interface design

There are two main aspects of the input UI design. First of them concerns which input methods will be supported and second which actions and functions will be assigned to the individual commands (e.g., button or recognized word). The proposed input user interface mostly relies on tactile hardware buttons and uses them as the primary source of user input. Navigation-related operations and other core functionality of the browser will be commanded through a keyboard. Subsection 3.4.1 describes functions assigned to tactile buttons. Dictation input method will be incorporated only for text inputs as a complementary method to keyboard, but it will not be able to input any in-page navigation related commands.

3.4.1 Controls and navigation in the page

There are two fundamental aspects of web surfing

1. surfing the content of a web page,
2. navigation through web pages,
   a. navigation in page history (back or forward),
   b. navigation through links.

Navigation through page history is straightforward function and will be handled in the browser menu. User will have the option to go back, forward or open an arbitrary item from history. Navigation among the documents through links and exploration of these links in the pages will be the subject to the main activity. Navigation in a web page content is the fundamental problem in both mobile and desktop web browsers for the blind. Against the regular user, VI user is unable to summarize a web page content at the first look visually. The most severe disadvantage of sequential screen reader approach is the need to explore possibly lenghty page to reach a single line or link of interest. It is a reason why browsers for the blind offer another navigation or search options. For navigation in web pages are most often used these elements: headings, links or landmarks, or integrated finder, based on preferences of screen readers users on preferred finding method of desired information in the page (see Figure 3.2).

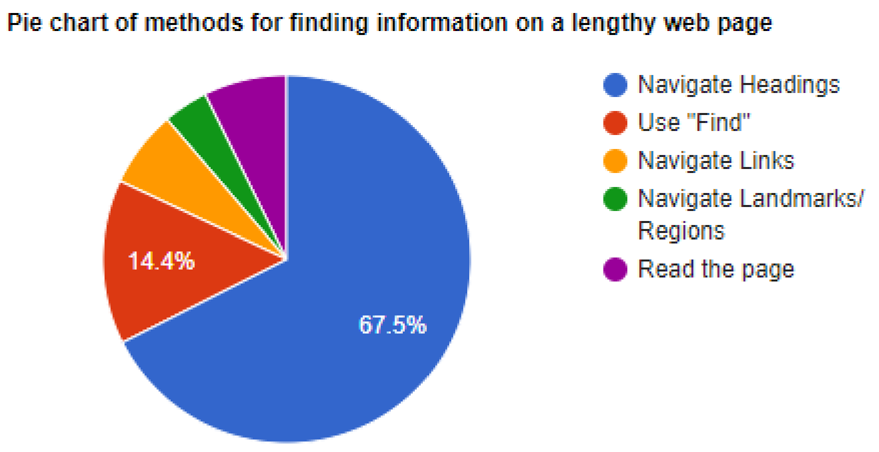


Figure 3.2: Most often used navigation methods for finding information on a lengthy web page according to the survey on preferences of screen reader users [33].

The suggested navigation modes are inspired by existing solutions, and due to the consultations with representatives of Czech Blind United (SONS) it is verified and efficient approach and users are familiar with it. The implemented navigation modes are

1. **Default.** Explore every element on the page in order.
2. **Headings.** Navigate by headings (level 1-6).
3. **Controls.** Navigate by other elements, such as form fields, buttons, or menus.
4. **Links.** Navigate by different kinds of links, such as visited, unvisited, or active.

5. **Words.** Navigate by individual words.
6. **Landmarks.** Navigate by ARIA landmarks, such as "main" or "navigation."

Compare proposed navigation modes to the navigation modes enumerated in Section 2.2 related to TalkBack. Navigation modes are very similar; however, in proposed implementation Special content category was omitted and is accessible using the Default mode.

Using the Default navigation mode, the whole page content is explored in logical order as sighted user percepts it. The navigation by headings filters headings of all levels and can be used to navigate quickly in extensive and structured texts (e.g., web encyclopedia or newspaper articles) to the area of interest. The navigation by controls iterates through control elements like form entries, a different type of buttons, checkboxes, etc. This kind of navigation allows quick localization of form entries (e.g., search entry). The landmark mode searches for ARIA landmarks in the page. ARIA landmarks provide a powerful way to identify the structure of a web page [32]. Due to the complexity of today's web content, all perceivable content should be present inside a semantically meaningful landmark so that the user does not miss content. The landmarks are meant to support keyboard navigation in the structure of a web page for screen readers. The roles are assigned by developers to landmarks based on the type of content in the area and allow nesting of roles for parent/child relationships of the information that is present. Some of these roles are

1. **Banner** identifies site-oriented content at the beginning of each page (e.g., logo, search tool or site sponsor). It responds to the header.
2. **Complementary** supports a section of the document designed to be complementary to the main content at a similar level but remains meaningful when separated.
3. **Content info** identifies the common information at the bottom of each page. It responds to the footer.
4. **Form** identifies a region that contains a collection of items that creates a from.
5. **Main** identifies primary or main content of the page.
6. **Navigation** identifies groups of links intended to be used for navigation.
7. **Region** represents the perceivable section with relevant to specific content and important enough that users will likely want to be able to navigate to the region easily.
8. **Search** identifies a collection of items that, as a whole, creates search functionality.

The effective navigation in a web page is achieved by combining different navigation modes. E.g., user can use headings to navigate in an article to the paragraph of interest and then switch to word navigation and read section by words.

The full list of browser functions and their assignment to individual buttons is available in Table 3.1 and Figure 3.3.

Key	**Action**
Volume up	Volume up or voice input on longpress
Volume down	Volume down or voice input on longpress
Menu	Open a web browser menu
Back	Go back in page history
D-pad center	Click current element
D-pad left	Previous page element
D-pad right	Next page element
D-pad up	Open global states of the *Core system*
D-pad down	Speak (or spell) current element text
Key 1	Default navigation option
Key 2	Headings navigation option
Key 3	Controls navigation option
Key 4	Links navigation option
Key 5	Words navigation option
Key 6	Landmarks navigation option
Key 7	Go to page top
Key 8	Find in page
Key 9	Go to page bottom
Key 0	Speak current page title
Key *	Lock or unlock screen
Key #	Switch keyboard type, if active

Table 3.1: Web browser controls.

Technically the navigation in a web page is implemented by navigation in page tree representation using filtering and extraction algorithms for generating the correct output information. The technical aspect of navigation is described in more detail in Section 4.3.3.

3.4.2 Voice input

Even though the voice input method incorporating speech recognition algorithm is not considered to be the primary source of input, it is integrated as a complementary input channel. To achieve consistency with the *Core system*, voice commands will not be used for navigation-related operations because are used for opening applications or dictating text to the *Core system* keyboard. See Figure 3.4 for an example of text dictation result inserted into the input entry of a search engine.

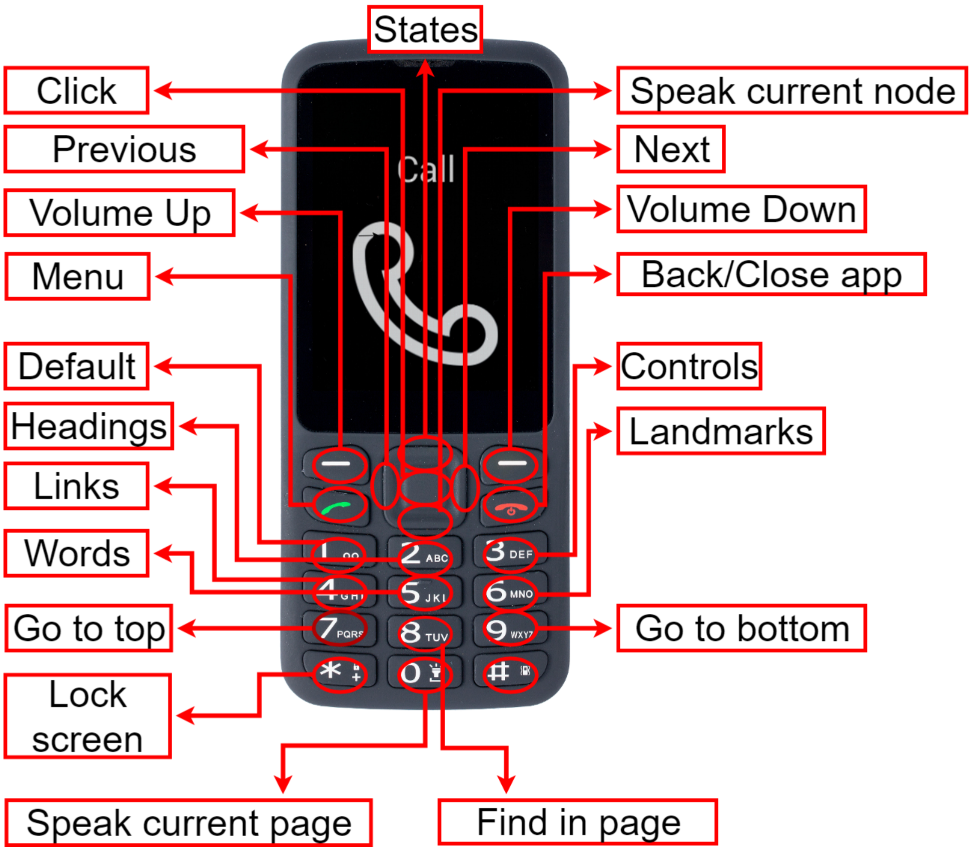


Figure 3.3: Target device tactile buttons with assigned functions.

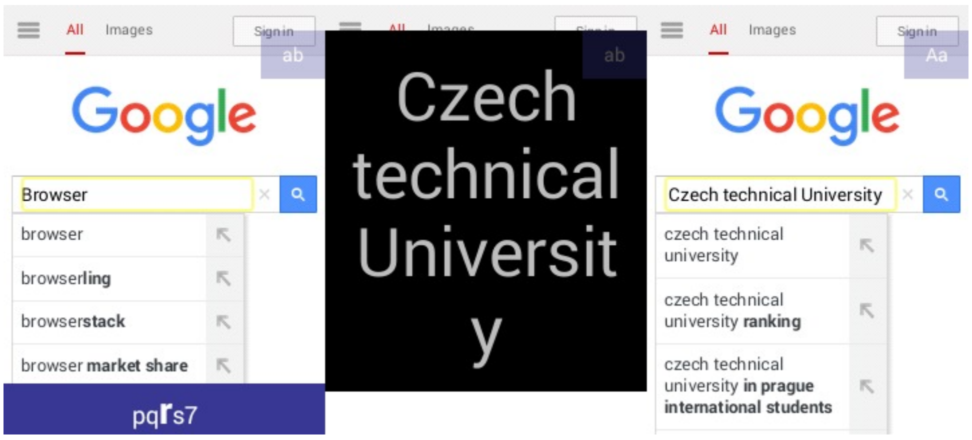

All
Images
Sign in
ab
Google
Browser
browser
browserling
browserstack
browser market share
pqrs7
Czech technical University
Aa
Czech technical University
czech technical university
czech technical university ranking
czech technical university in prague international students

Figure 3.4: UI screens from Browser documents the usage of different input methods (i.e. keyboard and speech dictation). From left to right: a) The classic keyboard input method with integrated TTS, b) Text recognized by speech recognition engine, c) Text inserted to the input entry in the page after speech recognition.

3.5 Output interface design

The proposed user interface of a mobile web browser for VIP consists of three output channels

1. visual (rendered web page with highlighted virtual cursor and UI components of the *Core system* used for dialogs and menu),
2. auditory (text-to-speech and earcons),
3. vibration (produced by integrated vibramotor).

The visual output is a standard output method for all web browsers and is produced by selected Gecko engine responsible for converting HTML document to its visual form. Auditory feedback (in form of spoken words synthetised using TTS engine or brief distinctive sounds called earcons) is incorporated as primary output as we assume that users are not able to perceive visual output sufficiently or at all. Vibrations are integrated as a reasonable alternative to the auditory feedback, because some blind users spontaneously suggest using vibrations instead of spoken word for some announcements.

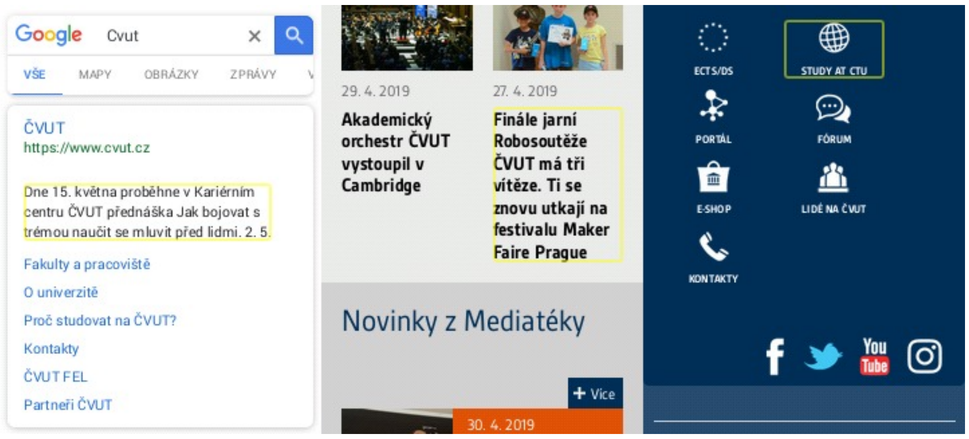
3.5.1 Visual

The modern web pages are visually impressive, demanding and thanks to the JavaScript often very dynamical. Even though, the visual output might be useless for blind users, for some users who have intermediate vision loss it could still be beneficial especially for mentioned visual summarization of a page content and navigation related operations. The only difference between the regular browser and browser for the blind regarding visual output is that page element focused by the virtual cursor (a.k.a. accessibility focus) should be highlighted with bounding rectangle (often green or yellow). See Figure 2.3 for an example of highlighting rectangles in TalkBack service or Figure 3.5 for an example of highlighting rectangles rendered by chosen Gecko engine.

3.5.2 Text-To-Speech

Text-To-Speech (TTS) engine refers to the system capable of converting the text in some language into the human speech equivalent. The quality of the synthesized speech can be judged by its similarity to human voice and clarity. TTS technology is the most often used output method by accessibility tools intended for VIP.

TTS is considered to be the primary feedback because we do not assume that mobile device will be used with external text to Braille devices besides desktop PC. When using TTS as a primary output, we must consider the possibility that the user can be quickly overwhelmed with never-ending audio


Cvut

ČVUT
https://www.cvut.cz

Dne 15. května proběhne v Kariérním centru ČVUT přednáška Jak bojovat s trémou naučit se mluvit před lidmi. 2. 5.

Fakulty a pracoviště
O univerzitě
Proč studovat na ČVUT?
Kontakty
ČVUT FEL
Partneři ČVUT

29. 4. 2019
Akademický orchestr ČVUT vystoupil v Cambridge

27. 4. 2019
Finále jarní Robosoutěže ČVUT má tři vítěze. Ti se znovu utkají na festivalu Maker Faire Prague

Novinky z Mediatéky

30. 4. 2019

Figure 3.5: Example of a virtual cursor bounding rectangle.

output. It is the reason why we give user opportunity specify in settings, which events should be or should not be announced by speech output.

Generation of proper descriptive TTS output for selected page element is essential mainly due to the often repetitions, and will be implemented as follows: the type of element and its text will be announced by default. If additional information (e.g., hint or content description) is present, it will be announced as well, but there has to be an option to configure if additional information should be spoken.

Browser implementation will use Google TTS [22] as TTS engine. Google developed Google TTS engine for the Android OS to be used by Google applications (e.g., Google Translate), third-party applications, or accessibility services (e.g., TalkBack). The engine supports approximately 50 languages a when using default voice with downloaded language packages it works completely offline.

3.5.3 Earcons

Icons are visual symbols that represent information about an application, object, or function. Earcons are characteristic, brief, and distinctive sounds that represent a specific item or event. Earcons are not relevant only for accessibility tools related to VI but are present in most of the operating systems (e.g., system sound in Windows or Linux). However, earcons gain importance in the context of VI as one of the primary output methods.

The proposed web browser implementation provides settings, which allows configuration of events and elements of a web page, which should be announced with earcons instead of spoken words. Earcons are one of the possible ways how to reduce the probability of overwhelming with auditory speech output.

The implementation allows a user to use earcons for these types of events and items:

1. link,
2. page load started or stopped,
3. first or last element of a selected type,
4. download completed,
5. keyboard opened or closed.

3.5.4 Vibrations

Vibrations are considered to be suitable complementary output method for the auditory channel (i.e., TTS and earcons) and partially help to reduce the risk of overwhelming user with auditory (most often speech) output. Most of the Android smartphones (if not all) are equipped with a vibrator, which is a mechanical device able to generate vibrations.

3.6 Menu design

Unlike TalkBack and some other accessibility tools currently available, this browser implementation will not distinguish between local and global context menu, and both will be merged into one dynamic menu.

The browser menu will have a tree structure (as any other *Core system* activity), which can be seen in Figure 3.6. The browser menu meets the requirements set in Section 3.3.

3.7 Conclusion

This chapter documented the design of a web browser UI. A short insight into the World Wide Web statistical properties valuable for the later design of in-page navigation principles was given in Section 3.1. Web browser engine candidates were examined in Section 3.2. Gecko engine developed by Mozilla Foundation was chosen for implementation and integration into the *Core system*. Requirements on the proposed browser application were formulated in Section 3.3. Concerning defined requirements, the UI design was exactly described in Sections 3.4 and 3.5 dedicated to input and output parts of UI. The design of input UI described the usage of tactile hardware buttons as the primary input source and principles of navigation in a web page in Subsection 3.4.1. The usage of speech recognition based voice input, which is incorporated for launching the application and as an alternative for keyboard inputs while filling forms, is documented in Subsecton 3.4.2. The key parts of output UI were enumerated in Section 3.5 and discussed in more detail in Subsections 3.5.1, 3.5.2, 3.5.4.

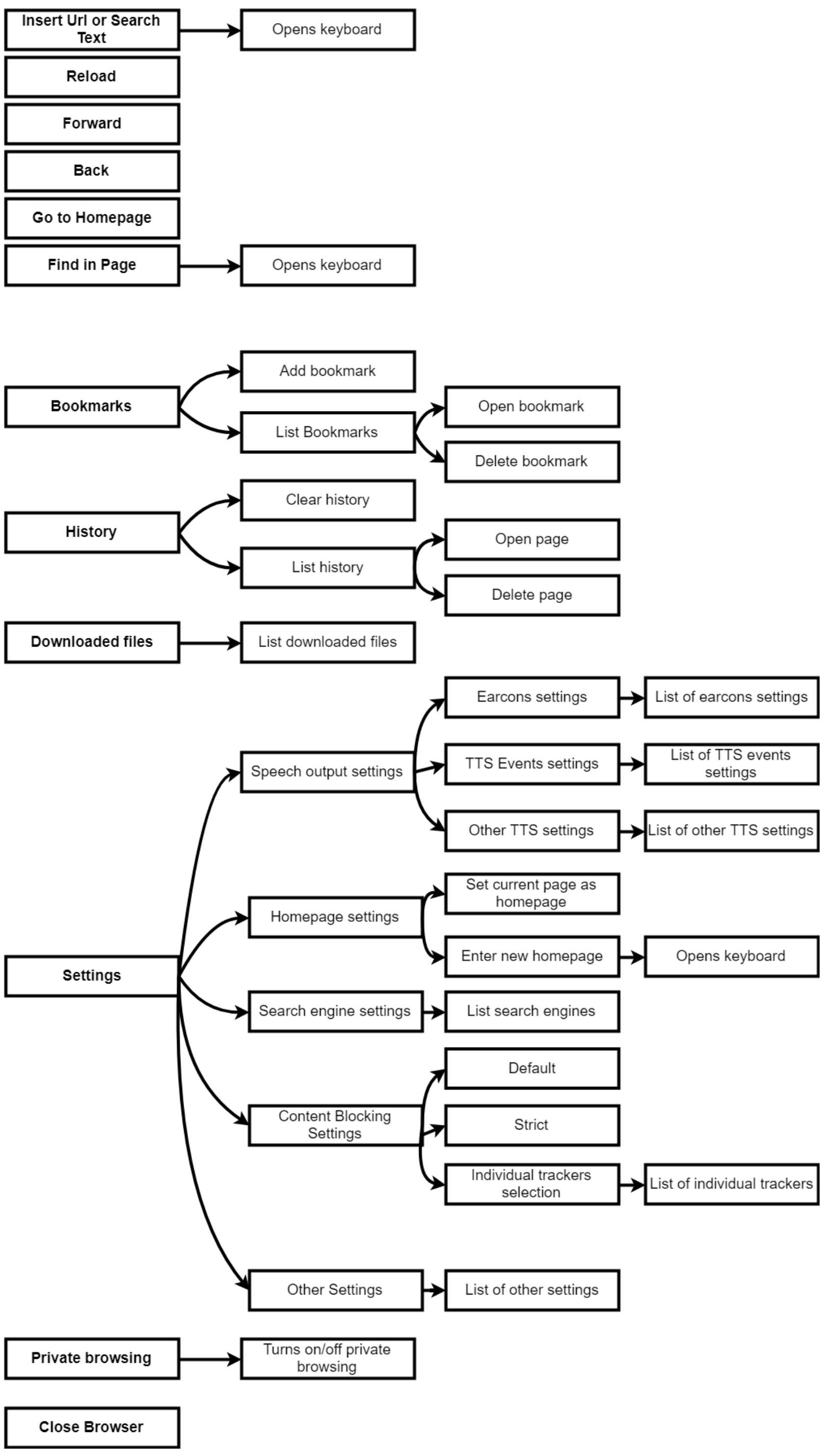


Figure 3.6: Tree structure of browser menu.

Chapter 4

Web browser implementation

4.1 Introduction

This chapter deals with the implementation and integration of a mobile web browser for VIP. The theoretical description of a proposed browser architecture is given in Section 4.2. The implementation of the UI proposed in Chapter 3 is described in Section 4.3. The integration into the existing *Core System* explained in Section 2.4 is documented in Section 4.4.

The implemented mobile web browser is an Android application and is implemented in the Android Framework environment. Our browser implementation consists of several Activities and Accessibility Service, which retrieves the page structure and navigates in that structure using a tree search algorithm described in Subsection 4.3.3.

4.2 Architecture

This Section describes the top-level architecture of the proposed mobile web browser for the VIP, which consists from the following modules (similarly like [16]):

1. **Input module** provides different input methods (hardware keyboard buttons, gestures, or voice commands in our case).
2. **Content extraction and filtering module** provides tools for extraction and filtration of a relevant web content.
3. **Representation and navigation module** provides suitable structures for a web page representation and methods for a navigation related operations.
4. **Output module** provides different output methods (such as Text-To-Speech (TTS), non speech auditory output or vibrations in our case).

Block diagram of a top-level architecture of a proposed mobile web browser can be seen in Figure 4.1. Note that Extraction and filtering module and Representation and navigation module are separated from the architectonical point of view, but are closely tied in order to exhibit the desired behavior. Also note that earcons, which could be played as regular sounds are also sent to the TTS engine, which is more convenient because these sounds are enqueued into the same buffer as spoken words, which makes synchronization of audio channels easier. Representation and navigation module and Content extraction and filtering module are described in following subsections in more detail.

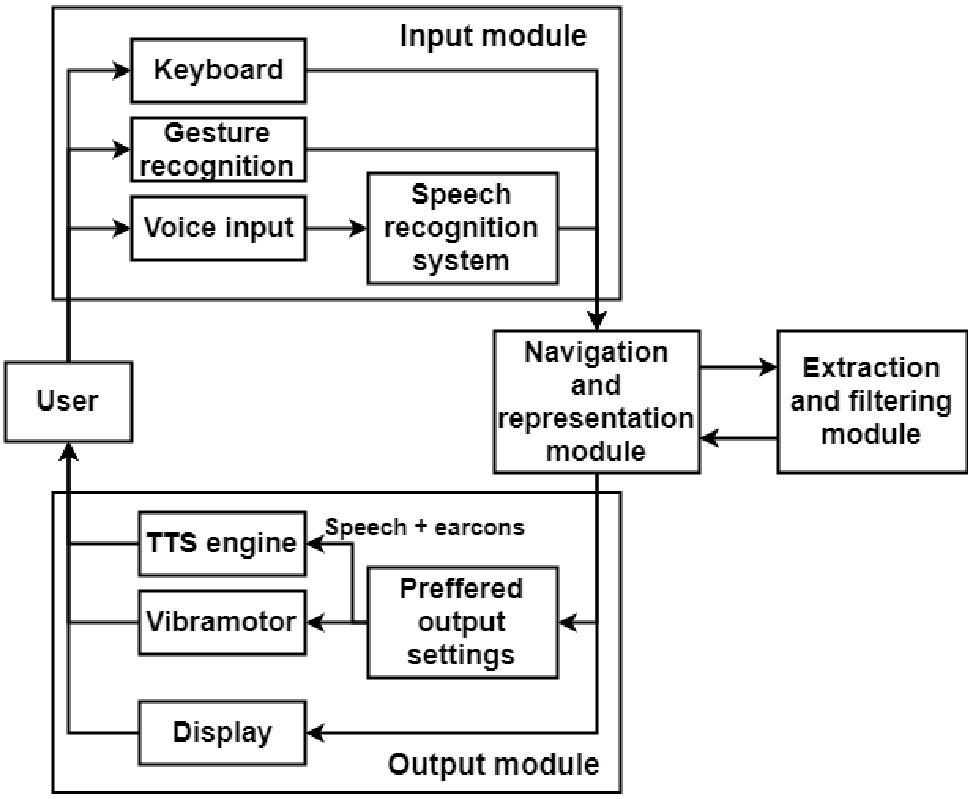
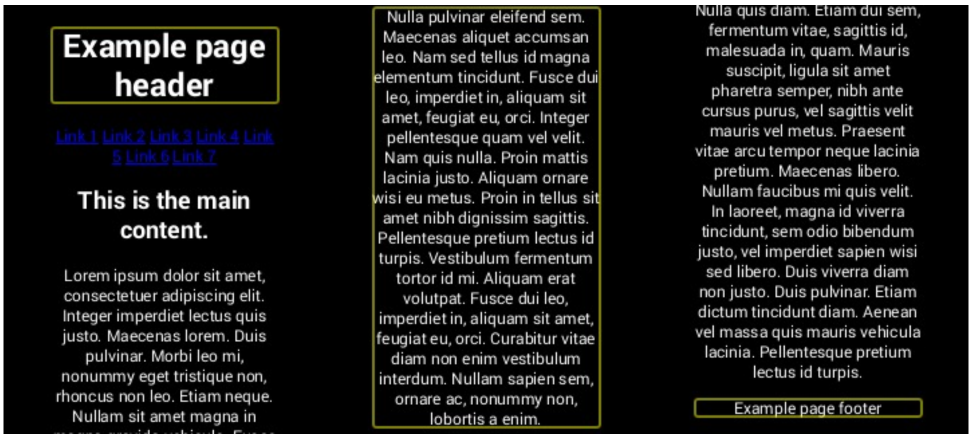

Input module
Keyboard
Gesture recognition
Voice input
Speech recognition system
User
Navigation and representation module
Extraction and filtering module
TTS engine
Speech + earcons
Vibramotor
Preffered output settings
Display
Output module

Figure 4.1: Block Diagram of the architecture of the proposed mobile web browser for the blind.

4.2.1 Representation and navigation module

Page content representation and navigation is a crucial part of the browser function. Web pages are documents written in HTML combining text information with meaningful mark-ups of the text. The tree structure of representation of a web page content is derived from the naturally tree-structured HTML. Compare Figure 4.2 capturing an example page with the Figure 4.3 with page representation used in the Representation and navigation module.

Navigation in the page is technically navigation in the page representation structure, which is a tree as stated before. Navigation has to be implemented using some graph search algorithm. In the first phases of development Depth-first search (DFS) algorithm was used for that purpose, because it reflects the natural ordering of page elements (against Breadth-first search (BFS)). Later the DFS graph algorithm was found inconvenient in terms of space complexity, which is for DFS $O(|V|)$, where $|V|$ is the number of vertices in the graph. Usage of DFS for navigation in lengthy pages would lead to memory wasting on storing of currently useless objects representing the page elements in closed and open lists. It is the reason for the design and implementation of a tree search algorithm, which visits the nodes of the graph in the same order as DFS, but exhibits lower worst-case space complexity. The proposed tree search algorithm is described in Subsection 4.3.3.



Example page header

Link 1 Link 2 Link 3 Link 4 Link 5 Link 6 Link 7

This is the main content.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer imperdiet lectus quis justo. Maecenas lorem. Duis pulvinar. Morbi leo mi, nonummy eget tristique non, rhoncus non leo. Etiam neque. Nullam sit amet magna in

Nulla pulvinar eleifend sem. Maecenas aliquet accumsan leo. Nam sed tellus id magna elementum tincidunt. Fusce dui leo, imperdiet in, aliquam sit amet, feugiat eu, orci. Integer pellentesque quam vel velit. Nam quis nulla. Proin mattis lacinia justo. Aliquam ornare wisi eu metus. Proin in tellus sit amet nibh dignissim sagittis. Pellentesque pretium lectus id turpis. Vestibulum fermentum tortor id mi. Aliquam erat volutpat. Fusce dui leo, imperdiet in, aliquam sit amet, feugiat eu, orci. Curabitur vitae diam non enim vestibulum interdum. Nullam sapien sem, ornare ac, nonummy non, lobortis a enim.

Nulla quis diam. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Mauris suscipit, ligula sit amet pharetra semper, nibh ante cursus purus, vel sagittis velit mauris vel metus. Praesent vitae arcu tempor neque lacinia pretium. Maecenas libero. Nullam faucibus mi quis velit. In laoreet, magna id viverra tincidunt, sem odio bibendum justo, vel imperdiet sapien wisi sed libero. Duis viverra diam non justo. Duis pulvinar. Etiam dictum tincidunt diam. Aenean vel massa quis mauris vehicula lacinia. Pellentesque pretium lectus id turpis.

Example page footer

Figure 4.2: Example page created to illustrate the mapping between the HTML structure of document and a internal page representation used by the Representation and navigation (captured in Figure 4.3).

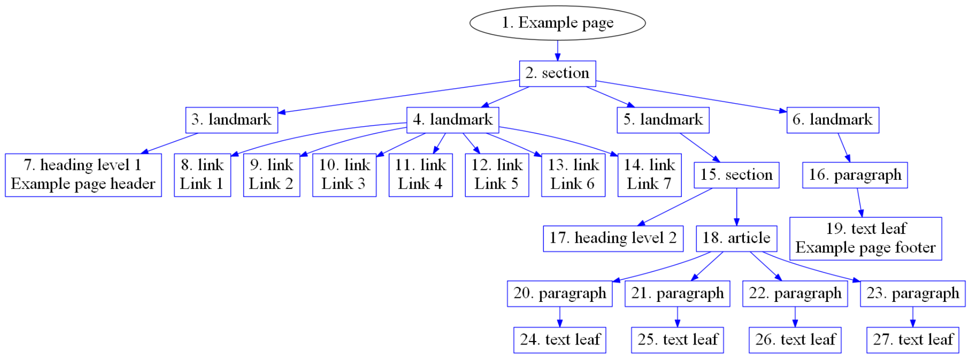


Figure 4.3: Representation of the page captured in Figure 4.2. The long texts of paragraphs were ommitted becauseof clarity.

4.2.2 Content extraction and filtration module

Extraction and filtering of relevant content is another crucial part of a web browser functionality. Extraction and filtering feature was a part of all examined desktop browser implementations in Section 2.1.

Content extraction and filtering module is responsible for filtering the proper content. E.g., if a user navigates by headings and wishes to go to the next one, we want to apply some filter accepting only headings elements. This module mainly consists of various filters, which can be arbitrary joined using logical operators, and is used by Representation and navigation module to test if currently visited node is a target node.

4.3 Implementation

Implementation of a proposed browser mainly relies on selected Gecko engine and Android Accessibility framework and was realized through 2 key parts: Main Activity and Accessibility Service. Main Activity and Accessibility Service communicates in order to mutually synchronize using local broadcasts. Main Activity of a browser is described in Subsection 4.3.1, the Accessibility Service enabling the accessibility features of Android-based smart-phones is subsequently examined in Subsection 4.3.2. The designed and implemented tree search algorithm incorporated for navigation in page representation structure and its properties are described in detail in Subsection 4.3.3. Some other important parts of implemented UI are discussed in the following sections as well.

4.3.1 Main activity

Activity is one of the fundamental concepts in the Android framework and represents a single, focused thing that the user can do. Main Activity is regular Android Activity [19] and is in the heart of the proposed mobile web browser. The layout of Main Activity contains full screen GeckoView component responsible for page rendering and progress bar visible while the page is loading. In this activity the most of the surfing happens. As user iterates over the page elements possibly using some navigation filter, virtual cursor highlighted with yellow bounding rectangle moves across the screen and currently selected content or happening event is announced by TTS, earcon or vibration.

4.3.2 Accessibility service

Android Accessibility framework is a part of the Android operating system and is built around AccessibilityService class. Accessibility Service is special type of a service, which should only be used to assist users with disabilities

[18]. They run in the background like any other services, but receives callbacks by the system when AccessibilityEvent are fired, they can also retrieve the content of the currently active window or intercept key events or gestures before being dispatch to activities. Due to privileged access to events and window content, Accessibility Service represents a threat to security (e.g. could be used as keyboard logger). That is reason why Accessibility Service must be explicitly enabled by the user in Accessibility Settings. Moreover, the developers which use Accessibility Services must justify benefits of that service for people with disabilities, otherwise, their application is not allowed to be offered at Google Play Store.

Unfortunately, this browser implementation relies on Accessibility Service, but the target device does not have any Accessibility Settings so that the enabling of the Service by the user is not possible. Moreover, it is not desired to force user of the *Core system* to enable and disable service in our case. We wish to enable or disable Accessibility Service programmatically as browser activity starts or stops. This behavior is achieved by direct modifications in the system security database, which requires system application permission.

The Accessibility Service can be used to retrieve the content of the current window in the form of a hierarchical structure of AccessibilityNodeInfo objects. Tree structure of AccessibilityNodeInfo object which corresponds to virtual views rendered by Gecko engine is used as page tree representation. AccessibilityNodeInfo object carries the information about node's text, hint, in screen position, children, description, etc. AccessibilityNodeInfo object also allows performing of specific actions on the view they represent, some examples of these actions are a click, long-click, scroll, cut, copy, paste, etc.

4.3.3 Tree search algorithm

As stated before, the page representation is a tree of AccessibilityNodeInfo objects, which represents information about the hierarchy of the page elements. Class AccessibilityNodeInfo has instance methods for getting parent node and children nodes. As mentioned before, the DFS algorithm was not suitable in terms of space complexity. Keeping a possibly very long closed list of objects is inconvenient and causes some troubles with AccessibilityNodeInfo lifecycle because objects must be refreshed or recycled to avoid multiple instances for a single element. The worst-case space complexity of DFS is equal to the number of nodes of the graph, while worst-case space complexity of proposed tree search algorithm is $O(|D_m|)$, where $|D_m|$ is a depth of the tree. This property is caused by the fact that algorithm stores only path to the current node of the graph. Algorithm visits nodes in the same order as DFS, which is desired because it reflects the natural ordering of page elements. The inputs of proposed algorithm are root of the tree r, currently selected node of the tree c, list of nodes, which form a path from r to c denoted as l, predicate filter f mapping each node to $\{0, 1\}$ space and bool b specifying if search should be performed in backward direction. Based on provided input, the algorithm might find another node in the page tree representation and return following outputs: the result of navigation (one of FOUND, FIRST, LAST, NONE), newly selected node (if result FOUND), the path to the newly selected node (if result FOUND). See Algorithm 1 for the exact definition of the algorithm. The backward search (i.e., user wishes navigate to previous page item) is performed by reversing the order of children of all nodes of a graph and then applying the same algorithm as for forward search.

For deeper insight into how algorithm works lets imagine the following use-case. User opens the page with structure captured in Figure 4.4 and wishes to navigate by heading. After the page is loaded, the algorithm is initialized with following values $r = 1$, $c = r = 1$ and $l = \emptyset$, $f = headings$ and $b = false$. Then algorithm visits the tree nodes in the same order as DFS until it reaches the node satisfying the filter predicate f. The states of the algorithm during individual iterations are documented in Table 4.1. The result of the algorithm corresponding to the last iteration is denoted in Figure 4.4 as well.

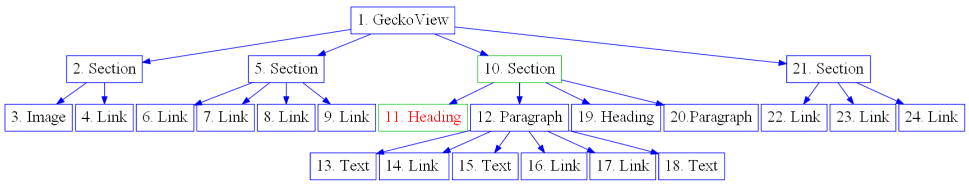


Figure 4.4: An example of the tree that represents the structure of a simple page with logo, menu formed by links and some text with headings. The node with red text denotes the selected node c and green bounding rectangles the path from root r to the selected node c after the algorithm finished.

Iteration i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
List of nodes l	-	2	2,3	2,3	2,4	2,4	2	5	5,6	5,6	5,7	5,7	5,8	5,8	5,9	5,9	5	10	10,11
Current node c	1	2	3	2	4	2	1	5	6	5	7	5	8	5	9	5	1	10	11

Table 4.1: States of the algorithm during the search. Note that the transition betweeen the 2 children of the same parent is done in 2 iterations. In first iteration, the current node is set to the parent, but the path remains unchanged. In second iteration, the unchanged path is used to determine previously visited child and next child to visit.

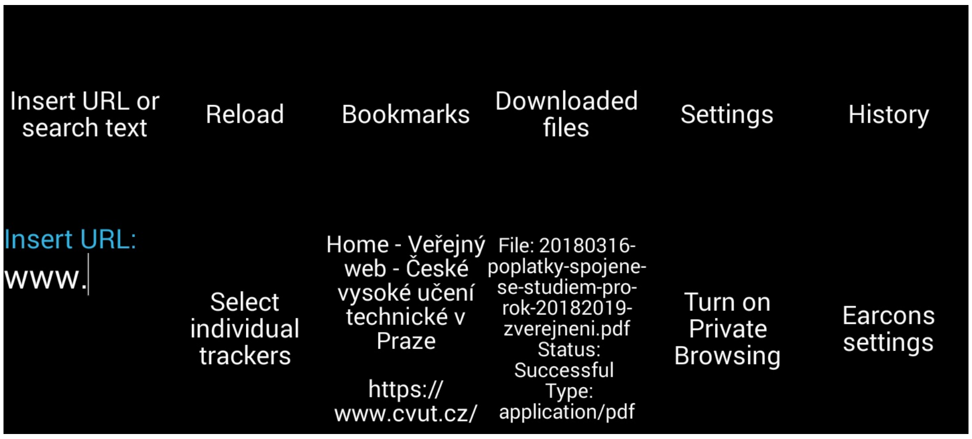
4.3.4 Menu

The menu of the Browser was implemented based on the tree structure designed in Subsection 3.6 using the standard UI components provided by the *Core system*. Figure 4.5 captures examples of UI screens from the implemented menu. Another Figure 4.6 shows the 'Media autoplay' option in 'Other settings' in menu.

```
Input   : Root of the tree $r$
Input   : Current node $c$
Input   : List of nodes $l$ from $r$ to $c$ (path in a tree)
Input   : Predicate filter $f$
Input   : Bool backward search $b$
Output  : Status $s$ - one of FOUND, FIRST, LAST, NONE
Output  : Found node $c$ if FOUND
Output  : List of node $l$ from $r$ to new node $c$ if FOUND

while true do
    if c.hasChildren() then
        $t \leftarrow -1$
        for $i \leftarrow 0$ to c.childCount()-1 do
            if c.child(i) == l.lastNode() then
                $t \leftarrow i$
            end
        end
        if $t == -1$ then
            $c \leftarrow c$.child($b$ ? $c$.childCount() - 1 : 0)
            $l$.add($c$)
            if f.accept(c) then
                return FOUND, $c$, $l$
            end
        else if (t < c.childCount() - 1 and not b) or (t > 0 and b)
          then
            $l$.removeLastElement()
            $c \leftarrow c$.child($b$ ? $t$-1 : $t$+1)
            $l$.add($c$)
            if f.accept(c) then
                return FOUND, $c$, $l$
            end
        else
            if l.notEmpty() then
                $l$.removeLast()
            end
            if $c == r$ then
                if f.accept(c) then
                    return $b$ ? FIRST : LAST
                end
            else
                return NONE
            end
            $c \leftarrow c$.parent()
    else
        $c \leftarrow c$.parent()
    end
end
```

Algorithm 1: Tree search algorithm used for navigation in a page representation.


Insert URL or search text
Reload
Bookmarks
Downloaded files
Settings
History

Insert URL:
www.

Select individual trackers

Home - Veřejný web - České vysoké učení technické v Praze
https://www.cvut.cz/

File: 20180316-poplatky-spojene-se-studiem-pro-rok-20182019-zverejneni.pdf
Status: Successful
Type: application/pdf

Turn on Private Browsing

Earcons settings

Figure 4.5: Examples of some menu screens implemented using the *Core system* UI components.

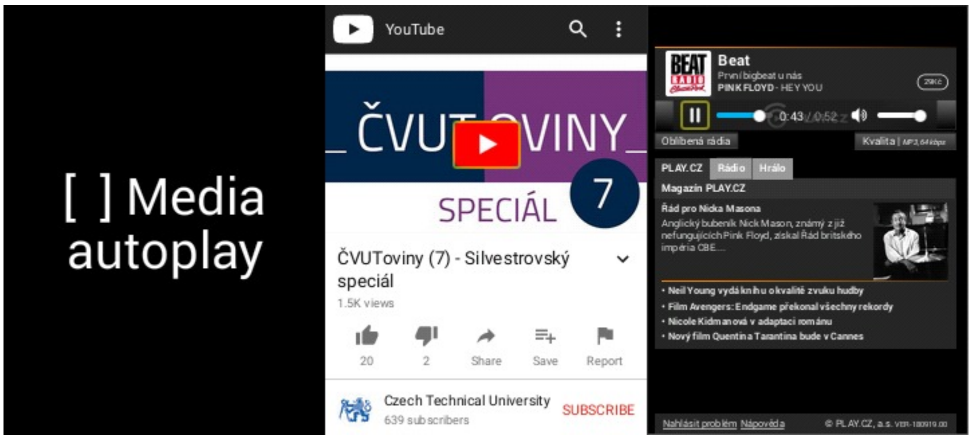

[ ] Media autoplay
YouTube
ČVUToviny
SPECIÁL
7
ČVUToviny (7) - Silvestrovský speciál
1.5K views
20
2
Share
Save
Report
Czech Technical University
639 subscribers
SUBSCRIBE
Beat
PINK FLOYD - HEY YOU
0:43
Oblíbená rádia
PLAY.CZ
Rádio
Hrálo
Magazín PLAY.CZ
Řád pro Nicka Masona
• Neil Young vydá knihu o kvalitě zvuku hudby
• Film Avengers: Endgame překonal všechny rekordy
• Nicole Kidmanová v adaptaci románu
• Nový film Quentina Tarantina bude v Cannes
Nahlásit problém
Nápověda
© PLAY.CZ, a.s.

Figure 4.6: Examples of UI screens related to the media playing shows that even the medial content can be accessed quite easily: a) Media autoplay settings option, if selected audio and video content present in pages starts automatically, b) Playing video at youtube.com manually without autoplay option on using 'Play' button c) Playing internet radio.

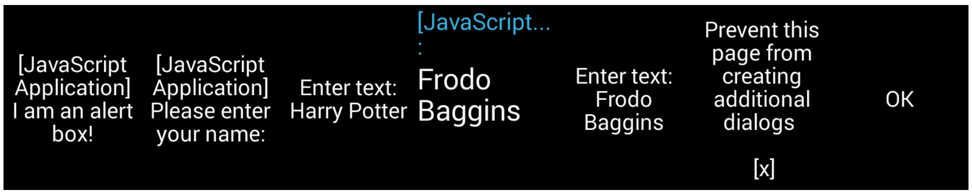
4.3.5 Prompt and dialogs

JavaScript prompts resulting in different types of a dialog are delegated by Gecko engine to the application for custom handling. Gecko recognizes and delegates these types of prompts to the client application

1. **Alert prompt** contains title, message and optional checkbox. (See Figure 4.7).
2. **Button prompt** is a child of Alert prompt, and further may contain three types of buttons: positive, neutral and negative.
3. **Text prompt** is a child of Alert prompt, and further contains text input field. (See Figure 4.7)

4. **Auth prompt** is a child of Alert prompt, and further contains text input fields for username and password.
5. **Choice prompt** contains array of choice. There are more types of this prompt: menu, single and multiple selections. See Figure 4.8 for an example choice prompt.
6. **Color prompt** allows a user to select the desired color using color picker dialog. (See Figure 4.9).
7. **DateTime prompt** allows a user to pick both time or date. There are variations of this dialog, some of them prompts only for a time, other for a date or a day of the week, etc.
8. **File prompt** allows a user to select a file using file picker for upload. This prompt is disabled for now because the *Core system* do not provide standard way of managing files.
9. **Media prompt** allows a user to grant or reject permission for accessing the video or audio source of the device (e.g., front or back camera or microphone). This prompt is disabled because it is not desired.
10. **Permission prompt** allows a user to grant or reject permissions prompted by a web page. Example of such a prompt could be a web page asking to share the location.

All these types of prompt were implemented using the *Core System* UI components, while standard browser implementation would use standard Android dialogs instead.


[JavaScript Application] I am an alert box!

[JavaScript Application] Please enter your name:

Enter text: Harry Potter

[JavaScript... : Frodo Baggins

Enter text: Frodo Baggins

Prevent this page from creating additional dialogs [x]

OK

Figure 4.7: Examples of UI screens of implemented JavaScript prompt dialogs: a) Simple alert dialog with message b,c,d,e,f) Text dialog with optional checkbox a predefined text.

4.3.6 Auto-scroll

Unlike regular mobile web browsers, which perform scrolling automatically in reaction to gestures or key presses, this specific browser implementation based on the filtered sequential movement of the virtual cursor has to handle scrolling operations in such a way, that virtual cursor is always on screen.

The auto-scroll feature was implemented with a custom timer using Handler class, which checks every 3 seconds if the virtual cursor is on screen, if not it scrolls.

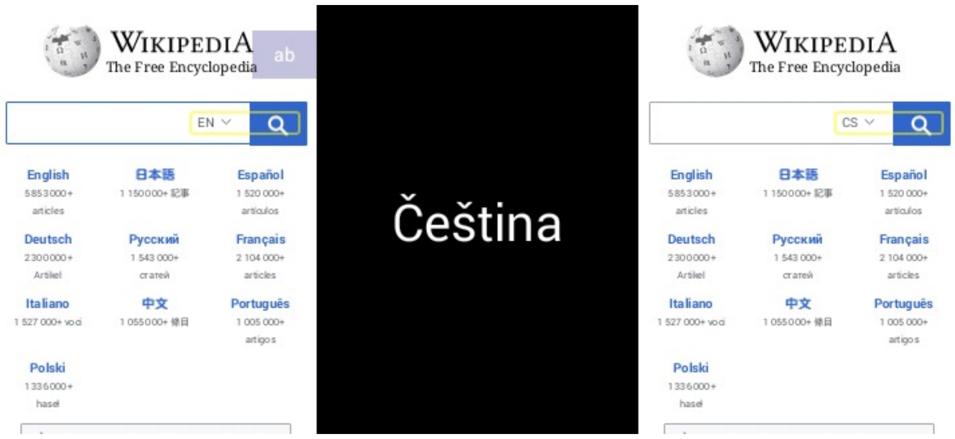
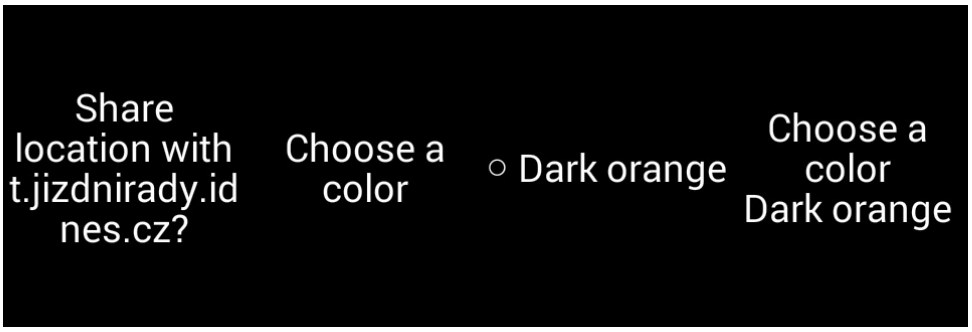


Figure 4.8: UI screens document an example of Choice prompt usage in following use-case: a) User clicks on combobox, which prompts for the language of search, b) Choice prompt is implemented using the *Core system* UI components, c) After user selected Czech language.


Share location with t.jizdnirady.idnes.cz?

Choose a color

○ Dark orange

Choose a color Dark orange

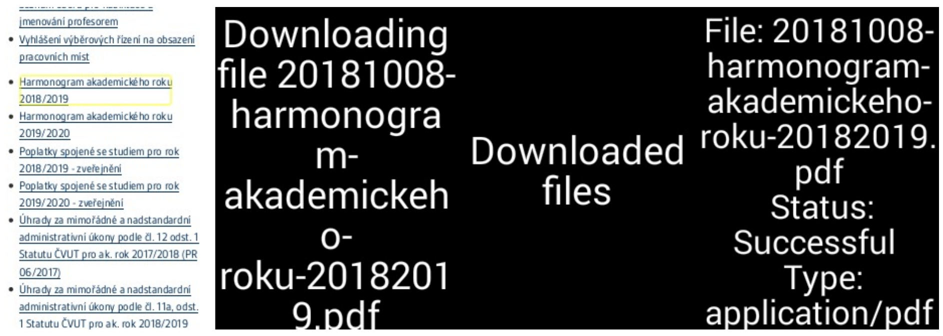
Figure 4.9: Examples of UI screens of implemented JavaScript prompt dialogs: a) Button prompt asking for sharing location, b,c,d) process of choosing color in Color prompt.

4.3.7 File downloads

Downloading files is a standard browser functionality. Downloaded files can be listed in the browser menu (see Figure 4.10 for examples of UI screens related to file downloading). For now, the downloaded files can be only listed, but not opened, because the *Core System* does not supports general mechanism for handling arbitrary types of files.

4.3.8 Finder

As shown in Figure 3.2 finder is the second most often used navigation method. The finder option is accessible through the menu or can be accessed using shortcut assigned to key 8. The finder was not implemented using Gecko incorporated finder but is performed using a proposed tree search algorithm with predicate f corresponding to the searched term so that the matches



- jmenování profesorem
- Vyhlášeni výběrových řizeni na obsazeni pracovních míst
- Harmonogram akademického roku 2018/2019
- Harmonogram akademického roku 2019/2020
- Poplatky spojené se studiem pro rok 2018/2019 - zveřejnění
- Poplatky spojené se studiem pro rok 2019/2020 - zveřejnění
- Úhrady za mimořádné a nadstandardní administrativní úkony podle čl. 12 odst. 1 Statutu ČVUT pro ak. rok 2017/2018 (PR 06/2017)
- Úhrady za mimořádné a nadstandardní administrativní úkony podle čl. 11a, odst. 1 Statutu ČVUT pro ak. rok 2018/2019

Downloading file 20181008-harmonogram-akademickeho-roku-20182019.pdf

Downloaded files

File: 20181008-harmonogram-akademickeho-roku-20182019.pdf
Status: Successful
Type: application/pdf

Figure 4.10: UI screens give an example of downloading files.

with the search term can be explored similarly to other page elements, e.g., headings. See Figure 4.11 for UI screens documenting finder usage.

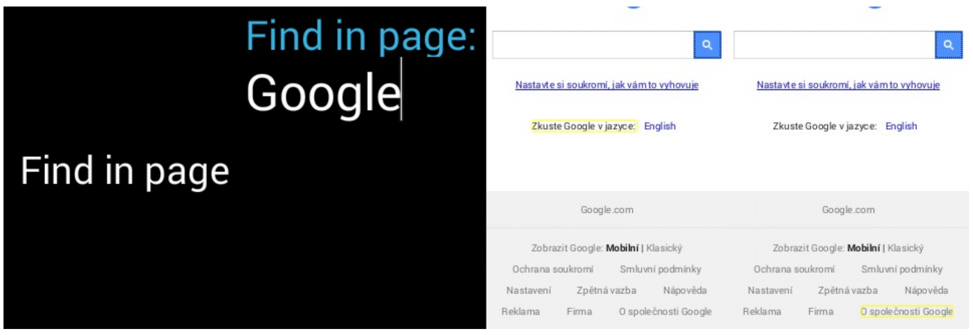


Figure 4.11: UI screens give an example of using browser finder. From left to right: a) Find in page item in menu, b) Keyboard, c,d) First and last match with the search term.

4.3.9 Error page

Error pages are implemented using an error page template, with a placeholder for error category, error type and title (to support localization). See Figure 4.12 for error page rendered for error unknown host often happening when connection unavailable and compare to Figure 4.13 capturing the tree structure of error page as stored in Representation and navigation module. After the error page is rendered, it can be explored like any other web page.

4.4 Integration into the Core system

This section documents the browser integration into the *Core system* described in Section 2.4. The browser can be launch from the *Core system* menu or by any of these voice command synonyms 'browser', 'web', or 'internet' (see Figure 4.14 for a UI screens documenting usage of voice command for opening the browser). Start of speech recognition is triggered by long pressing volume

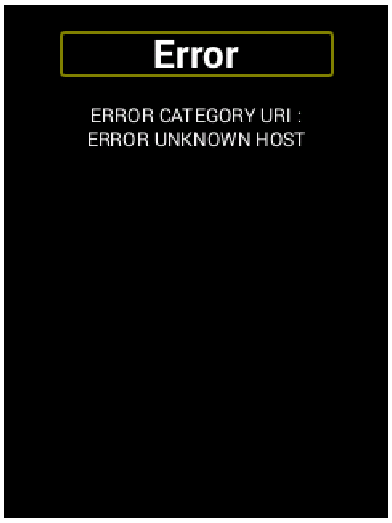


Figure 4.12: Example of Error page with virtual cursor (yellow rectangle) focusing heading 'Error'.

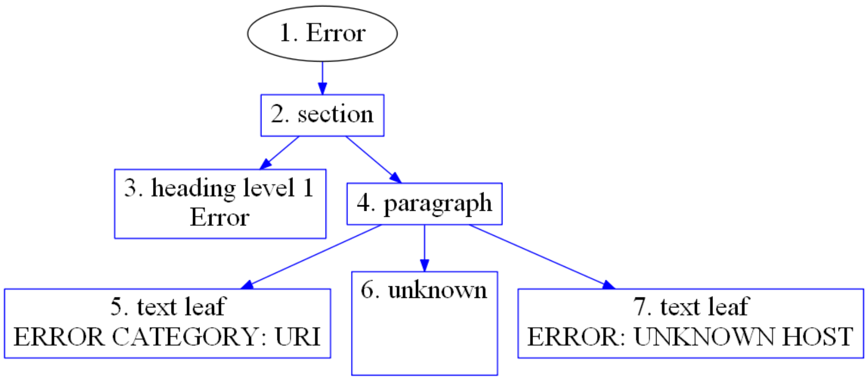


Figure 4.13: Example of page representation of Error page stored in Representation and navigation module. Element 6 corresponds to the line break tag so that the role is unknown, because is not important for accessibility.

up or down button. Figure 4.16 captures the *Core system* menu after the integration with highlighted entry point into the application. UI screens from the *Core system* menu and successfully integrated Browser are shown in Figure 4.15.

4.4.1 The Core System more technically

This subsection tries to describe the *Core system* more from a technical perspective that it was introduced in Section 2.4. The *Core system* consists of the following modules and technologies

1. the main java module, which is a standard Android 5.1 (API 22) project, is written in Java 8,

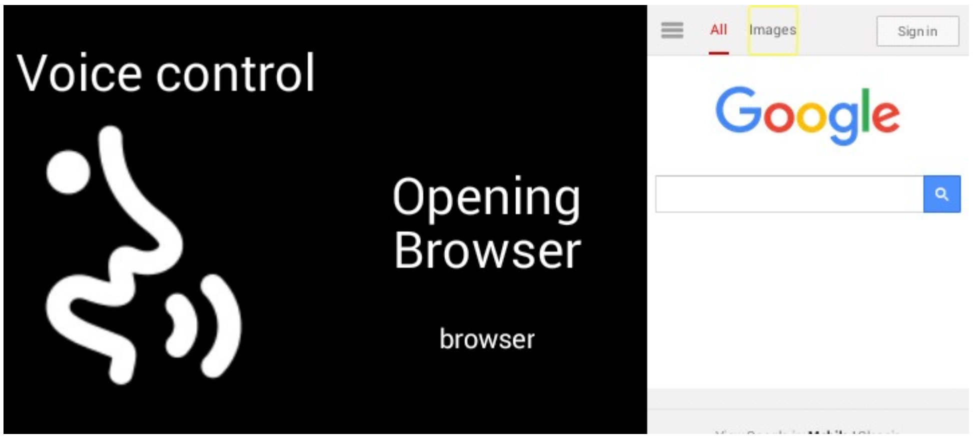


Figure 4.14: Example of voice control usage for opening the Browser. From left to right: a) An integrated voice control feature of the *Core system*, b) recognized command open the Browser, c) Browser opened with voice command.

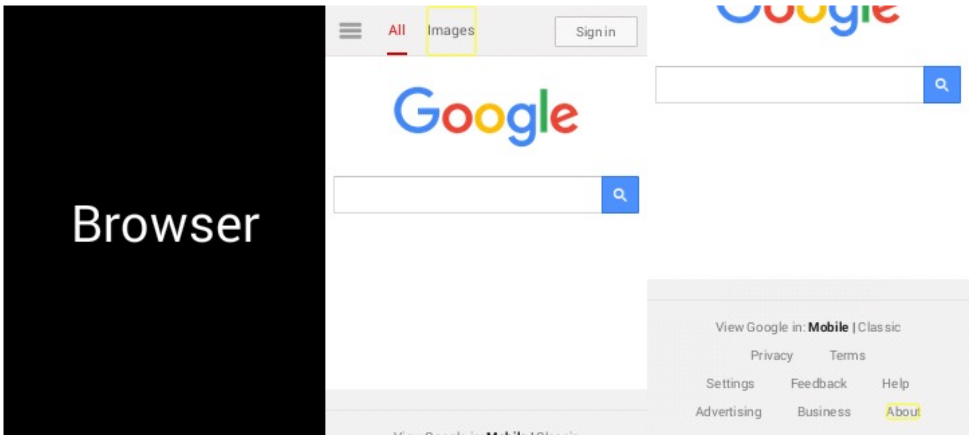


Figure 4.15: Screenshots of integrated web browser. From left: a) Browser launch icon in the *Core system* menu, b) Screenshot of www.google.com with selected link "Images", c) Screenshot of a www.google.com after page has been almost explored. Yellow rectangle (a.k.a accessibility focus) highlights currently selected page element.

2. third-party native libraries,
3. the Xposed Framework module, which enables the dynamic modifications of aspects of the target operating system,
4. some helper applications (Text-To-Speech) and hardware ROM customizations.

4.4.2 Target device

The *Core System* currently runs on target low-end hardware device with the following specification:

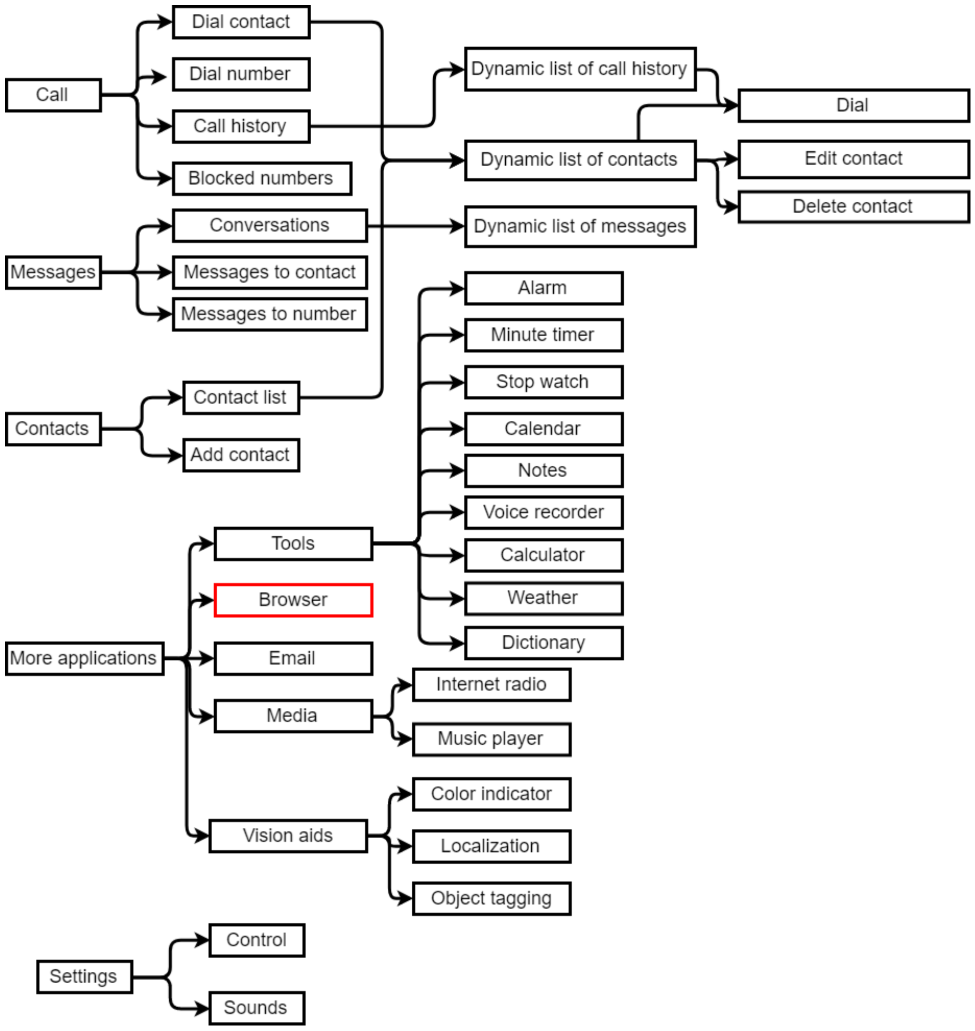


Figure 4.16: Part of the *Core system* menu after the integration with highlighed Browser.

1. Android 4.4 (API 19),
2. Dual core ARMv7 CPU 1.2GHz,
3. 512MB RAM,
4. hardware keyboard with 4 directional D-pad and SOS button,
5. 2.8GB + 1.2GB of internal flash memory (user + system),
6. TFT color display, width 240px, height 320px,
7. WiFi 802.11 b/g/n, Bluetooth 4.0,
8. GPS,
9. SD card up to 32GB support,

10. 2.0MPx camera.

The hardware equipment of the target device brings some constraints and limitations, which was taken into account during the design phase (e.g., design of custom graph search algorithm presented in Subsection 4.3.3).

Target device (see Figure 4.17) is one of a few remaining smartphones, which is equipped with a hardware keyboard instead of a touchscreen. Buttons are big, well spaced and tangible, which makes them suitable for VIP. Despite the fact, that most of the blind people overcame the end of cell phone era and adapted to the smartphones, hardware keyboard is still taken as an advantage.

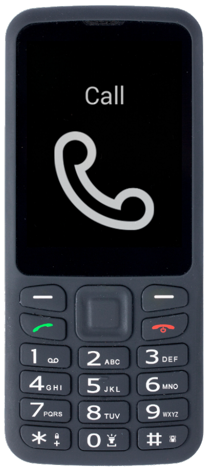


Figure 4.17: Target device

4.5 Conclusion

The theoretical description of a proposed mobile web browser architecture was given in Section 4.2. The Representation and navigation module was desribed in Subsection 4.2.1. The implementation of a proposed architecture and UI designed in Chapter 3 was documented in Section 4.3 including the explanation of designed tree search algorithm used for navigation in the page given in Subsection 4.3.3. The integration into the *Core system* environment (explained in Section 2.4) was described in Section 4.4. Specification of a target device was given in Subsection 4.4.2.

Chapter 5
User testing

This chapter documents the testing process methodology and the most important findings derived from the results of usability testing of the proposed mobile web browser for the blind. Usability testing is a user-centered technique for evaluating product ease of use by testing it on a user target group.

5.1 Introduction

The solution proposed in this thesis is targeted on VIP and is hopefully suitable for users with all levels of visual impairment (from intermediate vision impairment to no light perception).

The testing process was outlined as a standard usability testing process and consisted of 3 phases:

1. pre-test phase,
2. testing phase,
3. post-test phase.

The experiment was realized in cooperation with the Czech Blind United (SONS), which contacted and invited participants. 5 participant attended, and all of them were men. The answered pre-test and post-test questionnaires are present in Appendix A.

I want to thank Michal Jelínek (SONS) and Martin Procházka for their willingness and help with the realization of user testing and contacting test participants.

5.2 Pre-test

The pre-test phase aims to gather important details and user habits of test participants through short questionary. The collected information is crucial for the evaluation of test results in context. E.g., users with more experience with accessibility tools dedicated to web browsing might get familiar with application controls quickly than a slightly experienced user.

1. Your age?
    a. less than 20
    b. 20-35
    c. 35-50
    d. 50-65
    e. more than 65
2. Your gender?
    a. male
    b. female
3. Do you have a visual experience?
    a. yes
    b. no
4. How would you rate the level of your visual impairment from the functional perspective?
    a. intermediate vision impairment (able to read with reading glasses or optical magnifiers, navigates mostly visually)
    b. serious visual impairment (have some limited projection, able to read only with difficulties and with the help of digital magnifiers, navigates with the help of a white cane)
    c. blindness (no useful projection, unable to read at all, able to distinguish light from darkness)
    d. no light perception at all
5. Which mobile phone do you use?
6. Do you use any accessibility solutions for your mobile phone (e.g., TalkBack or VoiceOver)?
7. Do you use the internet?
    a. yes
    b. no
8. Do you use mobile internet?
    a. yes
    b. no

9. Do you access social media on your mobile phone?

10. For which kind of tasks do you use mobile internet (if you use it)?

11. How often do you use the internet?

    a. every day
    b. almost every day
    c. at least once a week
    d. at least once a month
    e. less than once a month

5.3 Tasks

This Section describes various and representative tasks chosen for usability testing. Tasks differ in difficulty and are sorted from the most easier to the hardest one. All tasks were chosen in a way, so it is possible to finish them in a reasonable time. First two tasks are not evaluated, because they are meant to get test participant familiar with browser control.

1. Read the manual of Browser application to get familiar with its usage and control.

2. Find the application called Browser in the phone menu under the 'More applications' or open it with voice command 'browser' (or 'web' or 'internet'). To get familiar with the browser more quickly, complete following mini-tutorial:

    a. The default homepage should be already loaded. Try to explore the homepage content in the default navigation mode. (By pressing left and right d-pad buttons).
    b. Use any different navigation mode in the page. (By pressing buttons 1-6)
    c. Using quick navigation actions Page start (by pressing button 7) and Page bottom (by pressing button 9) and use it to locate the first and last element on the page.
    d. Using link navigation mode (by pressing button 4) find all links in the page.
    e. Using controls navigation mode (by pressing button 3) find all control elements on the page.
    f. Repeat text for the currently focused element (by pressing down d-pad button).
    g. Repeat title of the current page (by pressing button 0).

   h. Go to the browser menu (by pressing button menu) and explore its items, then return to the page (by pressing the button back).
   i. Go to the browser menu, locate option 'Reload' and confirm.

3. Open browser menu, locate option 'Insert URL or search text' and confirm. Type URL of the Czech Wikipedia 'cs.m.wikipedia.org' to the keyboard and confirm. After the page is loaded, try to read 'Article of the week' present on this page. While reading, try to use word navigation mode (by pressing button 5).

4. Open browser menu, find option 'Bookmarks' and confirm. Try to save the current page as a bookmark and then locate it in 'List bookmarks.'

5. Open browser menu and confirm 'Insert URL or search text' again. Type 'CTU,' which is an abbreviation for the Czech Technical University, and confirm. The committed text should be searched using the default search engine. Try to locate the link to official university web site 'www.cvut.cz' among the search results and click on that link (by pressing center d-pad button).

6. Open browser menu, find option 'Bookmarks' and confirm. Try to save the current page as a bookmark too.

7. Open browser menu and confirm option 'Go to the homepage.'

8. Try to find the following information using 'Insert URL or search text' option in the menu (try use text dictation by long pressing volume down button while in keyboard at least for some searches):

   a. name of the capital city of state Kentucky (USA),
   b. the birth date and place of Winston Churchill,
   c. address of Department of Justice of the Czech Republic.

9. 'Go to homepage' and try to perform a search of your choice using the search entry in the page. (Pressing Dpad center performs a click, which opens and closes keyboard if the input element focused.)

10. Open the web page 'www.ctk.cz' and get the title of the second article on the main page.

11. 'Go to homepage,' locate 'Find in page' option in the menu (by pressing shortcut button 8) and type 'Google.' Explore all occurrences of that word on the page.

12. Open browser menu and try to use options 'Back' and 'Forward.'

13. Open browser menu and open any bookmark. After bookmark is loaded return to the menu again and find 'Settings,' confirm and then locate 'Homepage settings.' Try to set the current page as a new homepage.

14. Open browser menu, go to 'Settings,' then 'Speech output settings,' then 'Configure TTS spoken events' and enable speech output for all events. Then 'Reload' page.
15. Open browser menu, go to 'Settings' again, then find 'Speech output settings' and 'Earcons settings.' Try to enable Earcons for all events. Return to the homepage and try to trigger some events resulting in playing earcons.
16. Open browser menu, go to 'Settings,' then find 'Search engine' and select different search engine for searches made through 'Insert URL or search text.' Then find 'Insert URL or search text' and perform a search of your choice.
17. Open browser menu, find an option 'Turn on Private Browsing' and confirm. Now your browsing should be private.
18. Open the web page 'www.youtube.com' and try to play any of the videos present on the main page.
19. Open browser menu, find 'List history,' delete some page from history and then open some other page of your choice from history.
20. Open browser menu and try 'Clear history.' After dialog disappears, try 'List history' item and make sure history is empty.
21. Open browser menu and, find the option 'Take Screenshot' and confirm. See what happens.
22. Return to browser menu once more, locate last option 'Close Browser' and confirm the selection.

5.4 Post-test

The post-test phase aims to collect valuable feedback from test participants. The post-test questionnaire administered after the test and consists of the following questions:

1. Did you find the tasks difficult?
2. Which task was the hardest one?
3. How do you like the 'Browser' application?
4. Do you have any feedback comments or ideas for improvements?
5. Did you missed any specific function?

5.5 Testing summary

The youngest participant was 39 years old and the oldest 54. As mentioned before, all of them were men. The 4 of 5 test participants were blind, and 1 had a serious vision loss. The most often used smartphone among the participants was iPhone together with VoiceOver accessibility service. All participants answered that they use the internet every day on both smartphone and desktop PC, but only 2 of them access social media on the phone.

The common tasks participants use mobile internet for are email, reading articles, shopping and searching for information.

After an initial struggle with controls of the Browser, all participants finished all tasks quite well. The individual tasks performed by participants were rated by a number from 1 (finished without any problem in reasonable time) to 5 (did not finish the task at all) similarly to school grading systems. The grades were annotated with problems if occurred. The worst grade was 3, and it occurred only once in the first task. The significant part of the tasks among all participants was rated 1. In the end, all participants stated that tasks were not hard.

All participants liked the Browser application with some remarks. Participants said that the Browser is usable, simple, functional, and easy to control, but some of them missed specific functions or navigation options. Participants stated that with some improvements, the Browser could be perfect accessibility solution.

5.6 Important findings of the testing

The most important finding of the performed test is that blind participants considered the Browser to be usable and liked its simplicity and concept. However, each of them missed some functions and proposed some improvements. The most important are the points where all of them agreed. All participants missed automatic reading function, i.e., reading the text of the page until the user stops it. The current solution allows reading articles only by iterating over individual paragraphs. They suggested that this function should be assigned to the right d-pad button on long press. This feature is critical and should be implemented in the future. A significant part of them also suggested that there should be an option to send current URL address and currently selected link. The current URL should be also filled in the 'Enter URL or search text' menu option.

Some proposed improvements, which was suggested by single participant, but was found worth consideration are

1. use only one earcon for page start and page end,

2. allow choosing of navigation mode in the menu as an alternative for buttons,
3. there should be navigation option for localization of tables and lists,
4. it could be announced for links in which ARIA landmark they belong,
5. the earcon for a link should be shorter (e.g., click),
6. the announcement for page loaded and page scanned should be joined into one,
7. shortcut for 'Find in Page' assigned to 8 should open keyboard directly instead of an item in the menu,
8. assign navigation modes to the buttons as in old Nokia legacy phones.

5.7 Conclusion

This Chapter 5 documents the usability testing of designed web browser integrated into the *Core System*. The methodology of performed user testing is described in Section 5.1. Individual phases of testing are described in dedicated Sections 5.2,5.3 and 5.4. General testing summary is given in Section 5.5 and important findings are subsequently revealed in Section 5.6. The Browser was found usable; participants liked it and suggested some valuable improvements. They all missed Automatic reading function, which will be incorporated based on their feedback.

Chapter 6

Conclusions

The brief introduction into the world of ATs, web accessibility, and smartphones as a universal platform for delivering various types of accessibility tools was given in Chapter 1. Chapter 2 was dedicated especially to the *Core System*, which was the target system for the proposed web browser integration, but other existing relevant accessibility tools, browsers, and technologies were documented as well. The UI design was described in detail in Chapter 3 and its Sections. Implementation and following integration into the *Core system* is explained in Chapter 4. The process of usability testing experiment and derived valuable findings are documented in Chapter 5. It was found that the Browsed is usable, and we can say that blind participants liked the concept. They suggested improvements, which were analyzed, and some of them were used as valuable feedback for future development.

The contribution of this thesis is mainly the enrichment of the *Core system* used by many blind people with a mobile web browser designed especially for them. The Browser solution is inspired with verified approaches and integrates them with minimalistic *Core system* principles.

I want to state that I was impressed with the positivity of the blind people I met during the usability testing. Blind people struggle every day with situations most of us can hardly imagine while being smiley, positive and nice. I am convinced that blind people are very inspiring and beneficial for society if equipped with useful AT allowing them to live autonomously.

Bibliography

[1] Inc. Apple. Webkit. `https://webkit.org/`, 2019. [Online; accessed 1-April-2019].

[2] Alexy Bhowmick and Shyamanta M Hazarika. An insight into assistive technology for the visually impaired and blind people: state-of-the-art and future trends. *Journal on Multimodal User Interfaces*, 11(2):149–172, 2017.

[3] BrailleSurf. Braillesurf. `http://www.snv.jussieu.fr/inova/bs4/uk/index.htm`, 2019. [Online; accessed 1-April-2019].

[4] Chromium Contributors. Chromium/blink repository. `https://chromium.googlesource.com/chromium/blink`, 2019. [Online; accessed 1-April-2019].

[5] Corvus. Corvus. `http://www.corvuskit.com`, 2019. [Online; accessed 1-April-2019].

[6] CSS-Tricks. Average web page data analyzing 8 million websites. `https://css-tricks.com/average-web-page-data-analyzing-8-million-websites/`, 2019. [Online; accessed 1-April-2019].

[7] Mozilla Foundation. Accessibility add-ons. `https://addons.mozilla.org/en-US/firefox/collections/100508/accessibility/`, 2017. [Online; accessed 1-April-2019].

[8] Mozilla Foundation. Mozbraille. `http://mozbraille.mozdev.org/`, 2017. [Online; accessed 1-April-2019].

[9] Mozilla Foundation. Firefox. `https://www.mozilla.org/en-US/firefox/`, 2019. [Online; accessed 1-April-2019].

[10] Mozilla Foundation. Firefox focus. `https://www.mozilla.org/en-US/firefox/mobile/#focus`, 2019. [Online; accessed 1-April-2019].

[11] Mozilla Foundation. Firefox reality. https://blog.mozilla.org/blog/2018/09/18/firefox-reality-now-available/, 2019. [Online; accessed 1-April-2019].

[12] Mozilla Foundation. Gecko accessible role. https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XPCOM/Reference/Interface/nsIAccessibleRole, 2019. [Online; accessed 1-April-2019].

[13] Mozilla Foundation. Geckoview. https://wiki.mozilla.org/Mobile/GeckoView, 2019. [Online; accessed 1-April-2019].

[14] Mozilla Foundation. Mozilla. https://www.mozilla.org, 2019. [Online; accessed 1-April-2019].

[15] Mozilla Foundation. Mozilla accessibility. https://developer.mozilla.org/en-US/docs/Mozilla/Accessibility, 2019. [Online; accessed 1-April-2019].

[16] Ritwika Ghose, Tirthankar Dasgupta, and Anupam Basu. Architecture of a web browser for visually handicapped people. In *2010 IEEE Students Technology Symposium (TechSym)*, pages 325–329. IEEE, 2010.

[17] Google. Accessibility overview. https://developer.android.com/guide/topics/ui/accessibility, 2019. [Online; accessed 1-April-2019].

[18] Google. Accessibility service. https://developer.android.com/reference/android/accessibilityservice/AccessibilityService, 2019. [Online; accessed 1-April-2019].

[19] Google. Android activity. https://developer.android.com/reference/android/app/Activity, 2019. [Online; accessed 1-April-2019].

[20] Google. Github talkback repository. https://github.com/google/talkback, 2019. [Online; accessed 1-April-2019].

[21] Google. Google accessibility suite. https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback, 2019. [Online; accessed 1-April-2019].

[22] Google. Google text-to-speech. https://developer.android.com/reference/android/speech/tts/TextToSpeech.html, 2019. [Online; accessed 1-April-2019].

[23] Google. Talkback manual chrome. https://support.google.com/accessibility/android/answer/2633135?hl=en&ref_topic=3529932, 2019. [Online; accessed 1-April-2019].

[24] Google. Webview. https://developer.android.com/reference/android/webkit/WebView, 2019. [Online; accessed 1-April-2019].

[25] Lilit Hakobyan, Jo Lumsden, Dympna O'Sullivan, and Hannah Bartlett. Mobile assistive technologies for the visually impaired. *Survey of ophthalmology*, 58(6):513–528, 2013.

[26] Apple Inc. Vision accessibility. `https://www.apple.com/accessibility/iphone/vision/`, 2019. [Online; accessed 1-April-2019].

[27] A King, G Evans, and P Blenkhorn. Webbie: a web browser for visually impaired people. Citeseer.

[28] Mozilla. Mozilla gecko repository. `https://github.com/jostw/gecko/tree/master/mobile/android/`, 2019. [Online; accessed 1-April-2019].

[29] Center For Development of Advanced Computing. Shrutidrishti. `https://www.cdac.in/index.aspx?id=mc_st_shruti_drishti`, 2019. [Online; accessed 1-April-2019].

[30] PetGlobal. Smartphone ownership is growing rapidly around the world, but not always equally. `https://www.pewglobal.org/2019/02/05/smartphone-ownership-is-growing-rapidly-around-the-world-but-not-always-equally/`, 2019. [Online; accessed 1-April-2019].

[31] Petr Svobodník. Zpřístupnění mobilních telefonů se systémem android pro nevidové uživatele. 2013.

[32] W3C. Aria landmarks. `https://www.w3.org/TR/wai-aria-practices/examples/landmarks/index.html`, 2019. [Online; accessed 1-April-2019].

[33] WebAIM. Screen reader survey. `https://webaim.org/projects/screenreadersurvey7`, 2017. [Online; accessed 1-April-2019].

[34] WebbIE. Practical accessible web design. `https://www.webbie.org.uk/accessibilityinwebdesign.htm`, 2017. [Online; accessed 1-April-2019].

[35] World Health Organization (WHO). Blindness and visual impairment. `https://www.who.int/en/news-room/fact-sheets/detail/blindness-and-visual-impairment`, 2019. [Online; accessed 1-April-2019].

[36] Cameron Wong. eguidedog. `https://www.eguidedog.net/`, 2019. [Online; accessed 1-April-2019].

Appendix A

User-testing questionnaires

A.1 Pre-test questionnary

Question	Part. 1	Part. 2	Part. 3	Part. 4	Part. 5
1	d	c	c	c	c
2	male	male	male	male	male
3	no	yes	yes	yes	yes
4	d	d	c	c	c
5	I	CS	I,A	I	I
6	VO	-	VO, TB	VO	VO
7	yes	yes	yes	yes,	yes
8	yes	yes	yes	yes	yes
9	no	no	yes	no	yes
10	e, a	e, tt, sh,	tt, s	s, sh, a, m	a

Table A.1: Pre-test questionnary.

Legend for Table A.1:

1. CS - the *Core system*
2. I - iPhone
3. A - Android
4. VO - VoiceOver
5. TB - TalkBack
6. e - email, a - articles, tt - time tables, s - searching, sh - shopping, m - multimedia

A.2 Post-test questionnary

A.2.1 Participant 1

1. No.
2. None.
3. Usable, simple.
4. Current URL should be in ’Enter URL or seach text’ keyboard. Navigation modes should be in menu as well. Missing navigation mode for table localization.
5. Automatic reading.

A.2.2 Participant 2

1. No.
2. None.
3. Usable, simple.
4. Try use the *Core system* keyboard used in menu for in-page entry. Last search term should remain in ’Find in Page’ keyboard.
5. Automatic reading, medial element control.

A.2.3 Participant 3

1. No.
2. None.
3. Good, simple, easy to control.
4. Join announcments for page loaded and page scanned into one. Announce return to page after entering search text in ’Find in Page’ keyboard.
5. Automatic reading.

A.2.4 Participant 4

1. No.
2. Orientation in YouTube page.
3. Usable, liked it.

4. Announce number of headings after switching to headings navigation. Switch 'Back' and 'Forward' items in menu. Possibility to rename bookmark. Confirm dialog for 'Clear history'.

5. Automatic reading.

A.2.5 Participant 5

1. No.

2. None.

3. Easy.

4. Read 'link' after text of the link. Announce tables and lists. 8 should open keyboard directly not menu item 'Find in Page'. Link could announce ARIA landmark where it belong.

5. Automatic reading, sending of current URL and link, repeating of current URL.

Appendix B

Abbreviations used in thesis

Abbreviation	Meaning
VI	Visual impairment
VIP	Visually impaired people
QoL	Quality of life
ADL	Activities of daily live
UI	user interface
PDF	portable document format
WWW	World Wide Web
HTML	Hyper Mark-up Text Language
AT	Assistive technology

Table B.1: Abbrevations used in thesis.

Appendix C

CD Content

Directory name	Description
thesis	Contains this Master thesis in PDF format.
thesis sources	Contains LaTeX source codes used to generate this thesis.
java	Contains source code of the web browser for blind written in Java.
python	Contains source code of scripts written in Python used to generate plots.

Table C.1: CD content.