TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
40/2019

Automatic Implementation-Time Usability Evaluation for Web User Interfaces

JEVGENI MARENKOV

TAL TECH PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Computer Systems
This dissertation was accepted for the defence of the degree 11/04/2019

Supervisor: Doctor Tarmo Robal, PhD
Dept. of Computer Systems
Tallinn University of Technology
Tallinn, Estonia

Co-supervisor: Professor Ahto Kalja†, PhD
Dept. of Software Science
Tallinn University of Technology
Tallinn, Estonia

Opponents: Professor Janis Grundspenkis, Dr. habil.sc.ing.
Dept. of Artificial Intelligence and Systems Engineering
Riga Technical University
Riga, Latvia

Professor Flavius Frasincar, PhD
Erasmus School of Economics
Erasmus University Rotterdam
Rotterdam, The Netherlands

Defence of the thesis: 28/06/2019, Tallinn

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology has not been submitted for doctoral or equivalent academic degree.

Jevgeni Marenkov:

Signature

European Union
European Regional
Development Fund

Investing
in your future

ISSN 2585-6898 (publication)
ISBN 978-9949-83-440-2 (publication)
ISSN 2585-6901 (PDF)
ISBN 978-9949-83-441-9 (PDF)

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
40/2019

Veebi kasutajaliidese kasutatavuse automaatne hindamine realisatsioonifaasis

JEVGENI MARENKOV

TAL TECH
KIRJASTUS

Contents

List of Publications 7
Author's Contribution to the Publications 8
Other Related Publications 9
Abbreviations 10
1 Introduction 11
1.1 Motivation 11
1.2 Problem Formulation 14
1.3 Thesis Contributions 16
1.4 Thesis Organization 17
2 Preliminaries 19
2.1 Usability, Accessibility and User Experience of WUIs 19
2.2 Web Usability Guidelines 21
2.3 Usability Evaluation Methods 23
2.4 Automated Usability Evaluation 26
2.5 Related Works 29
2.5.1 Research in the Field of Usability Guidelines 29
2.5.2 Research on Automated Usability Evaluation 31
2.6 Chapter Summary 38
3 Usability Guidelines for Automated UI Evaluation 39
3.1 Introduction 39
3.2 Sources of Usability Guidelines 41
3.2.1 Accessibility Standards 41
3.2.2 HCI Research 43
3.2.3 Company Specific Usability Guidelines 44
3.3 Defining Usability Guidelines Programmatically 45
3.3.1 Embedded Usability Guidelines 46
3.3.2 Separated Usability Guideline Definition 46
3.3.3 Limitations of Existing Approaches 49
3.4 Ontology for Defining Usability Guideline 50
3.4.1 Usability Ontology 51
3.4.2 Using Usability Domain Knowledge to Define Usability Guidelines 55
3.5 Chapter Summary 58
4 Automated Evaluation of Usability 61
4.1 Introduction 61
4.2 Guideliner – a Tool for Automated Web Usability Evaluation 63
4.3 Ontology Repository and Ontology Processing Engine 64
4.4 WUI Evaluation Component 65
4.4.1 Usability Evaluation Report 70
4.5 Interfacing with the Guideliner 72
4.6 Performance Testing and Optimization of the Guideliner Tool 76
4.7 Guideliner and Other Similar Tools 80
4.8 Limitations of the Guideliner 89
4.9 Chapter Summary 90

5 Usability Evaluation within WUI Development 91
5.1 Introduction 91
5.2 Usability Evaluation during WUI Development Process 93
5.3 Guideliner Integration into WUI Implementation Phase 95
5.4 Guideliner in Continuous Software Integration Process 99
5.5 Integration Use Cases 101
5.5.1 Integration into RIA WUI Development Process 101
5.5.2 Integration into Kühne + Nagel WUI Development Process 106
5.6 Chapter Summary 110
6 Conclusions and Future Work 113
6.1 Conclusions 113
6.2 Future Work 115
References 116
Acknowledgements 125
Abstract 126
Lühikokkuvõte 127
Appendixes 129
Appendix 1 Example of Usability Guideline Structure 131
Appendix 2 Main Concepts of the Usability Ontology 133
Appendix 3 Usability Guidelines Used for the Research 134
Appendix 4 Guideliner Software Architecture 138
Appendix 5 Usability Evaluation Report Based on EARL 139
Appendix 6 WUIs Used for Guideliner Verification 140
Appendix 7 Comparison of the Benchmark Tools used for the Guideliner Verification 143
Appendix 8 Comparison of Features of Automated Usability Evaluation Tools 144
Appendix 9 Results of Usability Evaluation 145
Curriculum vitae 149
Elulookirjeldus 150

List of Publications

The list of author's publications, on the basis of which the thesis has been prepared:

I Marenkov, J., Robal, T., & Kalja, A.: Design-Time Web Usability Evaluation with Guideliner. Complex Systems Informatics and Modeling Quarterly, (15), 90–109. (2018)

II Marenkov, J., Robal, T., & Kalja, A.: Guideliner: a tool to improve Web UI development for better usability. WIMS '18: 8th International Conference on Web Intelligence, Mining and Semantics: June 25 - 27, 2018, Novi Sad, Serbia. Editors: Akerkar, R.; Ivanovic, M.; Kim, S.-W.; et al. Article No 17, 1–9. ACM, New York (2018)

III Marenkov, J., Robal, T., & Kalja, A.: A tool for design-time usability evaluation of web user interfaces. Advances in Databases and Information Systems: 21st European Conference, ADBIS 2017, Nicosia, Cyprus, September 24-27, 2017, Proceedings. Editors: Kirikova, M.; Nørvåg, K.; Papadopoulos, 394–407. Springer, Cham (2017)

IV Robal, T., Marenkov, J. & Kalja, A.: Ontology design for automatic evaluation of web user interface usability. PICMET '17: Proceedings, Technology Management for Interconnected World: July 9 - 13, 2017, Portland, Oregon, USA. Editors: Kocaoglu, PICMET, USA (2017)

V Marenkov, J., Robal, T., & Kalja, A.: A framework for improving web application user interfaces through immediate evaluation. Databases and Information Systems IX: Selected Papers from the Twelfth International Baltic Conference, DB&IS 2016: Riga, Latvia; 4-6 July 2016. Editors: Arnicans, Guntis; Arnicane, Vineta; Borzovs, Juris; Niedrite, Laila. IOS Press, Amsterdam (2016)

Author's Contribution to the Publications

Contributions to the papers in this thesis are:

I The author presented a tool called *Guideliner* that automates usability evaluation in implementation phase of web UI development. The author developed the *Guideliner* tool based on the designed concepts. The author prepared and conducted an experiment to prove the feasibility of the *Guideliner*. The author prepared the paper for publication in cooperation with supervisors.

II The author presented the *Guideliner* – a tool that evaluates the conformance of web UI to the set usability guidelines and provides feedback to the developer immediately after the UI has been modified. The author developed the tool based on the conceptual foundation presented in previous papers. The author wrote the paper in cooperation with supervisors and presented it at the conference.

III The author presented a new approach to the usability evaluation contributing to the design time usability evaluation enabling immediate cost-efficient automatic web UI evaluation. The author developed the concept, implemented all software components and conducted experiments. The author wrote the paper in cooperation with supervisors and presented it at the conference.

IV The author participated in ontology design and development through numerous discussions with the supervisors. The author was responsible for implementing numerous usability guidelines in the ontology and developing a software adapter for transforming ontology-based usability guidelines to the Java-specific format for the *Guideliner* tool.

V The author established a framework enabling automated evaluation of various usability guidelines. The author developed the software component for automated evaluation of visual characteristics of web UI. The author prepared the paper for publication in cooperation with supervisors and presented it at the conference.

Other Related Publications

I Marenkov, Jevgeni; Robal, Tarmo; Kalja, Ahto (2016). A study on immediate automatic usability evaluation of web application user interfaces. Databases and Information Systems: 12th International Baltic Conference, DB&IS 2016, Riga, Latvia, July 4-6, 2016, Proceedings. Editors: Arnicans, G., et al. Springer, Berlin (2016)

II Marenkov, Jevgeni; Robal, Tarmo; Kalja, Ahto (2015). A study on effective knowledge reuse in multi-platform Web applications user interfaces. PICMET '15: Proceedings, Management of the Technology Age [August 2-6, 2015, Portland, Oregon, USA]. Editors: Kocaoglu, Dundar F., et al. PICMET, USA (2015)

III Marenkov, Jevgeni; Robal, Tarmo; Kalja, Ahto (2014). A study on user click behaviour for WIS user interface improvements. Databases and Information Systems VIII: Selected Papers from the Eleventh International Baltic Conference, Baltic DB&IS 2014. Editors: Haav, Hele-Mai; Kalja, Ahto; Robal, Tarmo. Amsterdam: IOS Press, 173−186. IOS Press, Amsterdam (2014)

IV Marenkov, Jevgeni; Robal, Tarmo; Kalja, Ahto (2014). Improving sophisticated self-service portal user interfaces: a study on user click behaviour. Databases and Information Systems: Proceedings of the 11th International Baltic Conference, Baltic DB&IS 2014, Tallinn, Estonia, 8-11 June 2014. Editors: Haav, Hele-Mai; Kalja, Ahto; Robal, Tarmo. Tallinn: Tallinn University of Technology Press, 159−170. Tallinn University of Technology Press, Tallinn (2014)

V Marenkov, Jevgeni; Robal, Tarmo; Kalja, Ahto (2014). Studying the incorrect user behaviour in sophisticated self-service portal user interfaces. Proceedings of the 8th Annual Conference of the Estonian National Doctoral School in Information and Communication Technologies: December 5-6, 2014, Rakvere, 71−74. Tallinn University of Technology Press, Tallinn (2014)

Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
CI	Continuous Integration
CSS	Cascading Style Sheets
DAML	DARPA Agent Mark-up Language
DOM	Document Object Model
EARL	Evaluation and Report Language
ESP	Estonian State Portal
EU	European Union
GDL	Guideline Definition Language
GRF	Guideliner Report Format
HCI	Human-Computer Interaction
HE	Heuristic Evaluation
IDE	Integrated Development Environment
IT	Information Technology
JAR	Java Achieve
JRE	Java Runtime Environment
HTML	Hypertext Mark-up Language
JSON	JavaScript Object Notation
MB	Megabyte
OIL	Ontology Inference Layer
OWL	The W3C Web Ontology Language
PC	Personal Computer
RIA	Republic of Estonia Information System Authority
QA	Quality Assurance
RDF	Resource Description Framework
REST	Representational State Transfer
RT	Research Topic
SEO	Search Engine Optimization
TUT	Tallinn University of Technology
TV	Televis`ion
UGL	Unified Guidelines Language
UI	Use Interface
UK	United Kingdom
URL	A Uniform Resource Locator
US	United States
UX	User Experience
W3C	World Wide Web Consortium
WCAG	Web Content Accessibility Guidelines
WUI	Web User Interface
XML	Extensible Mark-up Language

1 Introduction

Nowadays, the amount of time people spend on the Internet is constantly increasing. People are spending approximately twice as much time online compared to 10 years ago [1]. We are using the Web for searching for information, checking e-mails, communicating with people, transferring money, listening to music, watching films, reading books, and so forth. Moreover, online e-commerce stores allow us to purchase products and services online. We are moving towards a seamless society of interconnected e-services where e-services and the technologies behind Internet of Things advance; services for everyday life, including government services are available online – including the e-health for medical data, e-voting for elections, e-school for educational services and many others. Overall, most services can be accessed online, so people can communicate to each other, use private, public and commerce services without leaving their home.

1.1 Motivation

In the modern world, technology advances extremely rapidly. For instance, in addition to desktop computers laptops have become increasingly popular starting from 2000s. The year 2008 was the watershed year for laptops when worldwide laptop shipment exceeded desktop computer shipment [2]. In 2007, the first mobile devices with touchscreen were introduced starting a new era of touchscreen smartphones replacing typical keypads. Mobile phones and tablets are on great demand amongst the users as people cannot imagine the world without social networks, online e-commerce services, entertainment portals (e.g. Spotify[1]) that are all widely used on personal computers and mobile devices. Devices use different operating systems (Android, iOS, Windows, Linux etc.) that manage computer hardware and software. Today, inevitable software of every commercial-grade computing device for general usage is web browser (a software application used to locate and display web pages [3]). The most widely used browsers according to the statistics[2] are Chrome, Firefox, Internet Explorer and Safari (presented in the order of popularity). Overall, there are about 113 different web browsers[3]. Web pages that are accessible from browsers are based on three primary technologies: HTML, CSS and JavaScript. HTML is the language that provides the structure of web pages; CSS is the language that describes the layout and representation of web pages; JavaScript is interpreted language for developing web pages. In fact, different browsers interpret standards HTML, CSS and JavaScript differently, and thereby, for example, a good-looking and fast web application in Chrome can be very slow and unusable in Internet Explorer or vice versa. Despite the standards for web content delivery, such as HTML, CSS and JavaScript, browsers render web pages differently as they use their own rendering engines (e.g. Blink engine is used in Chrome, Gecko engine is used in Firefox) to display requested content on screen. Finally, rendered visual result of web page is presented to users as user interface.

A user interface (UI) is the “face” of any application as users communicate with the system back end by the means of UI. UI is used in a very broad sense combining two main concepts: UI of native applications and UI of web applications. Native applications are

[1] https://www.spotify.com/

[2] http://gs.statcounter.com/#desktop-browser-ww-monthly-200807-201712

[3] https://www.webdevelopersnotes.com/browsers-list

software programs that are developed for a particular platform (e.g. Android specific native applications will not run on iOS platform). Current thesis concentrates on UI of web applications – a program that runs in whole or in part on one or more web servers and that can be run by users through a web site [4]; web application run in web browsers and can be accessed from any device that has a web browser installed on for accessing the Web (web applications). In the context of this thesis, Web application is a client-server system with which users interact over the network using Web browsers [5]. A client side (also called the front-end) component of Web application is named Web User Interface (WUI) that is formalised as a Web page which is accessible over the network and viewed using the Web browser [6].

WUI of web applications should be designed to support various devices with different screen sizes, resolutions, and operating systems. Moreover, WUI should be equally compatible with different browsers and their version. This is an important requirement as users can access WUI from different browsers and devices and if WUI is not understandable or it does not respond quickly to user actions then users can leave it in favour of other application. Although current thesis mainly concentrates on the most popular device types – desktop and mobile devices[1] – the methods introduced in the thesis can also be applied to the tablet devices.

In fact, WUI should be understandable, user-friendly, navigable, smooth and easy to use and to learn, having clear structure of information and navigation. All these requirements and characteristics are part of ISO 9241-11 usability definition – "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [7]. Usability covers comfort and acceptability of WUI to its target users (users that use or intend to use WUI). In order to design WUI that satisfies all aforementioned requirements, it should follow various usability guidelines that guide designers to establish solutions understandable for the majority of users. Usability guideline is considered in the dissertation as an evaluable concrete rule, criterion or principle pertaining to the evaluation of usability [8]. Usability guidelines are developed with the purpose to help WUI developers and designers to develop highly usable WUI. Usability guidelines are not part of usability definition but rather a separate area concentrating on usability research with the purpose to come up with concrete guidelines that user-friendly, understandable and navigable WUIs should satisfy. In order to verify that WUI is compatible with usability guidelines, WUI usability evaluation should be performed.

Usability evaluation is a method for identifying specific problems with usability of products [9]. There are two major groups of methods used for usability evaluation: inspection and empirical methods. Usability inspection methods include formal usability inspections, heuristic evaluation, pluralistic and cognitive usability walkthrough. Empirical evaluation methods contain interviews, questionnaires with user-test participants and card sorting. The main difference between inspection and empirical evaluation methods is that the latter involves real users into the evaluation, whereas inspection methods require usability inspection to be held by usability experts. Another difference between the methods is that empirical evaluation methods are concentrating on finding a wider range of problems whereas inspection usability evaluation is limited

[1] http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-201701-201712

by finding the problems defined in specification. Overall, both inspection and empirical evaluation methods can be automated.

Automated usability evaluation has multiple advantages over manual methods such as it does not require the involvement of potential users in usability evaluation. The benefits of automated evaluation are the increase of the coverage of evaluated WUI elements, the results of evaluation are processed by computer and the evaluation report is provided automatically. Overall, automated usability evaluation reduces the costs and time spent on usability evaluation.

From the perspective of automated usability evaluation, it is possible to automate certain aspects of empirical and inspection evaluation methods, e.g. automated evaluation of WUI conformance to usability guidelines, automated evaluation of usability based on collected user interaction data. Yet, none of existing tools or solutions for automated usability evaluation can entirely substitute empirical and inspection evaluation methods as tools cannot emulate human behaviour and evaluate usability based on human common sense. For the most part, automated evaluation on empirical evaluation methods is based on capturing user interaction data (clicks, mouse motion, navigation patterns) with the following data processing to identify possible interaction-based usability problems.

Automated usability evaluation on inspection methods scans WUI and checks its conformance to usability guidelines. The method of automated checking of the conformance of WUI to usability guidelines has been proved as the most effective method used for automated usability evaluation [16]. There are many existing tools that check WUI conformance to usability guidelines [10] [11] [16]. Nevertheless, there are unresolved problems in usability evaluation such as the inability to evaluate visual characteristic of WUI [12] and insufficient support for implementation-time usability evaluation [13]. Aforementioned shortcomings are the motivation for current research.

Current thesis concentrates only on automated usability evaluation on inspection methods as usability inspection methods are more flexible to automation and there are multiple methods how inspection methods are automated. In fact, tools on automating inspection evaluation are applicable to the most WUI without extra configurations, whereas tools for empirical evaluation require additional application specific configurations as they evaluate interaction-based problems that are in most cases application specific rather than general. Existing solutions for automated usability evaluation on inspection methods are mostly concentrating on the evaluation of WUI conformance to HTML code specific usability guidelines [10] [11]. Nevertheless, existing solutions of automated inspection evaluation cannot evaluate visual aspects of a WUI including measuring the presence of scrolling, the layout of elements on the web page and the positions of elements on the screen. Automated evaluation of visual characteristics of WUI increases the number of usability defects that can be evaluated automatically. Automatic evaluation of visual characteristics of WUI is one of the motivations for current dissertation.

Another common drawback of existing solutions for automated evaluation on inspection methods is that they are not used for checking each minor change of WUI separately but rather used to check when all functionality for release is finalised. It introduces the gap between the moment usability defect has been detected and the moment when it has been introduced by developer. This is the gap the research presented in this dissertation addresses.

As stated there are still unresolved questions like automated evaluation of visual characteristics of WUI and evaluation of WUI conformance to usability guidelines immediately after the code has been modified by developer [14]. The most existing solution presented for automated usability evaluation are concentrating on the evaluation of final product rather than on immediate evaluation of introduced changes [10] [15] [16] – implementation-time automated usability evaluation is the problem that is challenged in this thesis.

To sum it up, WUI usability evaluation as a research area is developing fast: new usability guidelines for checking the conformance to the latest technologies (e.g. HTML5) and devices (usability guidelines for smartphones are not entirely applicable to tablet devices) appear, new tools for automated usability evaluation on inspection and empirical methods are introduced in order to automate usability evaluation as much as possible making the process of usability evaluation less expensive and faster. Common characteristics of all existing tools for automated usability evaluation is that they are used during the testing phase of WUI and are concentrating on evaluating only HTML-specific usability guidelines without possibility to evaluate usability guidelines covering visual characteristics (e.g. scrolling, contrast rate, visual position, order of elements, the distance between elements) of finally rendered WUI.

The motivation of the thesis is to address the aforementioned problems of implementation-phase usability evaluation and automated evaluation of visual usability guidelines. The thesis introduces a novel method for WUI automated evaluation on inspection methods of usability evaluation that can be applied already during implementation phase of WUI development. The method would deliver valuable feedback to WUI developers, who are not necessarily usability experts, on usability defects immediately after WUI has been modified. The method also introduces automatic evaluation of visual characteristics of WUI – an aspect still uncovered by methods and tools commonly applied during WUI testing.

1.2 Problem Formulation

The evaluation of newly established or revamped web user interfaces and assurance that they conform to usability guidelines should not be resource-consuming and laborious task that many companies are attracted to skip. Instead, it should be a cost- and time-efficient step of development applied as a custom practice, especially when software gets developed according to agile methods consisting of short and flexible iterations with frequent releases to production. Yet, usability evaluation is still not optimized in a way that it can be automated excluding the involvement of usability experts into the evaluation; of course, it is impossible to entirely exclude usability experts from the WUI development process as they produce custom usability guidelines and check their viability. Requiring usability experts for reviewing the evaluation results or making usability inspections and involving potential users to participate in usability evaluation are the reasons for skipping usability evaluation. Automated cost- and time-efficient usability evaluation is one of the driving forces of current thesis.

A typical WUI development process consists iteratively of the next six phases: planning, analysis, design, implementation, testing and maintenance [17]. Certain inspection evaluation methods such as heuristic evaluation and cognitive walkthrough can be applied already during the design phase when the initial prototypes and mock-ups are evaluated by usability experts. UI code writing occurs during the implementation phase when a developer establishes or modifies WUI source code

according to the requirements set in planning phase. Afterwards, during the testing phase, it is ensured that the result of the development is exactly the same as it was planned and that the UI after modifications is usable and user-friendly. In common, inspection and empirical usability evaluation occur during the testing phase when WUI is finalized for the release.

In general, during WUI development usability is not tested with each minor change separately but rather checked when all functionality for release is finalized. It introduces a gap between the moment a usability defect is found and the moment when it was introduced by developer. In general, WUI developers do not have enough expertise and competency to evaluate usability of WUI as their main responsibility is functional development of WUI, e.g. dividing design into components, writing front-end code and covering this code with tests. The lack of competence in usability is the reason why evaluation results should be presented to developers in a way that even developers without a good usability evaluation experience can understand the cause and fix the problem easily.

As was stated earlier, there is a gap between the moment usability defect is detected and the moment it was introduced by developer. The main reason, why automatic usability evaluation, in general, is not part of implementation phase of WUI developments is the lack of tools for automated usability evaluation suitable for WUI implementation phase. There are existing solutions that can be used during the testing phase of WUI development but there are obstacles (e.g. these tools require special environment to run and they cannot be executed together with unit or functional tests) preventing them to be used during implementation phase of WUI development. Implementation phase of WUI development is code-centric and it incorporates activities directly connected to WUI development like writing UI code and covering code with unit tests. In its turn, testing phase of UI is concentrating on UI testing using various methods and tools that increase the efficiency and accuracy of evaluation such as user tests and tools for automatic usability evaluation. Existing tools for automated usability evaluation fall into the category of tools suitable for testing phase being development process-centric rather than code-centric.

Another shortcoming of existing solutions for automated usability evaluation is that they are limited in finding deviations in the HTML code processing source code of WUI as a text and finding problems in HTML syntax and in HTML-specific guidelines; evaluating visual aspects of a web application like the presence of scrolling, consistency of layout between WUI elements and the positions of elements on the screen is not technically possible with that method. It is important to understand that visual aspects of WUI should be evaluated on the final rendered result shown to user, i.e. when all scripts have finished their loading. This calls for adopting an alternative method compared to HTML parsing approach used by most existing tools to evaluate WUI conformance to usability guidelines — tools that are capable of evaluating both visual aspects of WUI as well as perform HTML-based evaluations. There are certain existing solutions that can evaluate a single aspect of finally rendered WUI. For example, WAVE[1] evaluates contrast of elements on the screen; Google Mobile-Friendly Test[2] evaluates the size of links and buttons on mobile screens. Nevertheless, there is no solution for automated usability

[1] http://wave.webaim.org/

[2] https://search.google.com/test/mobile-friendly

evaluation of finally rendered WUI that enables evaluating multiple visual aspects of WUI, and also that allows defining custom usability guidelines.

The purpose of current thesis is to address the latter gaps of implementation-time usability evaluation and automated usability evaluation of visual characteristics of WUI. In particular, these gaps will be addressed through the following research themes (RT):

RT1 – To what extent can WUI usability be evaluated automatically? What inspection and empirical evaluation methods can be automated?

RT2 – Is it practical and to what extent is it possible to bring forward WUI usability evaluation in web application development process? Would earlier evaluation deliver value and what modifications in general a development process requires?

RT3 – To what extent ontology as a method suits for storing knowledge about usability domain and how ontology can be used for formalising usability guidelines? Would ontology provide more structured and easy to understand the way of formalising usability domain knowledge than XML Schema?

RT4 – Would it be possible to automate evaluation of usability guidelines covering visual aspects of WUI? Is it technically feasible to design and implement a code-centric integrated solution for automated WUI usability evaluation covering both HTML-specific usability guidelines and guidelines for visual characteristics as a tool suitable for automated cost-efficient usability evaluation during WUI development phase, and on final released product?

Overall, current dissertation is focusing on the problem of automated usability evaluation during the implementation phase of WUI development, including visual characteristics of WUIs – a feature missing in tools available today.

1.3 Thesis Contributions

In order to tackle the problems of implementation-time usability evaluation and automated evaluation of visual usability guidelines, this thesis delivers a system (called *Guideliner*) that addresses these problems and enables automated evaluation of conformance to usability guidelines (HTML-centric as well as visual usability guidelines) already during implementation phase of WUI development. Also, the *Guideliner* provides possibility to perform usability pre-release testing verifying that all developed features are compliant with usability guidelines. In general, the proposed system increases the overall quality of WUI as it performs the evaluation of HTML-specific usability guidelines as well as guidelines checking the visual characteristics of WUI. Applicability of the *Guideliner* does not stick to any particular WUI development process (e.g., agile or waterfall) but rather it is a universal tool challenging a problem of immediate usability evaluation, especially during WUI development and implementation phases. Overall, dissertation contributions could be listed as follows:

1. Firstly, establishment of a framework to improve WUI development process introducing the automated usability evaluation already during the implementation phase of WUI development process. That enables immediate validation of introduced changes to WUI and their conformance to usability guidelines. Such enhancement allows fixing usability defects early in the implementation phase making the fix less time-consuming than found later. This contribution reflects the research themes RT1 and RT2.
2. Secondly, a structure of usability ontology for capturing domain knowledge and a way of declaration usability guidelines in machine-processable and human-readable format has been created and applied for defining usability guidelines. Usability

ontology contains predefined set of usability guidelines and also allows defining additional custom usability guidelines based on the established usability domain knowledge. The research covers research theme RT3.

3. Thirdly, a method for evaluating visual characteristics of WUI based on usability guidelines is introduced. The novelty of the method is that it evaluates WUI conformance to usability guidelines on the final layout a user sees through WUI, i.e. when all scripts have finished their loading. Proposed method allows evaluating HTML-specific usability guidelines as well as guidelines evaluating visual characteristics of WUI. This contribution reflects RT4.
4. Fourthly, establishment of a tool for automatic WUI usability evaluation that evaluates finally rendered WUI evaluating HTML-based usability guidelines and guidelines covering visual characteristics of WUI, and is applicable both at the final established WUI evaluation phase (traditional approach), and for evaluating WUI conformance to set usability criteria during WUI implementation phase (novel method introduced in the thesis). The tool called *Guideliner* incorporates the questions addressed in RT 1,2,3,4 into the practical output presented as a tool for automated usability evaluation. The uniqueness of the *Guideliner* consists of implementation-time automated usability evaluation and in automated evaluation of multiple visual characteristics of WUI.

1.4 Thesis Organization

This thesis consists of six chapters.

Chapter 2 describes state-of-the-art of the topics of formalization of usability guidelines, usability evaluation methods, automated usability evaluation including issues of usability evaluation, usability guidelines, empirical and inspection evaluation methods, essentials of automated usability evaluation, and finally, research works related to current research are presented.

Chapter 3 focuses on usability guidelines for automated usability evaluation. The chapter begins with presenting the definition and categorization of usability guidelines, followed by discussion on automated usability, sources of usability guidelines and approaches for defining these guidelines. The section delivers usability ontology for storing usability domain knowledge.

Chapter 4 concentrates on automated usability evaluation. The Chapter presents existing solutions for automated usability evaluation discussing their advantages and disadvantages. Afterwards, the *Guideliner* is presented – a tool for automated evaluation of WUI conformance to usability guidelines. It gives an overview of the main components of the tool including ontology processing engine, usability evaluation component and ontology repository. Finally, an approach for automated evaluation of visual characteristics of WUI is outlined.

Chapter 5 contributes to the usability evaluation during the implementation phase of WUI development enabling immediate cost-efficient automatic WUI evaluation and feedback to developers to ensure the WUI under development conforms to set guidelines. Hence, this approach will assist developers and WUI testers in finding out usability problems automatically in early stage of WUI development which implementation stage of UI development.

Chapter 6 finalises the dissertation with conclusions and presents ideas for future research work.

There are nine appendix sections (Appendix 1-9) included at the end of the thesis.

2 Preliminaries

Web usability evaluation identifies problems with usability of WUI. Usability evaluation includes two main method groups: empirical and inspection evaluation methods. Overall, multiple disciplines are related to the usability evaluation such as usability, accessibility, usability guidelines and automated usability evaluation. This chapter provides overview of the state-of-the-art research covering main disciplines related to the current study.

Current dissertation is focusing on automated evaluation of WUI conformance to usability guidelines, and automated usability evaluation on inspection methods. Related aspects of usability, user experience, accessibility, sources of usability guidelines including standards, recommendations and best practices are discussed together with an overview of existing tools for automated usability evaluation.

2.1 Usability, Accessibility and User Experience of WUIs

The diversity of computers, laptops, smartphones, and tablets has become an intrinsic part of modern life and culture. People use aforementioned devices to access various private and public services like news, e-commerce and entertainment portals. All these resources are accessed from web that is why WUI compatibility with these devices is an essential requirement for each web application. Furthermore, WUI should also be compatible with various browsers including their different versions; in fact, the same browser can have different behaviour on different platforms. The diversity of devices, platforms and browsers have introduced a problem of compatibility – how to guarantee high level of usability for WUI users between all these platforms and ensure equal satisfaction. It is important to consider that some usability guidelines are applicable only to certain platforms and cannot be reused on another one. For example, guideline requiring radio buttons to be aligned vertically on mobile devices because of the limited screen size of mobile devices does not apply to desktop devices where the horizontally aligned radio buttons do not decrease usability of WUI. Notwithstanding, compatibility with different devices and platforms is only one attribute of usability. WUI should be consistent between pages (the styles and design patterns used on one page should be followed within all other pages), clear, understandable, user-friendly, easy to use and navigate. Usability, as first defined by Nielsen [18] is "*a quality attribute that assesses how easy user interfaces are to use*". Usability applies to all aspects of a system with which a human might interact. Usability is traditionally related to the following five quality attributes [19]:

- learnability as the easiness to get something done rapidly,
- efficiency as the ability to productively use the system
- memorability as the capability for a user to return to system usage after a long pause without having to re-learn it,
- low error rate for user-performed actions,
- satisfaction addressing the pleasant use of the system.

ISO 9241-11 Guidance on Usability [7] provides a more general definition of usability as – "*the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*". Both definitions are quite similar covering the same usability characteristics. Current dissertation concentrates only on usability of Web User Interfaces (WUI) – a client-side (also called a front end) component of Web application formalised as a Web page which is accessible over the network and viewed using the Web browser [6].

Therefore, interfaces of native applications (software programs that have been developed for a particular platform) and other systems are out of scope of this thesis. The European Commission defines web usability in the EU Internet Handbook[1] as follows: „*Web usability is an approach to make websites easy to use for an end-user, without requiring her (or him) to undergo any specialized training. The user should be able to intuitively relate the actions he needs to perform on the web page with other interactions he sees in the general domain of life, e.g. press of a button leads to some action* ". The latter definition coincides with the objectives of the thesis the best. The goal of web usability is to present the information to the user on WUI in a clear and understandable way and provide the correct choices to them in a very obvious and unambiguous way.

Usability is considered to be a "critical condition for survival", its absence causes frustration and web application is likely to be rapidly abandoned [8]. It is found that the quality of navigation and how easy a web application can be used has an impact on how much information users read on the web page [20]. In fact, highly usable web applications influence positively on consumer satisfaction and increase their loyalty towards the company [21] encouraging visitors to visit the web application again [22]. In fact, the visitors' return rate for web applications with high usability rate is higher than for less usable applications. Web application usability is not a luxury but rather a basic determinant of productivity and of users' acceptance of web applications [19]. Moreover, usability is called the key to successfully conducting commercial Web site design [23]. In fact, usability is an essential attribute of every web application including web application of public sector organizations, banks, e-commerce applications and social network web applications as purpose of each web application is to provide information to users in a clear and understandable way. Of course, the purpose of public sector and e-commerce web applications are different; the latter is focusing on increasing the purchase rate that cannot be the case for public sector portals.

In addition to aforementioned characteristics (learnability, efficiency, memorability, low error rate and satisfaction) of usability, it is common to use usability in combination with accessibility. "*Web accessibility is an attribute through which people with disabilities can perceive, understand, navigate, and interact with the web, and they can contribute to the web*" (as defined by the World Wide Web Consortium[2]). In fact, usability considerably overlaps with accessibility when target users include people with a defined range of disabilities and context of use suggests accessibility observation such as assistive technologies [24]. Nevertheless, the requirements for people with disabilities are frequently not sufficiently considered in usability research and practice [24]. Accessibility contains both: general usability guidelines and guidelines that are more specific to people with disabilities. This thesis examines both usability and accessibility, as far as accessibility is often named as a subset of usability [25] than in thesis for incorporating various subsets of usability the generic term usability is used (usability guidelines also include accessibility guidelines in the context of dissertation).

In order to design WUI that is understandable, navigable, smooth, and easy to use and learn, it should follow various usability guidelines that guide designers to establish solutions understandable for the majority of users. Usability guideline is considered in the dissertation as an evaluable concrete rule, criterion or principle pertaining to the

[1] http://ec.europa.eu/ipg/design/usability/index_en.htm
[2] http://www.w3.org/WAI/intro/accessibility.php

evaluation of usability [8]. Usability guidelines are developed with the purpose to help WUI developers and designers to develop highly usable WUI.

Usability should not be confused with User Experience (UX). UX is a person's perceptions and responses that result from the use and/or anticipated use of a product, system or service [7]. Usability is a quality attribute measuring whether WUI is easy to use and learn, user-friendly, navigable. UX, in its turn, is wider concept also including emotional factors like the comfort of use and attractiveness. Overall, UX encompasses all possible aspects (including usability and its quality attributes) concerning the interaction between end-user and the company, its products, and services.

The research presented in this thesis is concentrating on web usability evaluation. Namely, the dissertation is concentrating on WUI usability aspects that can be automatically evaluated; usability aspects that require participation of potential users are not in the scope of the thesis.

2.2 Web Usability Guidelines

Current dissertation is focusing on inspection usability evaluation. In general, inspection usability evaluation is based on the evaluation of usability to usability heuristics, guidelines or specifications that contain the rules WUI should satisfy. That leads us to essential branch of studies about web usability – web usability guidelines. The definition, types and sources of web usability guidelines are discussed in this section.

An essential part of usability evaluation (including automated usability evaluation) is a set of guidelines against what WUI is evaluated. This dissertation considers usability guideline as an evaluable concrete rule, criterion or principle pertaining to the evaluation of usability. Usability guidelines should not be confused with usability heuristics. The latter defines broader rules and principles for usability evaluation. For instance, a heuristic could claim: *User interface should present elements simply*. In order to evaluate web application conformance to that heuristic UI should be evaluated according to a set of guidelines. In general, a heuristic is an abstract principle consisting of a set of specific guidelines.

Initially, usability guidelines are defined as a text in human readable way. Guidelines that are suitable for automated usability evaluation are then converted to machine-processable format (XML Schema is the most widely used method to define usability guidelines) [10].

Web Content Accessibility Guidelines (WCAG)[1] (developed by the Accessibility Guidelines Working Group (AG WG), which is part of the World Wide Web Consortium (W3C)) and Section 508 Standards for Electronic and Information Technology[2] (developed by U. S. General Services Administration Federal Government) are technical standards (each containing about 70 guidelines) providing guidelines that explain how to make web content more accessible to target users including people with disabilities. These standards were designed with two main purposes. Firstly, these standards provide a rich set of guidelines that make WUI more usable for people with disabilities, as these guidelines could not always be considered in common practices. Secondly, both standards contain general usability principles that have an impact on all potential users equally. WCAG guidelines are divided into three levels [A, AA, AAA]:

- *Level A* covers the most basic accessibility requirements.
- *Level AA* contains the most common guidelines for users with disabilities

[1] Web Content Accessibility Guidelines, http://www.w3.org/WAI/intro/wcag

[2] Section 508, https://www.section508.gov

- *Level AAA* is the highest and most complex level of accessibility containing very specific web accessibility guidelines.

In fact, WCAG and Section 508 standards contain similar and mostly overlapping accessibility guidelines. Moreover, Section 508 requires conformance to WCAG Level AA. WCAG is de-facto accessibility standard and it is ratified as an international standard being included in legislation of the European Union[1]. In the European Union, according to EUROPA - Web accessibility policy, all EU Commission websites should follow WCAG Level AA guidelines [26]. It is pointed out that following WCAG Level AAA guidelines are not mandatory as these guidelines are too strict and following all Level AAA guidelines may be extremely time-consuming for some content. For example, WCAG Level AAA Guideline 3.1.5 suggests providing spoken version of the text for each paragraph on the page and suggests providing a text summary that can be understandable by people with lower reading ability. Applying such guideline to existing content requires manual review of all text sections of all pages of WUI being a very resource-consuming task.

Interoperability Framework of the State Information System [27] defines a set of guidelines for preparing public sector Information Technology (IT) legal acts, designing IT solutions and organizing IT-related public procurements in Estonia. The framework contains requirements to user interfaces of public information systems created for residents of Estonia. One of the main statements it contains is that user interfaces of information systems must comply with WCAG 2.0 Level AA. Nevertheless, based on research conducted in the year 2015 less than 10% of public sector web pages in Estonia comply with WCAG standard guidelines [28].

Accessibility guidelines are only one subset of usability guidelines that cover the structure of HTML code and make the content of WUI understandable to the wider audience including people with disabilities. For example, accessibility guidelines require all image elements to have *alt* attribute that allows understanding the content if image fails to download. Also, *alt* attribute enables reading the page to visually impaired people who use screen readers to browse the page. The purpose of accessibility standards is not to make all WUI look similar way but to be more accessible to people with different needs. For example, WCAG says that all web applications should contain search functionality for searching the content of the whole application but it does not define where the search field should be located, what the optimal length of the search field is and how the search results should be presented. Thus, accessibility standards are focusing on the characteristics that could be standardised such as colour management, HTML semantics, link management, text alternatives, keyboard operability and focus management; they are not covering in details the guidelines for evaluating the visual characteristics of element on the screen, position of elements on the screen, distance between elements on the screen, usability of search elements, forms and inputs. That is the reason why many research works aim to establish additional usability guidelines covering the additional characteristics of WUI elements including images and links [29], the navigation and page layout [30], mobile usability guidelines [31] and search field usability [32].

Bringing web applications to mobile platform involves certain benefits and presents challenges and peculiarities in usability like accommodating less content on the screen because of the smaller size, optimizing the content on the page to be more usable for

[1] Directive (EU) 2016/2102 of the European Parliament and of the Council of 26 October 2016 on the accessibility of the websites and mobile applications of public sector bodies, http://eur-lex.europa.eu/legal-ontent/EN/TXT/?uri=CELEX%3A32016L2102

touch inputs as the errors for touch input are more probable than using mouse input. Fortunately, there are guidelines and recommendations for developing UI for mobile platforms [32]. Wessels, Purvis and Rahman in their paper [33] review the literature concerning usability on mobile devices and point out main mobile usability principles:

- Minimize and streamline – all not required complexity and excess content should be excluded from mobile page making mobile page striking a balance between load as fast as possible.
- Scalability – mobile WUI should be equally usable for mobile devices with various screen sizes; in fact, it should be the same usable if the device is switched to the landscape mode.
- Buttons and Hyperlinks – in common, finger is used as a pointer for mobile devices that is why clickable elements should be wide and high enough, so person could tap them with a small error rate.
- Consistency – the styles, logos, links and buttons of UI elements on mobile platform should be consistent with desktop styles [34].

Overall, current thesis concentrates on web usability guidelines (including accessibility guidelines) for desktop computers (including laptops) and smartphones. Current dissertation does not concentrate on tablet devices as they follow usability guidelines that cannot always be applied to smartphones. For instance, usability guideline encouraging developers to place radio buttons vertically on mobile devices because of the small screen size of smartphones is not valid for tablet devices that, mostly, having enough space to place radio buttons also horizontally.

Many companies design their own corporate usability guidelines obliging web applications within the organisation to follow similar style guidelines. That ensures that all applications share a common look and feel. Such guidelines establish consistency between applications encouraging developers to design highly usable web applications. Corporate usability guidelines focus on guidelines for presentation elements, such as visual design elements as colour, logos, fonts or icons; page or screen layouts including spacing, justification and common items; and the correct use of standard controls including buttons, drop-down selections, radio button or check boxes [35]. Corporate usability guidelines are widely used for back office web applications and well as for front office web applications where the UI design between different applications could be unified making it more usable for employees.

To sum up, web usability guidelines are the starting point for every research on automated usability evaluation as they are an integral part of every method or a tool addressing automated usability evaluation. Current dissertation is concentrating on web usability guidelines derived from the research works and analyse reports, platform-specific and company-specific usability guidelines. The main focus is set on web usability guidelines for desktop devices and smartphones.

2.3 Usability Evaluation Methods

Business requirements are a driving force for WUI development as WUI should always contain up to date information concerning services and values provided by the company. Business requirements, in the context of WUI development, are described as functional changes or sets of functionality that they should contain. In fact, business requirements to WUI could be controversial with usability guidelines. Simple example can be that business people very often ask to fit as much information on one page as possible assuming that such approach will make it easier for clients to find the information they

are searching for. In fact, such approach does not follow F-Shaped pattern of reading[1] and violates many usability guidelines that limit the length of scrolling, density of text.

In order to ensure that new features do not decrease usability of WUI usability evaluation should be performed. Usability evaluation is a method for identifying specific problems with usability of products [9]. Usability evaluation plays an important role in establishment of final design of WUI. It identifies usability problems matching the designed application and user needs. Thus, it allows detecting usability problems before the application is released to production where it is used by real users. Multiple research works on the influences of usability on overall customer satisfaction show that in case WUI has a high level of usability, users unintentionally increase the loyalty and trust towards the company [36] [37], and that usability contributes future revisits to the web application and majorly increases overall user satisfaction [22]. To achieve a high level of usability it is required to focus on the target users of WUI analysing their needs and patterns of behaviour. Potential WUI solution should be prototyped and validated iteratively to ensure that the final product will be highly usable by target users.

There are two main groups of methods to usability evaluation: inspection evaluation and empirical evaluation methods. The main difference is that inspection evaluation methods rely entirely on the expertise of usability expert (usability inspector), whereas empirical evaluation methods involve target users of WUI in the evaluation.

Empirical evaluation methods include interviews with potential target users [38], questionnaires [39], card sorting [40] and eye tracking [41] for usability evaluation. Empirical testing methods expose severe, recurring and global problems of navigation, information and content organisation as target users are involved in testing [42]. For example, in order to understand users' preferences, interviews are a good and effective approach in measuring users' satisfaction with the system [43]. Card sorting is an effective way to determine how logically and naturally the information, structure of navigation and layout are organised on WUI from the view of its end user. In general, empirical evaluation methods are effective in finding the problems of workflows and inefficient solutions in WUI. In general, empirical methods require the participation of usability experts for preparing the evaluation, conducting the evaluation itself and for analysing the collected data to provide the suggestions for improvements.

Another empirical method that observes the data that is not vocalised by user is eye tracking. Although eye tracking as a concept is not new to human-computer interaction (HCI) studies, the applicability of it and its precision has greatly improved with the development of technologies used. Eye tracking is widely used as an empirical research approach for mobile and desktop Web usability evaluation as an effective method for analysing UI [41] [44] [45]. Eye tracking measures the motion of the eye and where the eye is focused at the time participant views a web page. Seix et al. [41] in their study proved the efficiency and applicability of the method. They concluded that eye tracking as a methodology for assessing UI has numerous benefits such as accuracy and reliability in measuring, non-intrusiveness for user (as it records in a discrete way) and simplicity in preparing the set-up. Overall, the advantage of that approach is that it is possible to analyse the behaviour of users (where they are looking at, what parts of the interface they are missing etc.) without asking feedback from the users themselves.

[1] https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/

Another cutting edge approach to usability evaluation is all-around web analytics such as Google Analytics[1], Inspectlet[2] and Yandex Metrica[3]. In general, such tools for all-around web analytics provide the way to record users' sessions including user clicks, scrolling, mouse movements and keystrokes. The integration of these tools into web application is straightforward; a tiny piece of JavaScript code should be copied to the source code of WUI. Afterwards, stored data can be analysed by usability experts with the purpose to understand, how users are using web application, how much time do they require to perform certain actions on the web page and how users are navigating the mouse. An important advantage of that approach is that real users' data is stored without a need to conduct an experiment. For example, all-around web analytics (e.g. Google Analytics) can determine which field people are abandoning the form. Nevertheless, all-around web analytics does not prevent usability issues from being released but rather helps to find problems when WUI is already used in production, on the contrary, the problem current dissertation addresses – detecting usability defects immediately after the code of WUI has been modified.

Despite the fact that empirical evaluation methods are effective in finding usability defects, there are many obstacles preventing from applying these methods widely:

- Organising and conducting user tests is relatively expensive because it requires a high demand for human and time resources [46] [47].
- Small software companies do not always have the funds to pay for complete consultancy or involving usability specialist as they are expensive to hire [48].
- It is not always possible to increase the coverage of evaluated features by evaluating every single aspect of UI [46].

Empirical evaluation methods cannot be fully automated as real users should participate on evaluations and cannot be substituted by computers and it is impossible to simulate human-specific characteristics as human sense and personality as stated in [49]. The method for automated usability evaluation introduced in the dissertation does not substitute empirical evaluation methods (such as eye tracking, web analytics, user tests and questionnaires).

On the contrary, usability inspection evaluation methods are flexible to automated evaluation. Usability inspection evaluation methods such as heuristic evaluation and pluralistic walkthrough require the expertise of usability inspectors (usability designers and experts in usability evaluation) to detect usability problems in WUI design. Usability inspection methods include cognitive [50] and pluralistic [51] walkthrough and heuristic evaluation [52].

The cognitive walkthrough is a usability evaluation method when evaluators go through series of activities to fulfil and ask certain questions imagining that they are potential users of WUI [50]. The purpose of the method is to understand the learnability of web application for new or potential users. Currently, the cognitive walkthrough method is not used widely in scientific research works of usability because, in common, it uses superficial analysis of WUI focusing mostly on graphics and words on the screen. In practice, however, it is often used to understand the experience new users get while visiting WUI first time.

[1] https://www.google.com/analytics/

[2] https://www.inspectlet.com/

[3] https://metrica.yandex.com

The pluralistic usability walkthrough allows evaluating usability of WUI on the conceptual level when WUI initial mock-ups are ready. It is important to understand that usability problems detected in early prototyping phases are easy to fix then the problems detected on the later phases. The pluralistic usability walkthrough requires product developers, representation users, usability experts and other members of the product team to participate in the process of evaluation. The views of WUI (or WUI mock ups) are shown in the similar order as they appear to the user and the participants of evaluation are asked to emulate the role of the user. Afterwards, each screen is discussed within the participant and aspects to improve presented. The drawback of that method is that it does not guarantee the inspections of all available navigation paths on WUI [51].

Heuristic evaluation (HE), proposed by Nielsen and Molich [52], is widely used being an efficient low-cost inspection method for professional evaluation of WUI usability. With heuristic evaluation, usability experts look at WUI and come up with opinion what is good about that comparing it against guidelines and accepted usability principles. HE can contain different principles and guidelines for evaluating web applications of different categories (e.g. heuristics for governmental application [53] [54], educational portals [55] and health sector [56]). The output of HE is a list of possible usability issues. HE is being criticized for focusing on finding the errors, in contrast, empirical evaluation methods are concentrating on user satisfaction and effectiveness [57]. Nevertheless, HE is widely used for evaluating usability due to the low-cost and relatively easily implementable way to improve usability [58]. HE can be done at any time it is needed as it does not require preliminary planning [59]. As far as during HE WUI is checked against usability guidelines, then the evaluation can be performed by person who is not familiar with usability. In fact, usability guidelines can be converted to machine-processable format and evaluation can be executed by software programs. That is the area current dissertation concentrates on – performing evaluation of WUI to web usability guidelines automatically (without involvement of usability expert).

2.4 Automated Usability Evaluation

The main problem of manual usability evaluation methods including inspection and empirical evaluation methods is that every evaluation requires the participation of usability experts: in case of inspection evaluation usability expert is responsible for checking the conformance of WUI to the specifications and usability guidelines; in case of empirical evaluation methods usability expert is responsible for preparation of the evaluation and analysis of the collected data. Also, in case of manual evaluation, it is extremely time-consuming to perform the evaluation of all pages and every element of WUI (like link, button, search input, form inputs) with every new version of WUI. Aforementioned obstacles are the motivation for automated usability evaluation. Automated usability evaluation requires fewer resources to use as the configuration of the tool for automatic usability evaluation can be done once and, afterwards, evaluation can be performed multiple times without any preliminary configurations and involvement of human resources. Nevertheless, it is impossible to automate usability evaluation entirely, as there is a still needed to produce and update custom usability guidelines and check viability of usability guidelines.

According to [60], there is number of issues concerned with the use of automated tools: expandability and upgradeability, alignment with the latest technology and limited effectiveness of the reports.

Expandability and upgradeability are important issues that should be considered in automated usability evaluation [10] [14] [15] [61] as new usability guidelines appear by platform vendors and usability researchers, and the existing ones being updated. Some countries develop the lists of country-specific guidelines by extending WCAG standard with additional guidelines (e.g. US uses Section 508 accessibility standard based on WCAG). Also, companies define company-specific usability guidelines that are applicable only to the designed WUI. Such modification requires significant costs for resources. The issue of expandability has been challenged by many research works developing methods for machine-processable formalisation of custom usability guidelines [14] [15] [61].

Another common problem in tools for automated usability evaluation is limited effectiveness of the reports [62] [63]. Usability evaluation tools sometimes provide irrelevant information about the guidelines being violated. As a result, it cannot be straightforward to identify the main cause of the usability problem. It is important to consider that the reports can be analysed by people in different roles such as developers, product owners and usability experts. As a result, the report should provide sufficient information about detected usability issues including the reason for failure, suggestion how to fix the problem and how the problem can be reproduced.

Common requirement for usability evaluation is support for new technologies [18]. New languages for designing the style themes of a web application are being released introducing new features such as semantic tags that were introduced with the release of HTML5. The amount of dynamic content is increasingly growing; web pages are becoming more and more interactive. Turns out that some validators as pointed out in [18] are unable to efficiently evaluate the usability of web applications made with the most recent technologies like HTML5 and CSS3. Overall, it is an inevitable requirement to support recent technologies for tools suitable for automated usability evaluation.

Many researchers contribute to the development of automated usability evaluation tools [10] [11] [15]. Schiavone and Paterno [10] proposed Mauve – a tool for automated usability evaluation. This tool is capable of evaluating WUI conformance to WCAG accessibility guidelines of all three levels (containing around 80 different guidelines). In fact, the tool provides functionality for designing custom usability guidelines in addition to existing set of predefined guidelines. Dangli [15] developed a framework called USEFul for automating usability evaluation enabling a non-expert in the field of usability to conduct the evaluation. USEFul separates the definition of guidelines from the usability evaluation logic. Such approach allows adding, modifying and deleting the guidelines without altering the source code of the tool. The disadvantage of USEFul is a sophisticated way of adding guidelines. Gay and Li in [11] proposed an open source tool for automated usability evaluation called AChecker (contains around 100 guidelines) that allows checking the compliance of WUI to WCAG, Section 508 and BITV accessibility guidelines. The advantage of AChecker is that it provides easy to use WUI for triggering the evaluation process; the main disadvantage is that it does not allow defining custom usability guidelines.

Typically, the most advanced tools for automated usability evaluation such as Mauve [10], USEFul [16], and PowerMapper[1] support the evaluation of most WCAG guidelines out of the box. In general, they all have a possibility to define custom usability guidelines. The primary disadvantage of existing solutions is that it is not possible to evaluate visual

[1] https://www.powermapper.com/

aspects of a WUI including measuring the presence of scrolling, the layout of elements on the web page and the positions of elements on the screen; also, many other assessments like determining the distance, size and the contrast of elements on the screen are not possible with that approach. It is not possible because these solutions are based on parsing the HTML code and subsequent validation of HTML syntax against the guidelines and thereby ignoring rendered final representation of WUI that is accessed by end users from web browsers. There are certain attempts to develop tools that evaluate visual characteristics of WUI that concentrate on evaluating a particular visual aspect of WUI. For instance, Google Mobile Friendly Test[1] contains the test for evaluating size of tap targets; Wave[2] checks the contrast rate of elements. Nevertheless, there is no integrated solution that allows evaluating multiple characteristics of WUI rather than concentrating on one specific problem. This calls for adopting an integrated alternative method for evaluating various visual characteristics of final layout on the screen of WUI like the size of elements, distance between them, allowance and presence of scrolling. That is the problem current dissertation is researching – developing integrated method (discussed in Section 4.5 in more detail) for automated usability evaluation enabling both automated evaluation of HTML specific usability guidelines as well as evaluation WUI conformance to guidelines covering visual characteristics of WUI.

There are many other tools (for a list of different available tools for web accessibility evaluation one may refer to W3C tools list[3] of more than 80 tools) to conduct the task of automated usability evaluation (semi-)automatically and check whether the developed WUI meets accessibility guidelines. Tools that are based on WCAG or other accessibility standards do not consider corporate guidelines or specific guidelines set for a particular WUI. Moreover, existing tools for automated usability evaluation like Mauve [10], USEFul [16], AChecker [11], PowerMapper[4] are used on ready-made or pre-release solutions considered to be used during the testing phase of developed WUI but there are obstacles (e.g. these tools require special environment to run and they cannot be executed together with unit or functional tests) preventing them to be used during implementation phase of WUI development. Implementation phase of WUI development is code-centric and it incorporates activities directly connected to WUI development like writing UI code and covering code with unit tests. In its turn, testing phase of WUI is concentrating on WUI testing using various methods and tools that increase the efficiency and accuracy of evaluation such as user tests and tools for automatic usability evaluation. Aforementioned tools Mauve, Useful, AChecker and PowerMapper fall into the category of tools suitable for testing phase being process-centric rather than code-centric.

Unfortunately, it is not possible to evaluate every single usability guideline automatically because certain usability guidelines require the involvement of usability expert intelligence and common sense. For instance, the next guideline *Change the font characteristics to emphasize the importance of a word or short phrase* cannot be evaluated automatically as it is extremely complex and practically impossible to make software understand that the more important phrase has been emphasized and the less important is not (unless this data is somehow included in the HTML, e.g. with custom tags indicating the importance of a phrase).

[1] https://search.google.com/test/mobile-friendly
[2] https://wave.webaim.org
[3] https://www.w3.org/WAI/ER/tools/
[4] https://www.powermapper.com/

Overall, the most advanced solutions for automated usability evaluation have two common disadvantages. First, existing tools are suitable only for testing phase of WUI development; it is practically impossible to use these solutions during the implementation phase with the purpose to get feedback immediately after WUI modification [13]. Second, existing solutions do not allow evaluating various visual characteristics of finally rendered layout on the screen of WUI like the size of elements, distance between them, allowance and presence of scrolling [14].

Usability evaluation during implementation phase and automated usability evaluation of visual usability guidelines are the problems that current thesis addresses by presenting a special tool called *Guideliner.* The *Guideliner* is code-centric tool for automated usability evaluation during the implementation phase of WUI development. It is beneficial to conduct usability evaluation of modified code during the implementation phase of WUI development as it identifies issues and problems that are out of WUI developer scope. Another problem challenged by the *Guideliner* is automated evaluation of visual usability guidelines of finally rendered WUI that increases the coverage and the number of usability problems handled by automated usability evaluation.

Both aforementioned problems are addressed in this thesis: Chapter 4 is devoted to the development of the method capable of evaluating visual characteristics of web UI, whereas Chapter 5 is addressing the problem of automatic usability evaluation during the implementation stage of UI development process, using the proposed tool for automated web usability evaluation.

2.5 Related Works

The research in the area of web usability, web usability evaluation and automated web usability evaluation is concentrated on the next research directions: improving existing usability guidelines and heuristics [53] [64] [65], interaction-based [66] [67] [68] and metric-based [10] [15] automated usability evaluation. The next sections discuss main topics that are connected to the research presented in this thesis.

2.5.1 Research in the Field of Usability Guidelines

Usability guidelines help designers to create WUI that is equally understandable and clear for different groups of users. Research in the field of user interface usability covers all possible types of UI that people are interacting with. For example, Koppel et al. in their research [69] defined usability guidelines for DVD menus because DVD menus often miss out of usability. In its turn, Coelho et al. concentrated on guidelines for TV applications [70], and Moshnyaga [71] proposed guidelines for smart doors and smart systems of monitoring. Current thesis is concentrating only on WUI usability guidelines that is why the discussion on usability guidelines for other types of WUI is very limited herein.

The definition of usability guidelines is not an ordinary research activity as usability guidelines cannot prove effective if they are based on too few findings or the subjectivity influences the process of guideline formalisation. Most of the existing usability guidelines are based on the results of usability testing and the evaluation feedback retrieved from the users [72]. To make the process of usability guideline definition more effective Evan in [73] proposed an approach for formalisation of usability guidelines. The approach covers different steps of the usability guideline development including setting the design question that guideline solves, searching for evidence, systematic assessment of results and testing the resulting guidelines.

Hussain and Ferneley in [74] point out that heuristics are often developed as general design principles. As a result, the interpretation of heuristics can be different between designers. To make usability guidelines unambiguous, they applied Goal Question Metric (GQM) for development of usability guidelines with clearly defined metrics. In the context of their research the metric is an evaluable characteristic of guidelines such as task completion time, the size of the elements, colour of elements etc. Their research proved that including metrics to designed usability guidelines makes the guidelines equally understandable to designers.

Following usability guidelines makes WUI more usable for the users. The research conducted by Conti and Sobiesk in [75] studies opposite situation – deliberately constructed WUIs that violate design principles and usability guidelines to accomplish business goals and acquisition of revenue (selling a product or service, gathering personal information from the user that is not needed for business, increasing company recognition). The examples of such violations are animations and blinking objects or sounds to attract the attention, forcing users to view the content that they did not want to see and navigations that are designed deliberately to force user visit as many pages as possible. Users use different measures to protect from such WUI (e.g. pop-up blockers, text-only browsers, ad-blocking software, and browser plug-ins), nevertheless, study shows that the effectiveness of these measures is only marginal.

In general, following WCAG and Section 508 accessibility guidelines (each containing about 70 guidelines) makes WUI's more usable for all types of users. Nevertheless, Section 508 and WCAG are technical standards which do not cover all aspects of usability. Both standards provide accessibility guidelines to make the content of web applications accessible to a wider range of people including people with disabilities. To a lesser extent, the standards are covering other categories of guidelines like the ease of comprehension, navigation scheme and features, consistency of the context, screen based controls and others. The limitations of WCAG and Section 508 encourage researchers to cover aspects of usability ignored by the standards.

Platform specific (e.g. desktop or mobile) [64], UI element specific (e.g. link, button) [29], web application type specific (e.g. news web application) [53] and target users specific [65] usability guidelines have been explored by many authors, defining usability guidelines and usability heuristics for various areas of use and target audience including usability guidelines for teenagers [65] [76], mobile application specific heuristics [64] [77], heuristics for usability evaluation of government applications [53] [54], educational portals [55], and heuristics for the evaluation of captchas on smartphones [77]. All aforementioned research works contribute to the development of usability guidelines and heuristics. The results of these research works can be used for defining category-specific usability guidelines suitable for automated usability evaluation.

One of the widely used lists of usability guidelines is combined by Shneiderman and Leavitt in [78]. The collection of guidelines contains more than 100 research-based web design and usability guidelines. It is worth noting, that these guidelines are not established by the authors but are filtered out while analysing usability research works and usability evaluation reports. The compilation of usability guidelines helps to avoid potholes and swamps while designing highly usable WUI.

Other cutting edge collections of usability guidelines are produced by Schade and Nielsen in [79]. They are mostly focusing on e-commerce usability guidelines with the purpose to increase the rate of purchase and improve the online shopping experience making the process of online selling smoother and more effective. In fact, these

guidelines cannot be applied to the wide range of WUI as the purpose of public section web applications is to clearly outline the information about the governmental services rather than earn money.

The rapid development and evolution of mobile devices bring additional requirements to WUI because of the certain limitations of mobile platforms such as smaller screen size, finger as a typing and pointing device and additional features like camera and accelerometer [33] [80]. Alsalamen and Shahin in their research [81] demonstrated that users make more errors when filling in the forms on the devices with smaller screens emphasizing that usability guidelines suitable for devices with big screens cannot be always applied to the devices with the smaller screen. For instance, mobile web usability guideline presented by Baymard web usability research institute[1] says – *Place labels above the input fields in forms*. The main issue with left-aligned input field labels that are common for desktop web applications relates to the smaller size of the display and different aspect ratio in the case of mobile devices.

In [59] Gómez et al. presented the set of heuristic evaluation adopted for mobile WUI enriching the existing desktop heuristics with the support for mobile devices. They presented the evaluation checklist that has been readapted to mobile WUI overcoming mobile-specific limitations like amount of inputs/outputs, limited processing capabilities. The main target users of the checklist are WUI developers, product owners, analysts without competency in usability evaluation. Salgado and Freire [82] performed the systematic mapping of literature regarding the use of heuristic evaluation methods applied on mobile web applications. They concluded that the topic of mobile usability heuristics has been a relevant topic, and the interest since 2010 is becoming bigger in recent years. Many studies are continuously appearing in the literature developing new usability heuristics for evaluating usability of mobile web applications including heuristics defined in [83] [84] [85].

Alsalamen and Shanin in [81] performed the comparative research of usability of forms on devices with small and big screens. The set of form usability guidelines has been proposed based on their finding. For instance, their research showed that presenting error messages above the input field is not recommended on the devices with a small screen as this can be dismissed by users.

2.5.2 Research on Automated Usability Evaluation

The development of automatic usability evaluation tools is an attractive area of research in the era of multi-platform devices as it allows performing usability evaluation without the involvement of usability experts. There are many factors in favour of automated evaluation of usability such as that manual usability evaluation methods are time-consuming [47] [86] and resource-intensive [47] [86]; it is not possible to cover every section of WUI with manual testing [46]; it can be complicated to recruit usability test participants who represent the real target user group [47] [86]. Automated solutions, in turn, detect usability problems at a lower cost, and can cover more aspect of WUI during one evaluation. Nevertheless, available tools for automated usability evaluation are concentrating mostly on accessibility and HTML-centric guidelines leaving evaluation of visual characteristics of WUI out of the scope [12].

There are multiple approaches to automated usability evaluation, and currently, most of them could be suited one of the three main groups: interaction-based usability evaluation [66] [67], metric-based usability evaluation [87] [88] and model-based

[1] https://baymard.com/blog/mobile-form-usability-label-position

usability evaluation [87] [89]. The research presented in this thesis falls into the metric-based usability evaluation.

Interaction-based usability evaluation requires the interaction data to be recorded within WUI being evaluated. Interaction-based usability evaluation can be used during the testing phase of WUI development and after WUI has been deployed to production environment. Various data can be recorded like mouse cursor movements, keyboard input characteristics and the interaction data such as clicks on links, buttons, and images. Afterwards, the recorded interaction data is compared to the expected interaction data suggested by usability expert to identify potential usability issues [68].

Many research works are concentrating on developing tools for automated interaction-based usability evaluation. Studies in the field of usability evaluation, coming from the interactions between the user and WUI, are divided into two major groups – studies concentrating on a solution for gathering of data from users [66] [67] [68] and the studies that in addition to gathering the data, are suggesting possible solutions to the detected usability issues [90] [91] [92].

Grigera, Garrido and Rivero [68] presented a tool called Bad Smells Finder for automating the gathering of interaction data from real users. The purpose of the tool is to automate the evaluation of interactive usability problems such as inefficient navigation issues, unnecessary bulk actions and misleading or misused widgets. Based on the further client side and server side collected data processing, they introduced an approach to finding interactive usability problems automatically. In order to collect the data, a tool script should be included in the header of the web page. The tool can detect 12 different kinds of usability problems including detection of unnecessary bulk actions and form validation problems.

Dixit and Padmadas [91] presented a recommender system which improves the user's experience of WUI based concerning learnability and navigability. The proposed system provides links offering users an alternative navigation path as a suggestion. The recommendations are provided on the analysis of gathered data of users' navigation behaviour between web pages. Such approach helps to find out the required information in the web application more easily providing the most efficient path for the user. Users can ignore the suggestion and follow the navigation according to their own understanding. It should be considered that recommender systems require big amount of interaction data to be collected in order to make a conclusion on the navigation paths users select to perform certain activity. Also, the results provided by the system should be initially reviewed by usability experts in order to make sure that the recommended paths are valid in the use case evaluated. Overall, it is a promising research direction as it allows optimizing the navigation but it may be complicated to find the problems on testing environments because of insufficient amount of data.

Au *et al*. [90] added additional listeners to the code of WUI recording various events (e.g. scrolls, click and drags) as they happen. Afterwards, the tool evaluates the usability of WUI components (e.g. form or navigation) comparing, how the developer expected WUI component to be used with the recorded results. Then, it highlights the differences between actual and the expected use of WUI component. Based on the collected real interaction data, WUI developers can optimise and make WUI components more understandable and usable for users. The main outcome of the research is the tool that is capable of evaluating user inputs in forms and presents the results in forms. The main drawback of the approach is that it is limited to finding the problems only in one domain – usability issues in forms.

In addition to finding usability problems, more advanced tools provide suggestions, how to improve usability of WUI. Gonçalves et al. [92] presented MOBILICS – task-based evaluation tool allowing the detection of usability problems during the execution of tasks and pointing out the interface elements that cause usability issues. MOBILICS collects data about users' interaction with WUI and then compares the sequence of recorded task with the expected task path. If there are deviations between the expected and the actual interactions, then MOBILICS' provides detailed recommendation on how to fix discovered usability problems.

In general, tools for interaction-based usability evaluation cover only narrow issues over a certain section of WUI like input or navigation issues. The reason for that is that every usability problem connected to user interaction requires specific data to be collected and then analysed. For instance, in order to find out which components on the web form are the most unclear for users, special listeners should be developed measuring the clicks on the components under the study. Afterwards, automatic data analyser is created to find critical problems in the recorded interaction data. In general, the results of interaction based usability evaluation should be reviewed by usability experts as they may suffer from subjectivity due to the configuration of the tool.

One of the paramount disadvantages of interaction-based usability evaluation pointed out by Bakaev et al. in [87] is that usability can be evaluated only after WUI is already in use by real consumers. The primary reason for that is obvious: all interaction-based usability evaluation tools require plenty of captured interactive data to detect usability issues with high probability. That means the first usability issues are found after real customers are made to interact with the UI, having potential critical usability issues causing the damage to the reputation of web application and the whole organisation. Of course, it is also possible to collect the interaction data on the testing environment with the potential users but that is very resource-consuming to involve potential users with every release of WUI. Other problems of interaction-based usability evaluation pointed out by Speicher et al. in [93] is that it is hard to obtain reliable quantitative data with any of existing interaction-based tools as quantitative data does not reflect all possible use cases.

Metric-based usability evaluation is based on the metrics that define its usability. It is based on the automated evaluation of WUI compliance to usability guidelines covering various usability aspects like visual consistency and complexity of WUI [87] including accessibility, readability, navigation, layout consistency, content organisation [88]. Usability level of WUI and the potential usability issues can be detected based on certain metrics, and metric-based usability evaluation tools demonstrate a good correlation with human annotators' results [16]. The metric in the context of metric-based usability evaluation is an evaluable characteristic (e.g. length of links, order of elements, value of attributes etc.) defined in concrete units of measurement (e.g. *alt* attribute of *img* tag should be defined). Nevertheless, metric-based usability evaluation is limited by evaluating only the complexity and consistency of a web pages; evaluating the usability of the workflow or navigation is not possible with that approach as such evaluation requires the participation of real users in the process of evaluation. All aforementioned problems cannot be evaluated by any metric-based based usability evaluation including the solution delivered in the dissertation. Current dissertation is covering metric-based usability evaluation.

The metric-based tools for automated evaluation of WUI conformance to usability guidelines are divided into two groups: tools that separate guideline definition from the

evaluation providing a way to define new guidelines [10] [15] and tools where the definition of a guideline is part of evaluation logic excluding the possibility to define additional guidelines in an easy way [11] [94]. The latter approach is used more often due to simplicity in implementation. Unfortunately, in that case, updating or adding new guidelines cannot be done easily. Thus, another critical requirement to the tools for automatic evaluation of web application usability is extendibility of predefined usability guidelines with custom application specific usability guidelines. This is a vital requirement as the existing usability guidelines could change or new one appears (e.g. with the emergence of new devices like tablet computers or with the evolution of new technologies such as HTML 5). In fact, it is unlikely that the tool contains usability guidelines suitable for each web application. For example, e-commerce usability guidelines presented in e-commerce UX report [79] require to embrace large product images showing more details and multiple views of the product. For instance, e-commerce usability guidelines may not be applicable to the governmental health web application for retrieving prescription information. In general, if there is no way to keep usability guidelines up-to-date than the tool soon becomes obsolete.

One of the first attempts to propose the language for defining usability guidelines was done by Beirekdar and Vanderdonckt in [95]. They proposed to separate the editing and definition of guidelines from the evaluation logic. Afterwards, multiple approaches were tried out for developing the most effective and usable approach for defining the guidelines such as using XPath sentences to implement the guidelines as Takata, Nakamura and Seki proposed in [96] and XML schema-based languages designed by Leporini, Paterno and Scorcia in [40]. However, XPath based approaches have been abandoned as it had a strict drawback of using only XPath language limiting evaluation to find only simple problems in HTML code without possibility to create usability guidelines covering multiple WUI elements and visual characteristic of WUI. The successor of XPath is XML Schema-based language incorporating usability guidelines in XML. XML-based approaches allowed defining more sophisticated guidelines; in fact, they allow defining most WCAG accessibility guidelines. Nevertheless, these languages including the XML-based languages defined by Schiavone and Paterno in [10] and another attempt done by Arrue, Vigo and Abascal in [61] are limited mostly to evaluating the structure of tags, attributes and their relations; defining visual aspects of web UI is not possible with that language. That is not the problem of the language but rather the problem of XML-specific problems and restrictions. XML-based language becomes sophisticated and unclear when multiple WUI element relations and elements specific attributes are incorporated into the language. These deficiencies are particularly notable when necessary to define complex usability guidelines like evaluating the visual position of elements on the screen, positions and distances between the elements on the WUI. That calls for adopting alternative approach to definition of usability guidelines overcoming existing limitations. In Section 3.4 the author proposes a usability ontology as a successor to the XML-schema based languages.

Schiavone and Paterno [10] proposed a software environment called MAUVE for automatic accessibility evaluation specifying a high-level language for the definition of guidelines. The abstract language called Language for Web Guideline Definition (LWGD) as proposed stated to enable simple and flexible formalisation of guidelines. The language is not tied to a certain set of guidelines providing possibility to define various custom usability guidelines. LWGD is based on XML schema providing the ability to validate custom usability guidelines before performing the validation. The advantage

of their tool is that it can express and evaluate more complex conditions, regarding multiple objects, in comparison with any other similar alternative solution. Nevertheless, proposed language become hardly extendible as it contains multiple various guidelines defining the relations and properties of elements. Also, it lacks the structures for defining usability guidelines covering visual characteristics of WUI.

Dangli [15] developed the framework called USEFul for automating the usability evaluation enabling a non-expert in the field of usability to conduct the evaluation. His solution allows adding, modifying and deleting the guidelines without altering the code that references those separating guidelines and the code into separate components. He proposed a custom approach to the definition of guidelines based on the guideline definition table where each row represents the guideline. The solution is capable of finding problems in HTML code; identifying visual usability issues is not possible with that approach. The main drawback of the approach is that it is based on the parsing of HTML code. Such approach cannot evaluate visual aspects of the web application. Another disadvantage of the proposed solution is the sophisticated way of adding new usability guidelines using custom guideline definition table.

There are also multiple other alternatives to Mauve and USEFul. For example, Gay and Li in [11] proposed an open source tool for automated usability evaluation called AChecker that allows checking the compliance of WUI to WCAG and Section 508 accessibility guidelines. It also allows defining custom usability guidelines. There are certain commercial tools for automated usability evaluation such as PowerMapper[1] that in addition to WCAG accessibility guidelines also contain some HTML-specific usability guidelines. Moreover, it contains search optimization guidelines checking that web application is indexed by search engine properly. Both tools provide API for performing the evaluation of WUI based on the URL of web application. Also, there are several tools that can evaluate certain visual aspect of WUI with some big limitations. For example, Google Mobile Friendly Test[2] performs a sanity check of WUI checking if WUI is compatible with mobile devices. The tool checks the WUI conformance to six very basic mobile usability guidelines including five HTML-specific guidelines such as the usage of Flash, font size, Viewport configuration (three guidelines) and one guideline checking such visual characteristic of WUI as the size of tap targets. The purpose of the service is to perform initial test and to give feedback if WUI is compatible with mobile devices or not. Another freeware tool called Wave[3] also proposes additional value to WCAG guidelines by checking the contrast rate of elements. In fact, contrast guidelines are part of WCAG but very few accessibility tools are capable of evaluating the contrast rate of elements. Nevertheless, the disadvantage of WAVE is that there is no possibility to define custom usability guidelines in this tool.

Model-based usability evaluation is overcoming the drawbacks of metric-based usability evaluation (metric-based evaluation does not take into account the influence of user tasks and use case context on the metrics value [87]) evaluation detecting usability problems coming from interactions with WUI. Model-based tools contain task model and the context of using models (user, platform, and environment) that are constructed within model-driven interface development paradigm [89]. This approach allows introduction of additional information in models to allow a designer to tune easily the interaction technique and simulate the behaviour of the user [97]. For example,

[1] https://www.powermapper.com

[2] https://search.google.com/test/mobile-friendly

[3] http://wave.webaim.org

Humayoun and Dubinsky in [98] applied model-based usability evaluation for automated testing of user gestures. Namely, they presented a task-based method that emulated user gestures and asserts the behaviour of WUI to the gestures. The main drawback of model-based usability evaluation is that the modeling and computational complexity are still quite high and adjusting the models to changing trends in usability engineering may be costly [87].

At the current state of technological development when the artificial intelligence (AI) becomes widely used area, AI methods were adopted in order to automatically evaluate usability guidelines. Guidelines that cannot be evaluated by any existing tool because of ambiguous metrics could be evaluated by AI methods [16]. For instance, the next guideline *The headings that are used should be unique from one another and conceptually related to the content* cannot be evaluated automatically by any evaluation tool. Dingli and Cassar in [16] filled the gap proposing the solution based on the mathematic algorithms that are capable of evaluating that guideline. In their tool, they implemented only three guidelines mostly because evaluating every sophisticated guideline requires various custom solution and configurations. Liyanage and Vidanage in [99] applied machine learning algorithms for automatic evaluation of usability guideline that checks the meaningfulness of link labels. Aforementioned guideline was selected because of its uniqueness and difficulty to implement. Their research results proved that it is possible to automate the evaluation of this guideline. Nevertheless, the authors point out that their algorithm should be enhanced as the results of the evaluation were correct only in 70% of cases. In general, AI methods are mostly focusing on the narrow issues that cannot be covered by interaction or metric based usability evaluation that is the reason why there is no AI based mainstream tools for automated usability evaluation.

The evaluation of usability without the involvement of users is a challenge for the researchers as it may make the evaluation cheaper in terms of time and money. Oliveira et al. challenged aforementioned problem presenting the predictive usability framework. The main advantage of their solution is that based on the analysis of previous evaluations their model performs evaluation of WUI quality automatically. The limitation of their solution is that it requires a lot of usability evaluation data to train the solution; also the accuracy of usability defects prediction is very low at the current framework development stage.

There are studies concentrating on evaluation of usability of certain category web applications like homestay websites [100], public self-service applications and self-service providers and newspaper website [101] using heuristics evaluation. Isa, Yusoff and Nording [100] evaluated the usability of homestay websites in Malaysia using various automated tools such as Web Page Analyzer[1] and Dead Link Checker[2]. A study concentrating on evaluating the usability of newspaper websites presented by Abdullah and Wei in [101] found out that the main usability problems of newspaper websites are too large advertisement areas and start pages containing a lot of links rather than useful information.

Dungli and Cassar [16] claim that model-based and interaction-based approaches could suffer from subjectivity of results when the problems found during automated testing cannot be easily fixed for all users. On the contrary, metric-based automated

[1] http://www.websiteoptimization.com/services/analyze/

[2] https://www.deadlinkchecker.com/

usability evaluation provides precise results as the evaluation is performed based on known usability guidelines.

The inevitable outcome of every usability evaluation including both manual usability evaluation and automated usability evaluation is the report provided as a result of the evaluation. Usability report is an important aspect of every usability evaluation as properly structured report clearly outlines the elements violating guidelines, reason of violation and probable cause of the problem making it easier for developers to understand the cause and fix the problem. In general, reports contain the description of guidelines that have been checked, an overview of evaluation results pointing out the guidelines being violated, description of failures and other information relevant to usability evaluation. According to Schiavone and Paternò [10], tools for automatic usability evaluation can be additionally divided into the next classes based on the way they show the validation results: code-oriented approach, graphical approach and separate application approach. The importance of this division is that the main output target users of tools for automated usability evaluation receive are the validation results. Results should clearly outline found defects and the elements that violate the guidelines. According to the study conducted by Schiavone and Paterno in [10], most of the tools fall into the code-oriented approach, highlighting detected errors in HTML code. An advantage of such method is that WUI developers can easily identify the line in code that caused the problem. The disadvantage of this approach that in many cases finding the row of the code responsible for certain visual characteristics is practically impossible as the visual inconsistency can be caused by multiple violations at the same time. Graphical approach solved aforementioned problem rendering the Web page before the reporting usability problems and locating the deviations in UI page through placeholders, tooltips, labels and other graphical elements. Nevertheless, Dingli and Cassar in [16] point out that such approach does not allow the clear identification of the error type making it less attractive for developers. In its turn, separate application approach overcomes the limitation of both previous approaches as stated by Dingli and Mifsud in [15]. In general, separate application contains the description of failed usability guideline, the reason for failure and the suggestions on how to fix detected usability defect [15].

Yusop and Vasa [62] performed the analysis of 147 different usability reports of different tools and showed that developers and testers often provide the observed result, expected results and steps to reproduce when describing usability defects. The main outcome of their work is that they found the information most required by the software developers was the least provided by reporters such as the cause of the problem and insufficient information in steps. Friess in the research presented in [102] came to the same conclusion that many findings in usability evaluations are not reflected in reports. For that reason, the consistency and sufficiency of the data in evaluation results are highly variable.

Følstad et al. [63] performed the survey within usability experts on reporting usability issues during usability evaluation and found that only 4% of automated usability evaluation results are described according to the format defined in standards or literature. The remaining 96% of respondents reported they developed their own format for reporting usability issue.

One possible format for reporting usability problems is to use Evaluation and Report Language[1] (EARL). EARL is a machine-readable format for expressing results for usability

[1] https://www.w3.org/TR/EARL10/

evaluation based on the designed vocabulary for expressing test results. It is intended for developers of Web accessibility evaluation and validation tools simplifying the exchange of test results between evaluation tools in an environment independent way. The advantage of EARL is that it facilitates processing of evaluation results providing the unified format of presenting usability evaluation results; it is designed to support exchanging data between software tools, querying and analysing test reports, evaluating dynamic and multilingual websites and others. The disadvantage of EARL is that it strictly defines the format of the report without the possibility to extend it with additional information such as the screenshot of element violating usability guidelines or the type of element under evaluation.

Overall, multiple research works in the field of usability evaluation are concentrating on reporting of new usability guidelines. In its turn, the majority of research works in the field of automated usability evaluation are divided into three branches: interaction-based, metric-based and model-based usability evaluation.

2.6 Chapter Summary

This chapter presented required overview of research activities connected to the topics discussed in the dissertation. The chapter consists of two parts, first part provided an overview of usability and accessibility, usability evaluation covering empirical and inspection evaluation methods, automated usability evaluation including inspection-based, metric-based and model-based approaches. The second part focused on the state-of-the-art related works related to the current study.

Automated usability evaluation is a very popular and perspective area of research as growth of web applications and various devices for browsing the Web increases the need for automated usability evaluation. Interaction-based, metrics-based and model-based methods addressing automated usability evaluation have been introduced. In fact, the research achievements (e.g. established usability guidelines, methods for manual and automated usability evaluation) in this field have paramount importance for designing highly usable WUIs. Still, there is no complete solution that addresses the problem of automated evaluation of visual usability guidelines of finally rendered WUI and automatic evaluation of WUI usability during the implementation phase of WUI development enabling feedback to developers immediately after modification. That calls researchers to elaborate on new methods for automated usability evaluation.

3 Usability Guidelines for Automated UI Evaluation

A solid foundation for manual and automatic metric based usability evaluation is usability guidelines that contain well-defined evaluation metrics and detailed description of expected state of WUI elements and their properties. Multiple methods were used over time for defining usability guidelines for automated web usability evaluation. The most successful was XML Schema-based methods [10]. Regardless of this, however, existing methods (XML Schema and definition table based method) are concentrating on HTML-centric accessibility guidelines ignoring the usability guidelines covering visual characteristics of WUI [12].

In this chapter, the thesis addresses this problem proposing usability ontology for storing usability domain knowledge and for defining custom usability guidelines capable of defining HTML-centric accessibility guidelines as well as usability guidelines covering visual characteristics of WUI. The chapter focuses on main sources of usability guidelines, principles, standards, and also on the methods used for formalising usability guidelines, and the design of ontology for capturing knowledge of web usability domain for WUI evaluation is discussed.

3.1 Introduction

Usability guideline is a very broad term including various disciplines, devices, WUI segments, and approaches to the evaluation. Usability guidelines are developed to simplify usability evaluation by helping to check that WUI satisfies certain guidelines and criterions. Usability guideline is considered in the dissertation as an evaluable concrete rule, criterion or principle pertaining to the evaluation of usability [8].

General usability guidelines applicable to a wide range of WUI are provided by standards, WUI research works, surveys, and usability related scientific articles. Specific usability guidelines, in turn, are developed by companies, platform development teams and researches presented in scientific papers addressing the research of web usability guidelines in the specific areas such as for example news portals, government, and banking sector.

Usability guidelines are divided into two main categories: guidelines that can be evaluated only manually and guidelines that can be evaluated both manually and automatically without the involvement of human evaluator. Current thesis is concentrating only on web usability guidelines that can be evaluated automatically. Usability guidelines suitable for automated and manual evaluation, in turn, can be divided into common usability and accessibility guidelines (applicable both to mobile and desktop devices), guidelines designed especially for mobile and guidelines specific for desktop devices only (Figure 1).

Common usability guidelines should be followed by most WUI. Application specific usability guidelines including company (e.g. guidelines specific to Google material design), platform (e.g. iOS guidelines) and application type specific usability guidelines (usability guidelines suitable for e-commerce) are tied to a certain application type, platform or a use case.

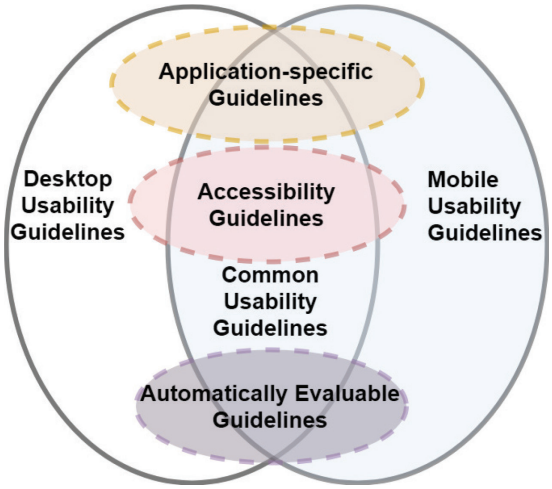


Figure 1. Categorization of usability guidelines suitable for evaluation.

In theory, usability guidelines can be categorized with more precision pointing out the guidelines for specific devices or the screen resolution specific guidelines. In practice, however, desktop and mobile platform usability guidelines constitute separate categories as mobile devices have more constraints than desktop devices such as smaller screen, single window accessible at one time and shorter user sessions. An important concern about usability guidelines is that every single usability guideline is not applicable to all kind of devices. Desktop device specific web usability guidelines are not always applicable to the mobile web applications and vice versa.

In its turn, usability guidelines specific to tablet devices (portable PC with a mobile operating system and touchscreen) can be moved into the category of mobile usability guidelines or can establish their own category. Certain mobile device specific usability guidelines are not applicable to tablet devices because of the bigger screen size of tablet device. For instance, usability guideline encouraging developers to place radio buttons vertically on mobile devices because of the small screen size of smartphones is not valid for tablet devices because tablet devices have bigger screens and placing radio buttons horizontally will not affect usability of WUI.

Guidelines suitable for automated evaluation should be precise and contain concrete evaluation metrics that include the characteristic and the value of characteristic being evaluated. For example, usability guideline presented in Appendix 1 (Example 3) is suitable for automated usability evaluation as it accurately declares that the width of the link is a characteristic being evaluated and 48 CSS pixels is the value of characteristic.

Nevertheless, not every web usability guideline is suitable for automated evaluation. For instance, the next guideline *A typical first-time visitor can do the most common tasks without assistance* derived from Userfocus Usability List[1] cannot be accurately evaluated automatically as there is no measurable characteristic.

In general, usability guidelines suitable for automated usability evaluation are inevitable part of every tool for automated usability evaluation. Tools are divided into two types: tools that have guidelines predefined (guidelines that are part of the tool and

[1] https://www.userfocus.co.uk/resources/taskchecklist.html

that can be checked without preliminary configurations) and tools that have customizable guidelines.

It is a limitation if a tool has only predefined guidelines and there is no ability to define custom usability guidelines. There are different possibilities how to describe usability guidelines. One of them is XML-based representation of guideline declarations. In general, such XML-based descriptions are delivered as a part of tools focusing on the automated usability evaluation.

WUI development technologies are now experiencing very rapid progress: HTML 5 has been released as a successor of HTML 4, new WUI development frameworks have appeared such as Angular[1], Vue.js[2], React[3]. Moreover, new techniques appearing allow more opportunities for WUI testing (such as PhantomJS[4] and Selenium[5]). They provide methods for fetching the page, locating elements on the fetched pages, clicking elements, filling inputs, moving between windows and others according to pre-programmed scenario. Some of them can be reused also for automated usability evaluation, especially for evaluating visual characteristics of WUI over DOM. Current research demonstrates, how latest achievement in the area of WUI testing can be reused for automated usability evaluation. Namely, this dissertation shows how to apply Selenium for automated usability evaluation. For example, Selenium provides API for evaluating finally rendered WUI including the evaluation of visual WUI characteristics. With this development, the methods for defining usability guidelines should also be reviewed as none of existing approaches allows defining usability guidelines related to visual characteristics evaluation of WUI.

The main contributions to the community delivered herein is a method for defining custom usability guidelines based on an ontology storing usability domain knowledge.

The rest of this chapter is organized as follows, Section 3.2 discusses the sources of usability guidelines, while Section 3.3 explores existing methods to the definition of usability guidelines. In Section 3.4, the ontology design for capturing knowledge of web usability domain for WUI evaluation is introduced, and, finally, Section 3.5 presents the summary of the chapter.

3.2 Sources of Usability Guidelines

Usability guidelines are developed with the purpose to help WUI developers and designers to develop highly usable WUI. For WUI evaluation, usability guidelines are derived from standards, HCI research and company-specific style guides.

3.2.1 Accessibility Standards

Accessibility is often named as a subset of usability [25] because it is focusing on making WUI more understandable and perceivable including for people with disabilities. The primary source of accessibility guidelines is accessibility standards. One of such standards is Web Content Accessibility Guidelines (WCAG) [103] – a stable, referenceable technical standard (containing about 70 guidelines) developed in collaboration with organisations and individuals all over the world, with a purpose of providing a shared international standard for web content accessibility that meets the need of governments,

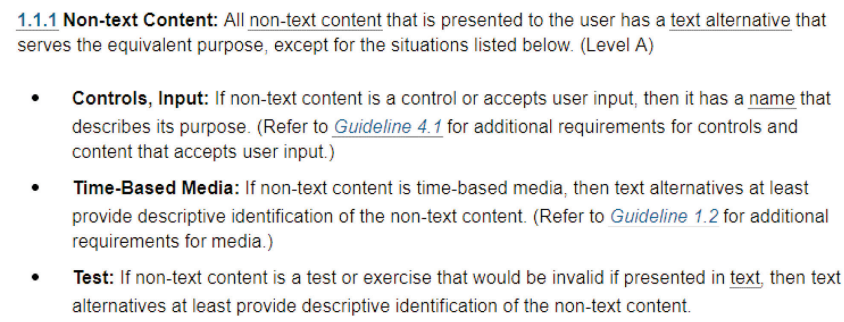
[1] https://angular.io
[2] https://vuejs.org
[3] https://reactjs.orgw
[4] https://phantomjs.org
[5] https://www.seleniumhq.org

companies and individual people. WCAG provides multiple layers of guidance including principles, general guidelines, success criteria and a collection of sufficient and advisory techniques. There are four principles at the top of WCAG: perceivable, operable, understandable, and robust. Each principle contains general guidelines and basic goals to make WUI more accessible. For each guideline, multiple testable success criteria with sufficient and advisory techniques are provided with three levels of conformance A, AA and AAA [104]:

- *Level A* covers the most basic accessibility requirements. Example of Level A accessibility guideline: *All non-text content that is presented to the user has a text alternative that serves the equivalent purpose*.
- *Level AA* contains usability guidelines being the most common barrier for disabled users. Example of Level AA accessibility guidelines: *The visual presentation of text and images of text has a contrast ratio of at least 4.5:1*
- *Level AAA* is the highest and most complex level of accessibility containing very specific web accessibility guidelines. Example of Level AAA accessibility guideline: *Sign language interpretation is provided for all prerecorded audio content in synchronized media*

The World Wide Web Consortium (W3C) that is owner of WCAG specification claims that most organisations will not achieve WCAG Level AAA as these guidelines are too strict and following all Level AAA guidelines may be extremely time-consuming for some content. For example, applying Level AAA guideline "*Sign language interpretation is provided for all prerecorded audio content in synchronized media* [104]" to existing media content should be reviewed and sign language interpretation is presented.

WCAG guideline description defines the characteristics WUI elements must hold to be compliant with the guideline; such guidelines can often be checked automatically that is why WCAG guidelines are often addressed by tools for automated usability evaluation. For example, Figure 2 presents requirements WCAG defines towards the use of alternative text declaring that all non-text content that is presented to the user has a text alternative. Also, additional requirements are defined for controls, inputs and time-based media.



1.1.1 **Non-text Content:** All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below. (Level A)

- **Controls, Input:** If non-text content is a control or accepts user input, then it has a name that describes its purpose. (Refer to *Guideline 4.1* for additional requirements for controls and content that accepts user input.)
- **Time-Based Media:** If non-text content is time-based media, then text alternatives at least provide descriptive identification of the non-text content. (Refer to *Guideline 1.2* for additional requirements for media.)
- **Test:** If non-text content is a test or exercise that would be invalid if presented in text, then text alternatives at least provide descriptive identification of the non-text content.

Figure 2. Example of WCAG guideline declaration – all non-text content has a text alternative [105].

An alternative accessibility standard to WCAG is Section 508[1] (USA standard) – a technical standard against which products can be evaluated to determine if they meet the technical compliance. Section 508 is expected to require conformance to the W3C WCAG 2.0 Level AA. Section 508 covers different aspects of IT including Web and desktop applications, and telecommunication equipment. All United States government agencies

[1] https://www.section508.gov

are required to comply with Section 508[1]. The standard contains subparts being separate technical standards specific to various types of technologies and performance-based requirements. Web-based intranet and internet information and applications, and software applications and operating systems are the groups of technical requirements (that are part of Section 508 standard) intended for improving the accessibility for WUI. Section 508 provides a detailed explanation of every guideline to be followed including success criteria, evaluation techniques and sample examples of following the guideline in different contexts.

In general, WCAG is the only international accessibility standard that became ISO/IEC International Standard[2]. Nevertheless, some countries develop their own accessibility standards. For example, BITV[3] is a German variant of the internationally recognised web accessibility standard WCAG 2.0; RGAA 3[4] is a standard that all French central government websites should follow (based on WCAG). The main difference between BITV 2.0 and WCAG 2.0 is that BITV contains guidelines that are divided into two priorities (Priority l and Priority 2) instead of three levels as in WCAG (Level A, Level AA, Level AAA) where Priority l mostly contains guidelines from WCAG Level A and Level AA and Priority 2 contains WCAG Level AAA guidelines. In fact, there are single guidelines that were moved from Level AAA to Priority l. Also, BITV 2.0 contains several additional guidelines for sign language.

In reality, however, developing new accessibility standards will not benefit much as all existing accessibility standards are using WCAG as a fundament. That is the reason, why many countries require government web applications to comply with WCAG Level AA guidelines including Estonia [27], Australia[5], United Kingdom[6] and many others. Moreover, according to EUROPA - Web accessibility policy, all EU Commission websites should follow WCAG Level AA guidelines [26].

Interoperability Framework of the State Information System [27] defines a set of guidelines for preparing public sector IT legal acts, designing IT solutions and organizing IT related public procurements in Estonia. One of the main statements it contains is that WUI of information systems should comply with WCAG 2.0 Level AA.

3.2.2 HCI Research

The sources of usability guidelines are not limited only to standards. In fact, standards like WCAG or Section 508 cover only accessibility guidelines being a narrow scope of usability. Multiple other specific parts of a webpage like form inputs, layout of elements, search field usability, content organization and many other categories of usability guidelines are not covered by standards at all or covered superficially. That is the reason why many research works aim to establish usability guidelines covering various characteristics of WUI elements including usability guidelines for images and links [29], guidelines addressing the navigation and page layout [30] and guidelines concentrating on mobile devices [31]. One of the most popular evidence based user experience research companies is Nielsen Norman Group. They conduct various usability studies and

[1]https://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards

[2] https://www.iso.org/standard/58625.html

[3] http://www.bitvtest.eu/

[4] http://references.modernisation.gouv.fr/referentiel/

[5] https://www.finance.gov.au/archive/publications/wcag-2-implementation/

[6] http://www.legislation.gov.uk/ukpga/2010/15/contents

present reports on various topics such as e-commerce usability guidelines and search form design recommendations [79] and design recommendation for content management [79]. The results provided in reports contain suggestions, guidelines, recommendations and considerations on how to make WUIs more usable for users. For example, the report of Content Management and Supporting Multiple Locations and Languages [79] contains 80 design recommendations for intranet. Every recommendation contains a name, a brief description, and screenshots with examples of pages that violate and follow presented usability guidelines.

One of the widely used lists of usability guidelines is combined by Shneiderman and Leavitt in [78]. The collection of guidelines contains more than 100 research-based web design and usability guidelines. These guidelines are not established by the authors but a result of their analysis on usability research works and usability evaluation reports. The compilation of usability guidelines helps to avoid potholes and swamps while designing highly usable WUI. Shneiderman and Leavitt unified the structure of all presented usability guidelines and presented them through a structure consisting of guideline, comments, category, sources, relative importance and strength of evidence. Appendix 1 shows the example of usability guideline structure using the next usability guideline as an example *Design for User's Typical Connection Speed*. Relative importance and strength of evidence are two characteristics being quite subjective as the importance of guidelines varies between web applications. Nevertheless, these characteristics for the definition of guidelines are widely used [15] [106].

With the emerge of mobile technologies and widespread of smartphones, research focused on improving WUI usability has spread to mobile platform. Bringing web applications to mobile device platform involves certain benefits and presents serious challenges and peculiarities in usability and user experience because of smaller screen size, finger as a typing device and additional features like camera and accelerometer [33] [80]. In fact, mobile WUI should also follow same accessibility guidelines as other WUI. Nevertheless, WCAG does not provide separate guidelines for mobile accessibility. W3C provides Mobile Web Best Practices [32] that contain certain basic guidelines (around 60 guidelines) for developing WUI for mobile platforms (the document has not been updated since 2010).

3.2.3 Company Specific Usability Guidelines

It is not always enough to follow guidelines derived from the standards and usability reports. Many companies design or outsource from design companies their own corporate usability guidelines obliging web applications within the organisation to follow similar style guidelines. That ensures that all views and pages of different company applications share equal level of usability. These guidelines are commonly known as style guides. A style guide contains guidelines that establish consistency between applications encouraging developers to design highly usable web applications. As a result, users using multiple applications will have a better understanding of WUI content. For example, style guide can contain guidelines for submit and cancel buttons that define the colour scheme for each of them (e.g. submit is blue and cancel is red). When the same colour scheme is used across one web application or multiple web application of the company then users can easily find submit and cancel buttons as they already know the colour for different types of buttons.

Company-specific usability guidelines focus on guidelines for presentation elements, including visual design elements such as colour, logos, fonts or icons; page or screen layouts including spacing, justification and common items; and the correct use of

standard UI controls such as buttons, drop-down selections, radio button or checkboxes [35]. Figure 3 shows an excerpt from Tallinn University of Technology style guide demonstrating colour styles for buttons and examples of fonts. The excerpt provided contains only a small part of the style guide that should be followed. The entire style guide also includes allowed icons, styles for input and forms, colour palette, guidelines for styling form controls (e.g. input fields), page components (e.g. pagination) and navigation bars. Based on style guide strict usability guidelines are created. For example, based on the data presented in Figure 3 next usability guidelines can be defined: primary buttons should have a purple background colour with white font in front. The guideline can also be evaluated automatically.

Company-specific usability guidelines are widely used for both back office web and front office web applications where WUI design between different web applications could be unified simplifying their use for the company employees and external users in case of the front office web application. These styles are about corporate identity and not applied only for web but also for publications, name cards etc.

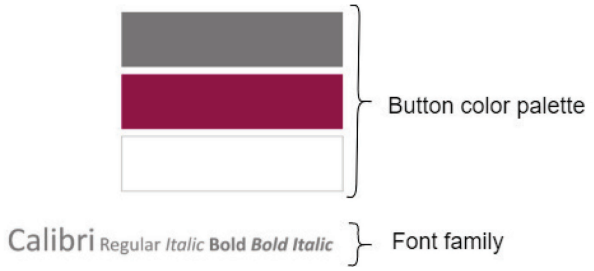


Figure 3. Example of a company-specific usability guideline, excerpt from style guide of Tallinn University of Technology[1].

3.3 Defining Usability Guidelines Programmatically

The formation of comprehensive usability guidelines suitable for automated usability evaluation is not an ordinary activity as every single usability guideline is not appropriate for automated evaluation. Properly defined usability guideline suitable for automated usability evaluation should contain a reference to an element (e.g. link, button), attribute or property (e.g. contrast, position) being evaluated and the acceptance condition (e.g. size > 10px); Figure 4 presents example of usability guideline (Example A) that can be evaluated automatically whereas Example B presents example of a guideline that cannot be evaluated automatically.

In order to automatically evaluate usability with a tool, guidelines must be formalised and implemented programmatically. Broadly, two different approaches exist:

1. embedding usability guidelines into the evaluation logic of a tool.
2. separating guideline definition from the evaluation logic by the mean of an external language. The vast majority of external languages are based on XML schema.

[1] https://www.ttu.ee/public/u/ulikool/TTU_stiiliraamat_2017/mobile/index.html

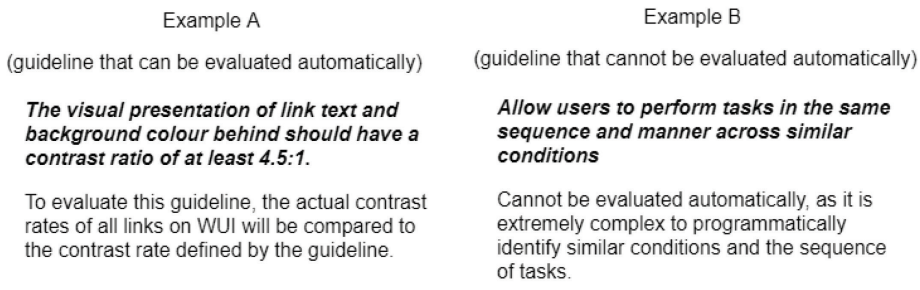


Figure 4. Example of a guideline that can be evaluated automatically and a guideline that cannot be evaluated automatically (Example A and B).

3.3.1 Embedded Usability Guidelines

Usability guidelines can be embedded (or hard-coded) to evaluation logic. Hard-coding usability guidelines into the evaluation logic is more straightforward and easy to implement because it does not require developing the component for reading custom usability guidelines. Nevertheless, modifying existing or adding new guidelines to such type of tools is extremely complicated as it requires rewriting the code of the tool. There are certain tools that do not provide a possibility to define new usability guidelines such as AChecker [11], TAW[1] and TotalValidator[2].

AChecker introduced by Gay and Li in [11] is an open source tool used to evaluate accessibility of WUI by automatically evaluating WUI conformance to WCAG and Section 508 guidelines. The only way to add additional guidelines to the AChecker is to modify source code of the tool. Another example of a tool that embeds usability guidelines into the evaluation logic is Test Accessibility Web (TAW)[1], which allows checking WUI conformance to WCAG guidelines automatically.

Embedding usability guidelines into the evaluation logic is not an advised approach as extendibility of predefined usability guidelines with custom application specific usability guidelines is a vital requirement as the existing usability guidelines can change or new appear. In fact, it is unlikely that these tools contain usability guidelines suitable for each application. Another drawback of the approach that it makes complicated to append additional usability guidelines to the existing list of usability guidelines. To add additional or modify existing guidelines source code of the tool should be modified. The modification of the code requires additional expertise in programming languages that may be an obstacle for the users of the tool. Also, some tools such as TAW[1] do not provide the source code of the tool meaning that only the tool developer can amend usability guideline. Yet, they are useful for evaluating WUI against the set of guidelines established in them.

3.3.2 Separated Usability Guideline Definition

In order to enable extendibility and a possibility to modify usability guidelines, guidelines need to be formalised in some formal language and separated from the logic of the tool. Several attempts have been proposed. Beirekdar et al. [95] developed a guideline definition language (GDL), which was one of the first languages for guidelines' definition based on the XML schema. Nevertheless, the approach did not have any traction due to the certain severe disadvantages, such as the absence of dynamic web support (like

[1] http://www.tawdis.net
[2] http://www.totalvalidator.com

JavaScript and CSS) and inability to define relations between WUI components (e.g. the relations between the label and associated field). These disadvantages called for adopting alternative XML-based language overcoming the limitations of GDL.

The successor of GDL becomes another XML-based language for guidelines definition – Unified Guidelines Language (UGL) [61]. The language was developed based on a robust review of various types of accessibility guidelines integrating necessary elements into the language defining a wide range of various test cases. The latest and the most comprehensive XML-based language for defining usability guidelines have been proposed by Schiavone and Paterno in [10]. They proposed Language for Web Guideline Definition (LWGD) enabling a simple and flexible formalisation of guidelines, usually defined using natural language and expressing more fine-grained guidelines than any existing XML-based alternative. Figure 5 shows an example of describing WCAG guideline "*Every image should have alternative text defined"* using LWGD.

This natural language description can be translated into a more structured sentence: For each *img* HTML element inside the Web page, check that it has an *alt* attribute defined. As we see from Figure 5, LWGD is a low-level HTML element-centric language requiring solid knowledge of HTML to combine a new usability guideline. Despite the fact that LWDG is the most advanced language for defining custom usability guidelines and it is capable of defining most WCAG accessibility guidelines, it is limited mostly to defining the structure of tags, attributes and their relations; defining visual characteristics of WUI is not possible with this language, as it was intended for defining accessibility guidelines. Despite the fact that LWDG was designed as HTML-centric language, extending LWDG with the elements and characteristics from the domain of visual usability characteristics will make LWDG more sophisticated.

```
<gdl_set xmlns="http://giove.isti.cnr.it"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://giove.isti.cnr.it.lwgdl.xsd">
    <name>Web Content Accessibility Guidelines</name>
    <guideline id="24.4" summary="Every image should have alternative text.">
        <description>
Check that every HTML img element contains valued attribute alt
</description>
        <criterion id="24.4.1" type="accessibility"
target="page">
            <checks>
                <cp>
                  <eval_object mandatory="no" code="html">
                      <object type="tag">img</object>
                  </eval_object>
                  <conditions>
                      <condition>
                          <evaluate operator="check" cond="followedby">
                              <eltype="attribute">alt</el>
                          </evaluate>
                      </condition>
                  </conditions>
                </cp>
            </checks>
        </criterion>
    </guideline>
</gdl_set>
```

Figure 5. The definition of accessibility guideline using LWGD.

An alternative to XML-based languages for defining custom usability guidelines – definition table based approach that was proposed by Dangli [15]. Definition table based

approach incorporates usability guidelines through the use of special database table allowing defining evaluation conditions. Each column of the table in database represents certain characteristic such as HTML element, attribute, the corresponding text or the value being evaluated. Also, the table contains conditions to be evaluated. For instance, Figure 6 shows an example of incorporating usability guideline into definition table: "URLs should not be complex and should ideally be less than 70 characters". Every element with a tag <a> that has *href* attribute as stated by the contents in the fields under the *tagA* and *attributeA* columns respectively. For the guideline not to be failed, this content of *href* needs to be less than 70 characters long, as stated by the contents in the fields under the *compareOperator* and *sizeA* columns respectively. The guideline presented in Figure 6 does not depend on any other HTML element that is why they are all set to NULL.

Definition table-based approach is an HTML-centric approach that is capable of evaluating values of HTML tags and corresponding attributes. Moreover, it allows evaluating the order and position of HTML tags on the screen. The advantage of the approach is that it is relatively easy to define new guidelines using that approach because of simple and logical table structure. The primary disadvantage of definition table-based approach is that extending definition table with additional concepts like contrast, position, distance, width, length, scrolling will make the table more complex and less understandable reducing the main purpose of definition table-based approach to provide lightweight structure for defining HTML-centric guidelines. Also, it is impossible to validate the correctness of usability guidelines automatically in definition table based approach; custom validator should be developed for that purpose.

tagA	**Attribute**	**valueA**	**sizeA**	**compareOperatorA**	**ruleA**
A	Href	NULL	70	<	True
tagB	**Attribute**	**valueB**	**sizeB**	**compareOperatorB**	**ruleB**
NULL	NULL	NULL	NULL	NULL	NULL

Figure 6. Example of incorporating the guideline into Guideline Definition Table.

One of the objectives of the thesis is to propose the tool for automated evaluation of visual characteristics of WUI. As far as LWGD and UGL do not provide functionality for defining custom usability guidelines for checking visual characteristics of WUI, in the first preliminary prototype of the *Guideliner* published in [14] XML based metalanguage to define such guidelines was proposed, calling it usability guideline definition language (UGDL). Figure 7 presents an example of custom usability guidelines containing self-explained elements. Each guideline has a name and description followed by its evaluation metrics. The main difference between LWGD and UGDL is that the latter defines elements on the screen rather than their representation in HTML code. Figure 7 shows that element *link* (HTML syntax does not contain *link* tag or attribute) was used to define usability guideline checking the length of the links. LWGD manipulates with HTML tags and attributes (refer to Figure 5). As a result, UGDL is not tight to any category of guidelines (like HTML-specific or visual guidelines). However, development of UGDL was discontinued (the limitations of XML-based approaches are discussed in Section 3.3.3) in favour of more flexible and feature-rich ontology for defining custom usability guideline described in Section 3.4.

```
<guideline name = "10.11 Use Appropriate Text Link Lengths"
description = "Make text links long enough to be understood, but short enough to
minimize wrapping. " >
   <link>
     <length notMoreThan = "50" />
   </link>
</guideline>
<guideline name = "5:7 Limit Homepage Length"
 Description = "Width scroll is not allowed.">
   <page type = "HOME_PAGE">
     <scroll type = "width" allowed = "false" />
   </page>
</guideline>
```

Figure 7. Example of custom usability guideline description using UGDL.

3.3.3 Limitations of Existing Approaches

All the aforementioned approaches including embedding usability guidelines into the code, definition table based approach and XML-based approaches are capable of defining various usability guidelines describing the structure of HTML code. They are capable of incorporating most of the WCAG accessibility guidelines suitable for automatic evaluation covering the proper structure of tags, attributes and the relations between them. In fact, these tools allow expressing advanced guidelines regarding multiple objects and more complex requirements regarding the greater number of types of relations between objects.

Nevertheless, defining guidelines for evaluating visual aspects of a WUI like the presence of scrolling, layout of elements, the positions of elements on the screen, the distance between elements and many other assessments are not possible with these approaches [12]. Of course, it is possible to extend existing language with new concepts and relations. Nevertheless, there are conceptual disadvantages of XML Schema being used as a language for defining usability guidelines. The main disadvantage of XML Schema is that XML Schema defines the structure of document providing the prescriptions how the document is styled. It introduces additional layer of complexity when it is used to describe the domain of knowledge [107]. For example, it is good for defining an element, element attributes and metadata about attributes such as occurrence, value pattern, minimum and maximum length.

Another drawback of XML based languages is that they are HTML centric consisting of HTML tags, attribute and their relations. Such method does not suit for defining visual usability guidelines as visual characteristic (e.g. there is no HTML tag responsible for scrolling) cannot be always defined with HTML tags. Adding additional language constructions that enable to define visual usability guidelines will make the language opaque and unclear as it should support both HTML tag specific definition of guidelines and the definition of visual object specific usability guidelines.

The inability to perform the automatic validation of usability guidelines and sophisticated way of integrating visual usability guidelines to the existing languages for defining custom usability guidelines calls for adopting alternative approach for defining usability guidelines compared to existing XML-based approaches. In this thesis the author has explored a possibility to solve the shortcomings of XML-based languages by constructing special domain ontology.

3.4 Ontology for Defining Usability Guideline

Ontology is a formal, explicit specification of a conceptualization [108]. Ontology is a prominent component of intelligent system. It enables storing and capturing domain knowledge in a human-understandable and at the same time in a machine-processable way. In general, ontology contains entities, attributes, relations and axioms, allowing formal presentation of knowledge as a concept within a domain, and the relations between concepts.

Ontology can be described in many languages, e.g. Ontolingua, Loom, and Semantic Web languages, such as OIL, DAML+OIL, W3C Web Ontology Language (OWL) and RDF Schema. This thesis is concentrating on OWL – a standard language for ontology description recommended by the W3C. *OWL is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things*[1]. OWL was selected for the dissertation as it is the most widely used language for creating ontologies [109]. OWL enables capturing knowledge by representing the concepts and relations between them.

The knowledge in ontology is presented via the terminology box (TBox), and assertions box (ABox). The primary components of OWL ontologies are classes (sets consisting of individuals), properties (relationships that link two individuals together) and individuals (also called instances). Classes and properties are combined into Terminology Box (TBox) – a set of definitions and specifications. In the established usability ontology, the TBox contains the concepts and terms of usability guidelines. Whereas, Assertion Box (ABox) contains individuals or a set of data axioms; in particular, the ABox contains various usability guidelines defined as individuals.

The mostly applied relations between ontology concepts are the *is-a* relation, that defines the hierarchy between class to sub-class, and the part-of relation, defining the relationship of an entity and its components. The formal semantics of OWL allows inferring of classification taxonomies and thus helps to identify inconsistencies in the established ontology at any time. Thus, OWL is progressive language that is used in developing intelligent systems, that also coincides with the aim of current work.

Other important units of ontology are instances. Instances (also known as Individuals) can be referred to as being 'instances of classes'; they are specific instances of the objects or concepts. "*A class in ontology is a classification of individuals into groups which share common characteristics. If an individual is a member of a class, it tells a machine reader that it falls under the semantic classification given by the OWL class*" [105].

Reasoners are used to check the validity of ontology concepts and infer consequences. Reasoners infer logical conclusions from the asserted facts or axioms. Moreover, they check the consistency (if a class can have any property, relation or not based on description) of an ontology. They are used to calculate inferred ontology class hierarchy based on the definitions of the concept. Hence, the application of reasoners ensures coherent and consistent hierarchy in ontology.

The purpose of current thesis is to define usability ontology for storing usability domain knowledge. As already mentioned in Section 3.3.3, existing XML based approaches are focusing on defining HTML specific guidelines describing HTML tags, their attributes and the order. Extending these languages with the concept of visual usability guidelines makes the language opaque and ambiguous. Expressive possibility of XML based languages is limited, and thereby there is a need for something more expressive

[1] https://www.w3.org/OWL/

than just a metalanguage. For the reasons presented above, ontology would give more powerful means to store usability domain knowledge, in particular, different types of guidelines (e.g. HTML-specific, visual guidelines). Therefore, the exploitation of ontology instead of XML-based language allows inferring of classification taxonomies and helps to define inconsistencies in the designed ontology. Overall, it is providing an effective way to uniformly describe and store usability domain knowledge through various aspects, being both human- and machine-readable, and deliver a shared vocabulary describing types of concepts that exist in the domain, and their properties (metrics) and relations.

The idea of integrating WUI usability guidelines into ontology is not new. Xiong *et al* in [110] presented ontology-based approach for organising and generalizing usability guidelines. They point out that the main drawback of their approach is that ontology contains only domain knowledge and in order to apply it to the evaluation process additional mappings should be introduced. Another problem is that proposed ontology contains only HTML-specific domain knowledge; it contains concepts for defining the tags, HTML attributes and relations between them. Authors point out that it is impossible to define usability guideline for describing background colour or contrast of element on the page with their approach.

The usability ontology presented in this chapter addresses the aforementioned problems providing possibility to define HTML-specific domain knowledge as well as domain knowledge of visual characteristics of finally rendered WUI. Another advantage of usability ontology presented in this chapter (Section 3.4.1) is that it is designed with the possibility to be used for automated usability evaluation. This means that ontology contains all required data and object properties that can potentially be reused by automated usability evaluation tools.

3.4.1 Usability Ontology

Ontologies are mostly designed using specific software – ontology editors. To establish usability ontology, the author used the Protégé[1] ontology editor version 5.3 for the task. Protégé is a free open-source platform providing a suite of components to build domain models and knowledge-based applications with ontologies. The Protégé ontology editor can be easily integrated with various reasoner tools, for example, the HermIT, Pellet, and Fact++. In practice, there are no limitations on editors to be used.

The underlying idea of the ontology is to deconstruct the WUI view, and page as a whole is one of the views. Such deconstruction is very important, as in general, usability guideline covers only a single view (or WUI element) of the page. Overall, usability guidelines are divided into the characteristics; characteristics are grouped into the corresponding objects, object properties, data properties and relations between various objects. The data was used as a fundament for the structure of the usability ontology. Every guideline suitable for automated evaluation was split into different parts that characterize the element or the attribute being evaluated. For example, usability guideline *The contrast ratio of the link text and its background has a contrast ratio of at least 4.5:1* presented in [103] by WCAG is divided into next characteristics:

- Element – link.
- Element attribute – contrast.
- Value of attribute – 4.5:1.

In order to formalise such structure in established usability ontology primitive classes (they have only necessary conditions defined) have been used including *Guideline*,

[1] Protégé, https://protege.stanford.edu/

GuidelineElement, *ElementAttribute, PageAttribute* and *ValuePartition* (a special class used to refine guideline descriptions through a pre-defined set of value concepts). Figure 8 outlines class hierarchy of defined usability ontology and its some descendant classes. Overall, ontology contains around 250 defined classes.

The main concept of usability ontology is class named *Guideline* that is needed for storing usability guidelines. All guidelines defined in ontology are stores as a subclass of *Guideline* class. Different types of usability guidelines such as mobile, desktop and accessibility guidelines (subclasses such as *MobileUsabilityGuideline*, *DesktopUsabilityGuideline* and *AccessibilityGuideline*) are defined as a subclass of *Guideline.* Class *Guideline* also holds defined classes that are used to infer new subclasses (e.g. *GuidelinesConcernedWithLink* – a defined class that holds all classes concerned with links in the set of primitive guidelines) via reasoning – the actual class hierarchy for a defined class is computed by the reasoner according to definitions given in the ontology. Defined classes are subsequent subclasses of the *Guideline* parent class and contain only class definitions. All guidelines are defined in the primitive class *UsabilityGuideline*. Each guideline is linked to the *GuidelineElement* class via the *hasGuidelineElement* object property. That relation defines which *GuidelineElement* is described by guideline. Also, each guideline is annotated using the following annotation properties (predicates that provide additional informal documentation annotations about ontologies):

- Guideline – short description of the guideline (custom defined annotation).
- Comment – provides details of the guideline (Protégé built-in field).
- Reference – source of the guideline (custom defined annotation).

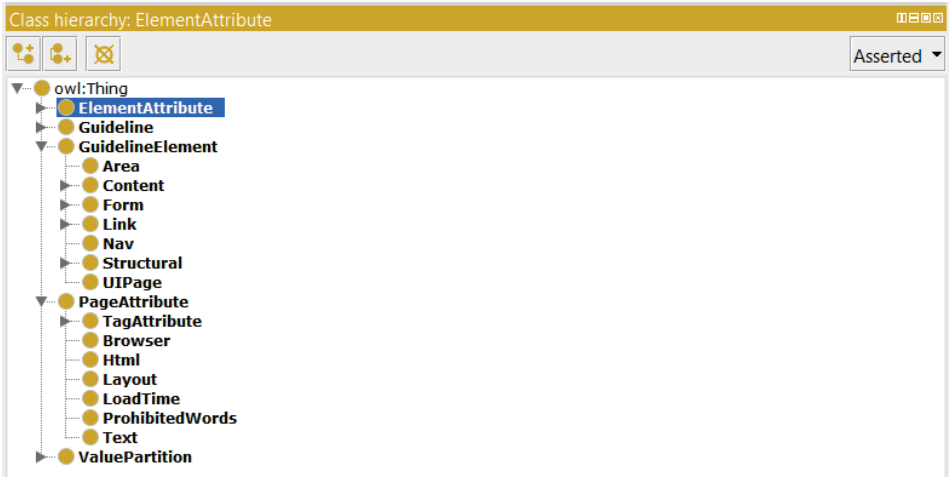


Figure 8. Class hierarchy of defined usability ontology and some of its descendant classes (excerpt from the Protégé ontology editor).

The *GuidelineElement* class holds subclasses describing elements a guideline may be applied to such as button, text input, frame, link, and paragraph. *GuidelineElement* is used to represent the element being evaluated by usability guideline; it can represent WUI element (e.g. link or button) as well as the page as a whole (e.g. when page-specific characteristics such as load time or the length of scrolling are evaluated). Appendix 2 outlines a selection of usability ontology showing the main concept - *GuidelineElement* and its descendant classes over the is-a relationship. This taxonomy of subclasses reflects

WUI elements or their equivalents and their related structure. *GuidelineElement* class contains elements the guideline can be applied to. For instance, the WUI element correspondent to the class *TextualInput* is the element defined in HTML code as `<input type="text">`. From the perspective of this thesis, the purpose of usability ontology is to describe and store usability domain knowledge. Defined ontology does not contain any direct mapping between the *GuidelineElement* classes and corresponding HTML elements.

One important problem that is addressed with the designed ontology is ability to define both HTML-centric and usability guidelines covering visual characteristics. These types of usability guidelines are distinguished by data properties. *ElementAttribute* class incorporates both HTML attributes and visual characteristics of element like width, distance and contrast. In order to distinguish HTML and visual characteristics, the object property *hasAttributeType* was introduced. Object property *hasAttributeType* links *ElementAttribute* class to enumerated class *AttributeType* restricted to the two individuals – *VisualAttribute* (for marking visual characteristics) and *HtmlAttribute* (for marking HTML attributes); *VisualAttribute* and *HtmlAttribute* are defined as disjoint classes meaning that the attribute cannot be both *VisualAttribute* and *HtmlAttribute* at the same time. Figure 9 presents the description of *Contrast* class outlining classes it is subclass of. Statement *hasAttributeType only VisualAttribute* states that *Contrast* is a visual characteristic and all *GuidelineElements* linked to the contrast will be treated as usability guideline covering visual characteristics.

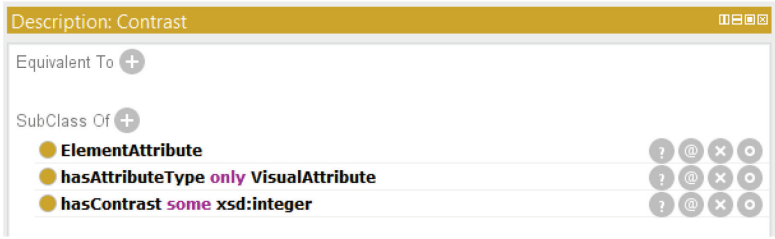


Figure 9. Example of visual usability guideline regarding contrast: description of class Contrast in the Protégé ontology editor.

GuidelineElement class is linked to class *ElementAttribute* via the *hasAttribute* object property. In this case, the domain of *hasAttribute* object property is *GuidelineElement* and the range is *ElementAttribute*. That enables to describe the attributes that are associated with the *GuidelineElement.* Let us take a closer look at the example of an existential restriction *hasAttribute some Contrast* on the *Link* class. It acts along the *hasAttribute* property and has a filler *Contrast*. This restriction describes the class of individuals that have at least one *hasAttribute* relationship to an individual that is a member of the class *Contrast.* In fact, *GuidelineElement* class can be linked to multiple classes *ElementAttribute* via the *hasAttribute* object property. Additionally, the usability ontology contains the next object properties: *hasAttribute, hasAttributeType, hasCase, hasContentType, hasDeviceType, hasLengt, hasDistanceType, hasGuidelineElement, hasGuidelineType, hasLayoutType, hasPageAttribute, hasUnit, hasPositionType, hasRelativeImportance, hasStrengthOfEvidence, hasTagAttribute, hasUnitAction.*

In addition to object properties, multiple data properties are defined in the ontology. The difference between data property and object property is that while object property links to objects (e.g. *GuidelineElement* class can be linked to subclasses of *ElementAttribute*), then data property links class to an XML Schema Datatype value or an

RDF literal (e.g. String, Integer). Data properties are used in the usability ontology to set certain characteristic of the guideline to a concrete value. Data property can have a domain and a range defined. They link together individuals from domain to the XML Schema Datatype value or RDF literal. Thus, they describe relations between an individual and data values. For example, data property *hasMinContrastValue* is used to state the contrast rate of a particular WUI element (Figure 10). The domain of data property *hasMinContrastValue* is *GuidelineElement* and the range is *xsd:string*. Data property assertions can be used to associate individuals with data properties. The usability ontology has the following data properties defined: *hasContentLength, hasMaxNumberOfInput, hasMinNumberOfInput, hasScroll, isVisited, hasUnit, hasValue, isFollowedBy, isOneDirectional, isPrecededBy, isSelected, isUnique, isValid, isValued.*

As it is discussed before, *ElementAttribute* class describes attributes of WUI elements. Nevertheless, there are characteristics applicable only to the page as a whole. For example, page load time or the length of scrolling are the attributes of the whole web page. That is why class *PageAttribute* was introduced uniting page specific characteristics. Currently, only *UIPage* class (subclass of *GuidelineElement*) is linked to subclasses of *PageAttribute.*

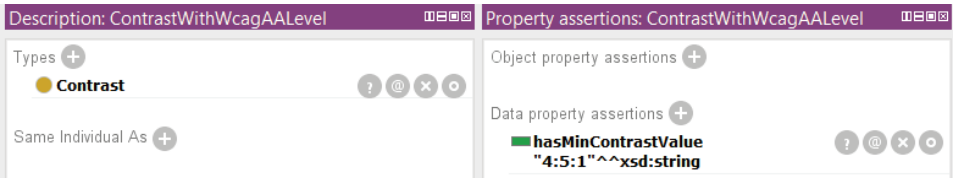


Figure 10. Property assertion view for hasMinContrastValue data property (excerpt from the Protégé ontology editor).

ValuePartition class plays an important role in defined usability ontology restricting the definition of certain classes to the defined set of values. Overall, seven subclasses of *ValuePartition* class were defined:

1. *StrengthOfEvidence* defines strength of evidence of a particular guideline on the scale of 1-5 (added to the guideline description over *hasStrengthOfEvidence* object properties accordingly in the ontology).
2. *RelativeImportance* defines relative importance of the guideline on the scale of 1-5 (added to the guideline description over *hasRelativeImportance* object properties accordingly in the ontology).
3. *Unit* contains measurement for the element's characteristic being evaluated. For instance, width of WUI element could be defined in Pixels or Ems. Usability guidelines should be strictly defined as designing usability guideline for evaluation the width of element leaving the unit of measurement out can be interpreted ambiguously. *Unit* class reduces uncertainty of usability guideline meaning.
4. *DeviceType* specifies the device to which the guideline could be applied. It is an essential *ValuePartition* as there are guidelines applicable only to mobile or to the desktop devices.
5. *PositionType* defines the relative position of the element in the set of elements. For instance, there are usability guidelines checking the visual position of element regard to another element such as checking that the labels should be above the input fields on mobile devices.

6. *LayoutType* specifies possible types of layout of the WUI elements on the screen. The appropriate example could be that in order to define usability guideline claiming that vertically aligned radio buttons are more usable on mobile device; *Layout* type class described the vertical alignment via subclass *Vertical*.
7. *AttributeType* allows distinguishing HTML attributes from visual characteristic of WUI. For instance, *alt* attribute of the Image is HTML attribute as it could be retrieved directly from the *img* tag. On the contrary, the contrast of image cannot be retrieved from the HTML code of element but additional calculations are needed.

ValuePartition class lists all individuals that are members of the class. For example, *ValuePartition* class has a subclass *PositionType* that contains the individuals (and only the individuals) *Above*, *Below*, *Left* and *Right*. Classes such as this are known as enumerated classes. All subclasses of *ValuePartition* class are defined as enumerated classes. Figure 11 shows *Class Description View* of *PositionType* class demonstrating that *PositionType* is equivalent to anonymous class defined by enumeration {Above, Below, Left, Right}.

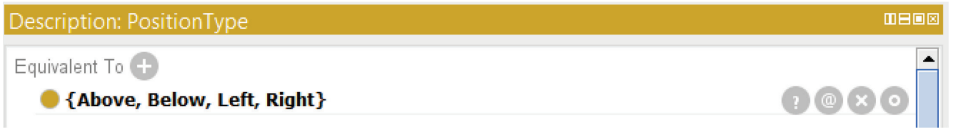


Figure 11. The class description view displaying an enumeration class PositionType (screenshot from the Protégé ontology editor).

Overall, proposed ontology has been designed to be flexible enough to capture and store usability domain knowledge covering both HTML elements and their layout, and also the visual aspects of WUIs. The design of the core ontology is extendible, and new guidelines can be easily defined. Section 3.4.2 discusses the exploitation of the usability ontology for WUIs and how to define usability guidelines in it.

3.4.2 Using Usability Domain Knowledge to Define Usability Guidelines

Having established the ontology design, let us now focus on re-using established domain knowledge to define custom usability guidelines. Usability guidelines are defined in the ontology as subclasses of class *Guideline*. The process of adding new usability guidelines is based on the example of mobile web usability guideline presented by Google Mobile group[1]: *All buttons should be at least 48 CSS pixels wide*. Figure 12 shows an example of a guideline concept description in the Protégé ontology editor defining mobile usability guideline *28-ButtonShouldBeWideEnough*.

In order to define a new usability guideline in ontology, new subclass is added as a sub class *UsabilityGuideline*, and a naming convention shown by Figure 13 must be applied. The name of the class starts with unique identifier (is needed to distinct usability guidelines), following by the short name of the guideline. The class is made disjoint with other subclasses of class *UsabilityGuideline*, so instance of any subclass of *UsabilityGuideline* class cannot be the instance of another subclass (as each usability guideline is unique). As shown in Figure 12, class *28-ButtonShouldBeWideEnough* contains object property *hasGuidelineElement* which defines the element being

[1] https://developers.google.com/speed/docs/insights/SizeTapTargetsAppropriately

evaluated. Statement *hasGuidelineElement only Button* shows that the defined guideline evaluates only buttons; statement *hasDeviceType only Mobile* outlines that the guideline is applicable only to mobile devices.

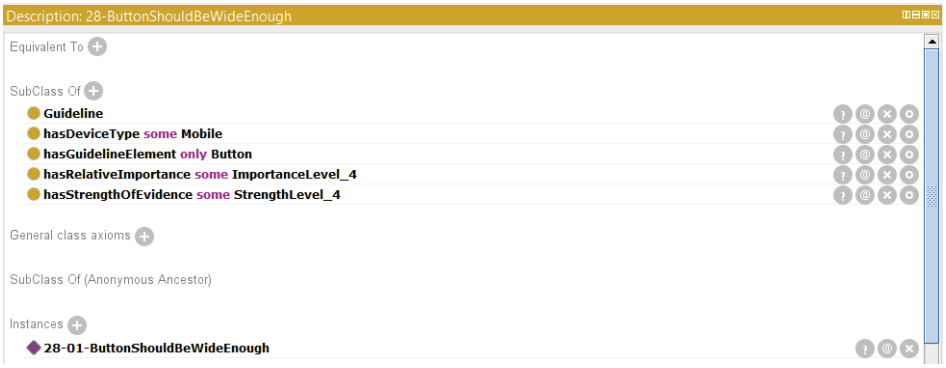


Figure 12. Example of a usability guideline concept definition for the guideline "Buttons should be wide enough" (screenshot from the Protégé ontology editor).

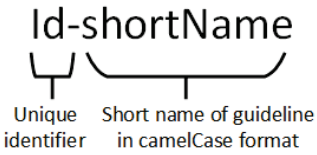


Figure 13. Naming convention applied while describing classes of usability guidelines in the ontology.

Additional information about the guidelines is provided as *Class Annotations* where guideline's general description is added as the 'guideline' annotation, the annotation 'comment' provides further details and 'reference' provides URL for the particular guideline (Figure 14).

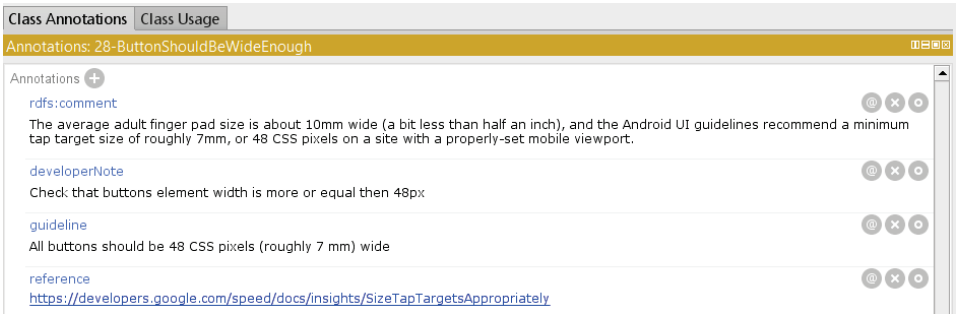


Figure 14. Example of guideline annotation with human-readable comments (screenshot from the Protégé ontology editor).

Next step is to create individuals based on the aforementioned description. Individuals (also known as Instances) can be referred to as being 'instances of classes'. Individuals may represent the objects as an abstract individual such as numbers and words. Thus, it is possible to create multiple individuals based on the same guideline class. In the context

of established ontology, individuals allow defining the same usability guidelines but with different metrics. Figure 12 contains the individual with name *28-01-ButtonShouldBeWideEnough*. After individual is created the *Width* of *Button* should be defined (according to the guideline it should be at least 48 CSS pixels). An instance of ontology class *Width* is created defining the value and the unit of measurement of the metric being evaluated. Figure 15 demonstrates the description and property assertion view for an individual of *Width* class. Statement *hasUnit Pixel* means that the *Pixel* is a unit of measurement for width of WUI element, and the statement *hasMinContentLenght 48* determines minimum value for the element measured in the units defined by *hasUnit*. The advantage of the approach that the evaluable metric can be easily modified. For example, a *Width* instance with *hasMinContentLenght* of 48 pixels can be created to define the length of button of mobile device and another instance *hasMinContentLenght* of 15 pixels to define the length of the button for desktop device. Usability domain knowledge defines that the object of type *Width* contains data property *hasMinContentLength*. Whereas, the definition of *Width* instances specific to the usability guidelines is done using individuals. Such approach is used to define all metrics such as length, width, contrast, position of WUI elements defined in usability guideline.

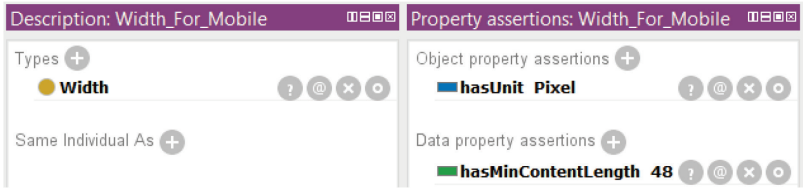


Figure 15. Example of guideline property assertion view (screenshot from the Protégé ontology editor).

To create the ontology of usability guidelines, the author analysed recommendations in scientific publications [29] [111], WCAG [103] and Section 508 guidelines[1], Research-Based Web Design [106] and Usability Guidelines from U.S. Dept. of Health and Human Services [78], and supplemented with recommendations from the Nielsen Norman Group [8] [31] [65]. Moreover, evidence-based user experiences and usability research works have been inspected [30] [31]. Overall, around 300 usability guidelines were analysed and 98 guidelines that can be processed automatically were filtered out. The purpose of usability ontology is to store only those usability guidelines that can be evaluated automatically, thereby only these usability guidelines were used as a basis for ontology definition. Table 1 contains categories of usability guidelines for the established ontology with the number of guidelines for each category.

Overall, ontology contains 98 usability guidelines including 55 accessibility guidelines, 23 common usability guidelines suitable for desktop and mobile devices and 20 usability guidelines suitable only for mobile devices. Appendix 3 presents some examples of guidelines for each category.

Categorisation of usability guidelines by WUI category (refer to Table 1 for more details) is widely used in researches on usability guidelines [106]. In the context of desktop devices, defined usability guidelines address laptops and desktop computers with diagonal 10 – 24 inches. Usability guidelines for tablet devices and smart phones with diagonal bigger than six inches and desktop devices bigger than 24 inches are not in

[1] https://www.section508.gov

the scope. Aforementioned groups of devices were taken as according to the research conducted in year 2016 and published in [112], the most popular screen size of mobile devices (more than 90% accessed web) are from 4 till 6 inches. In case of desktop devices, computers with diagonal 10 – 24 inches cover more than 90% of all web usage from desktop devices[1]. Overall, devices with screen size 6 – 10'' form less than 10% usage all over the world.

Overall, established ontology has been published in [113] and complemented with additional concepts and guidelines that are presented in current thesis. Presently, established ontology is used only for describing and storing usability domain knowledge. It does not perform any kind of evaluations of WUI conformance to the guidelines; however, based on available descriptions this can be achieved. The evaluation process of WUI is carried out by the WUI evaluation component, described in detail in Section 4.4.

Table 1. Categories of usability guidelines for the established usability ontology.

Category	**Mobile**	**Com-mon**	**Accessi-bility**	**Total**
Organisation of information and content	2	8	5	15
Tag attributes	0	0	14	14
Links	4	3	3	10
Screen-based controls	7	3	0	10
Tags	0	0	6	6
Radio Button	1	2	3	6
Text Appearance	0	0	6	6
Checkbox	0	2	3	5
Heading	0	0	5	5
Button	4	0	0	4
Text Appearance	0	3	0	3
Graphics, images and multimedia	0	1	2	3
Select	0	1	2	3
Scrolling	2	0	0	2
Password input	0	0	2	2
File	0	0	2	2
Textarea	0	0	2	2
Overall	20	23	55	98

3.5 Chapter Summary

This chapter focused on the sources of web usability guidelines and the methods used to formalise usability guidelines to use in different tools, e.g. XML Schema and definition table. Major contribution of this chapter is the usability ontology. The author analysed the structure of usability ontology he proposed as a solution to overcome the limitations of existing approaches such as inability to define visual usability guidelines, and presented an alternative method to define usability guidelines in a machine-processable

[1] https://techtalk.pcpitstop.com/2017/01/16/pc-monitor-display-size

form. The established usability ontology makes it possible to capture domain knowledge in human-understandable, yet in a machine-processable way. Design of the ontology is extendable; additional guidelines can be easily added based on the established usability domain knowledge.

This chapter constitutes to the first phase of the author's studies about web usability guidelines continued by research into automated evaluation of WUI conformance to the usability guidelines and usability evaluation during its implementation phase, discussed in the forthcoming chapters.

4 Automated Evaluation of Usability

High demand for human and time resources [47], difficulty to get potential users participating in usability evaluations [114] and limited coverage of evaluated features [46] are the main drawbacks of manual usability evaluation. The alternative to manual usability evaluation is automated usability evaluation requiring fewer resources to use, as the configuration of the tool for automatic usability evaluation can be done once and, afterwards the evaluation can be performed multiple times without any preliminary configurations. Automatic evaluation will save on involvement of human resources. In contrast, manual usability evaluation requires the involvement of experts with every usability evaluation.

This Chapter addresses automated evaluation of WUI conformance to usability guidelines, including visual usability guidelines, and proposes the *Guideliner* tool – a solution for automated evaluation of HTML-centric and visual usability guidelines. The *Guideliner* tool is one of the main contributions of the thesis author.

4.1 Introduction

Usability evaluation is a method for identifying specific problems with usability of products [9]. The evaluation can be conducted manually or executed automatically using special tools. Automated usability evaluation has multiple advantages over manual methods such as it does not require the involvement of potential users in usability evaluation, the evaluation is conducted by computer and the evaluation report is provided automatically, and it also increases the coverage of evaluated WUI elements. Overall, automated usability evaluation reduces the costs and time spent on usability evaluation.

Typically, research of automated usability evaluation is divided into three main areas: interaction-based usability evaluation [66] [67], metric-based usability evaluation [87] [88] and model-based usability evaluation [87] [89]. As described in Section 2.5.2, model-based and interaction-based approaches to automated usability evaluation suffer from subjectivity as the problems found during automated testing cannot be easily fixed for all users. On the contrary, metric-based automated usability evaluation provides precise results as the evaluation is performed based on known usability guidelines and measurable characteristics. Current thesis concentrates only on metric-based automated usability evaluation as (in comparison with interaction and model-based usability evaluation) it is more flexible to automation, the evaluation results, in general, are better interpretable and comparable to well-established guidelines and standards, and there are many methods to automate metric-based usability evaluation.

Multiple tools exist that are developed for automated metric-based usability evaluation. The widest variety of such tools – a list of 116 tools[1] – is provided by W3C in their Web Accessibility Evaluation Tools List[2]. The Web Accessibility Evaluation Tools List contains tools for evaluating WUI conformance to WCAG, Section 508, BITV (German government standard), JIS (Japanese industry standard) and conformance to other country-specific accessibility guidelines. The list also contains WUI element specific tools that evaluate only certain characteristics of elements such as the contrast rate of buttons, or broken links. A general characteristic of most of the automated accessibility

[1] as of 16.09.2018

[2] https://www.w3.org/WAI/ER/tools/

evaluation tools presented in this list is that they are concentrating on finding web accessibility problems parsing the HTML code of the WUI and finding deviations from accessibility guidelines in the HTML code [10] [15]. Only ten out of 116 tools listed address some visual aspects of WUI described by guidelines.

While W3C maintains a well-represented list of tools for WUI evaluation, Timbi-Sisalima et al. in [115] performed a comparative analysis of 126 online accessibility evaluation tools including tools listed by W3C, free and the commercial tools, and tools presented in scientific publications that are not available online. They found that the most popular tools in the market are those that have a web interface for running the evaluation and that perform the evaluation of complete web page. They filtered out 18 tools (e.g. AChecker[1], WAVE[2], Tenon[3], Tanaguru[4], Examinator[5]) out of 126 based on the characteristics of the tools including the guidelines tools support, report format, API support, repair recommendation. The results of their comparative analysis shows that about half of the evaluated tools (10 tools out of 18 most popular) are configurable (e.g. it is possible to select the category of guidelines for evaluation), the vast majority of tools (17 tools) are capable of evaluating only one page at a time (it is not possible to evaluate group of pages with them). In general, all evaluated tools provide detailed reports and the recommendations on how detected usability defects can be fixed. Overall, Timbi-Sisalima et al. point out that there is still a need for further research for developing tools for automated usability evaluation as mostly the existing tools are limited only to accessibility evaluation (they are all mostly concentrating on WCAG) with minor differences in a feature set they provide and the way they propose evaluation results.

Only a few tools out of 126 are able to assess visual characteristics of WUI in addition to accessibility guidelines – these are Wave[2], Google Mobile-Friendly Test[6], and TestMySite[7] etc. WAVE in addition to usability guidelines allows evaluating contrast of elements on the screen. Google Mobile-Friendly Test is only targeting mobile platforms and allows checking the size of links and buttons on mobile screens as well as mobile-specific HTML characteristics (e.g. HTML attribute viewport values). TestMySite[7] tool allows in addition to HTML-centric guidelines to evaluate page load time, optimization of images and scripts on the page for mobile devices.

Overall, all previously discussed tools are concentrating mostly on HTML-centric usability guidelines. Of course, there are tools that evaluate single visual usability guidelines such as contrast of elements and the size of buttons (e.g. WAVE). Nevertheless, none of the aforementioned tools provide a solution that is capable to evaluate multiple visual characteristics of WUI (e.g. length of scrolling, contrast, position, distance, width, length) at the same time, as their capabilities are limited to only a certain aspect of user interface and its usability. The primary motivation of the current chapter is to address the aforementioned gap by concentrating on the methods of automatic evaluation of WUI conformance to HTML-specific as well as visual usability guidelines on

[1] https://achecker.ca/checker/index.php
[2] http://wave.webaim.org/
[3] https://tenon.io/
[4] http://www.tanaguru.com/en/
[5] http://examinator.ws/
[6] https://search.google.com/test/mobile-friendly
[7] https://testmysite.withgoogle.com/

the final layout of WUI as a user sees it rendered in browser window, i.e. when all scripts and styles have finished their loading and have been applied.

In this chapter, a solution for automated evaluation of WUI conformance to usability guidelines called the *Guideliner* is presented. The *Guideliner* tool concentrates on automatically evaluating both types of usability guidelines – HTML-centric accessibility guidelines and guidelines covering visual characteristics of WUI. The main contribution to the community delivered herein is twofold – first a method for automated evaluation of visual usability guidelines of finally rendered WUI, and second – a tool to carry out automatic evaluation of WUI and its accordance to guidelines, including automated visual evaluation.

The rest of this chapter is organized as follows: Section 4.2 introduces the *Guideliner* tool and provides an overview of its architecture. Section 4.3 discusses the structure of ontology repository of usability guidelines and ontology processing engine used within the *Guideliner* tool, while Section 4.4 concentrates on the description of WUI evaluation component of the tool. In Section 4.5 the User Interface and technical interfaces of the *Guideliner* are described. Section 4.6 focuses on the *Guideliner* performance optimization, while Section 4.7 performs evaluation of the *Guideliner* in comparison with other tools. The limitations of the *Guideliner* are pointed out in Section 4.8. Finally, Section 4.9 summarizes the chapter.

4.2 Guideliner – a Tool for Automated Web Usability Evaluation

Most of the tools for automated accessibility evaluation focus on checking the compliance of the page's HTML code to HTML-specific guidelines; evaluation of correspondence to visual guidelines of finally rendered WUI is not possible with these solutions. The novelty of the *Guideliner* is that it evaluates WUI conformance to visual as well as HTML-centric usability guidelines on the final rendered result (after page has finished its loading and all scripts have been applied).

Architecturally, the *Guideliner* is divided into four self-sufficient software modules – name *WUI Evaluation Component*, *Ontology Processing Engine, Ontology Repository* and *Guideliner User Interface* – based on the aspect of functionality they cover. Such design principle is called separation of concerns [116] and is widely used in software development industry when one component has a very limited and narrow scope of functionality it is responsible for. Limiting scope of functionality allows development and testing of each component in isolation from the rest of the system, eliminating failures caused by unintentional side effects. The high-level architecture of the *Guideliner* is presented in Figure 16.

The *Guideliner Core* is a primary back-end component and contains three sub-components: *Ontology Repository*, *Ontology Processing Engine*, *and WUI Evaluation Component*. The *Guideliner User Interface* is a separate component (easy-to-use web application) providing a user interface for managing usability guidelines and triggering the evaluation process. The advantage of this modularization is that every component has its own area of responsibility such as *Ontology Repository* is the source of usability guidelines, while *Ontology Processing Engine* transforms the guidelines to the required format, and WUI Evaluation Component performs the evaluation of input WUI according to the usability guidelines. Each of these modules will be addressed in detail throughout Sections 4.3-4.5. The *XRebel* module is a temporary addition to the Guideliner to analyse the performance of the tool through performance tests presented in Section 4.6.

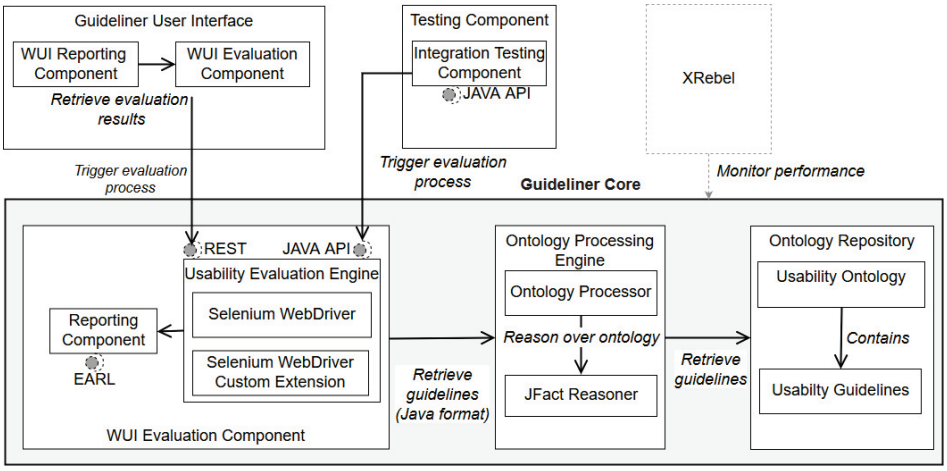


Figure 16. High level architecture of the Guideliner.

4.3 Ontology Repository and Ontology Processing Engine

The role of the *Ontology Repository* is to store the usability ontology (described in Section 3.4) that captures knowledge of web usability domain, consisting of a set of predefined usability guidelines suitable for automatic evaluation of WUIs. The usability ontology has been designed with Protégé Ontology Editor using the standard Web Ontology Language (OWL) version 2 [117] (informally OWL 2) to describe the ontology. OWL 2 is an ontology language for the Semantic Web with formalised meaning for representing the concepts and relationships among concepts. In current case, the usability ontology containing the definitions of usability guidelines is stored in OWL/XML syntax in an OWL file that is fully compliant with OWL 2 Web Ontology Language Structural Specification defined in [117]. The use of the OWL/XML format is justified as it has a comprehensive support by Java OWL API [118] which allows easily transforming guidelines described in OWL/XML to corresponding Java objects. This transformation is required because processing guidelines in native OWL 2 format is not trivial due to the complicated API of OWL 2 language. The purpose of the transformation is to ensure that the guidelines described in the ontology repository will be in an understandable format for the *WUI Evaluation Component.* The design of the usability ontology has been described in Section 3.4, and thereby its discussion within this Section is limited.

The transformation task is carried out by the *Ontology Processing Engine* (Figure 16) that consists of *Ontology Processor* and *JFact Reasoner*. *Ontology Processor* is responsible for loading and processing the ontology, while, *JFact Reasoner* allows automatic classification over the captured concepts. *JFact Reasoner* is a Java port of FaCT++ reasoner [119] having full compatibility with OWL API library. *JFact Reasoner* is used to compute actual class hierarchy for defined classes according to definitions given in the ontology (e.g. the reasoner is used to infer all usability guidelines suitable for mobile devices).

Ontology Processing Engine provides a simplified, well documented, and clear Java API for retrieving usability guidelines from *Usability Ontology*. It provides the following public methods (refer to *Guideliner* class diagram presented in Appendix 4 for more details on the class formation of the *Guideliner core*) for retrieving usability guidelines:

- *reloadOntology* – a method that fills Java objects with the latest version of usability guidelines defined in OWL/XML file; this method is called by *WUI Evaluation Component* every time OWL/XML file is modified (e.g. when existing usability guidelines are updated or the new ones created).
- *findAllCategoriesOfUsabilityGuidelines* – a method to retrieve all possible categories of usability guidelines such as mobile or desktop guidelines. The method is called by the *Guideliner User Interface* to present all categories of guidelines supported by the *Guideliner.*
- *findUsabilityGuidelinesByCategory* – a method to find all usability guidelines of the selected category. For instance, the method can be called to retrieve all usability guidelines of mobile or WCAG category.
- *retrieveUsabilityGuidelineByName* – a method to retrieve a usability guideline by its name.

Overall, the guidelines stored in the *Ontology Repository* in OWL/XML format are transformed by the *Ontology Processing Engine* into Java-based guideline descriptions which will then be consumed by *WUI Evaluation Component* for WUI evaluation.

4.4 WUI Evaluation Component

The *WUI Evaluation Component* is a component of the *Guideliner* tool that is responsible for assessing WUI conformance to usability guidelines stored and described in the *Ontology Repository*. This component consists of the following entities: *Usability Evaluation Engine* performing the evaluation of WUI conformance to usability guidelines, and the *Reporting Component* responsible for generating usability evaluation reports. The WUI Evaluation component contains evaluation logic including opening the browser instance with the WUI being evaluated and comparing the conditions set in usability guidelines with actual results extracted from WUI.

The *WUI Evaluation Component* is a central unit of the *Guideliner* tool being responsible for checking the conformance of measurable values defined in usability guidelines to the values extracted from the evaluated WUI.

Before continuing with the discussion of the different subcomponents of the *WUI Evaluation Component*, let us analyse the two main possible approaches for WUI evaluation against usability ontology – performing WUI evaluation within the usability ontology using reasoning, or exploiting the knowledge described in the ontology to carry out evaluation in separate.

In the first case, the evaluation process is carried out using the ontology, reasoning capabilities and rules described in it. A Knowledge Base Reasoner (e.g. Fact++ [119] or Hermit[1]) is used to evaluate WUI conformance to usability guidelines. For this, actual values of WUI attributes are extracted and delegated to the Knowledge Base Reasoner. Figure 17 outlines a possible architectural layout for this approach. As can be seen, with this setup, *WUI Evaluation Component* is responsible for parsing the user interface that is being evaluated into WUI attributes and then transferring them for the evaluation to the *Ontology Repository,* using *Ontology Processing Engine* as a mediator. *Ontology Repository* is responsible for storing usability guidelines (consisting of *TBox* and *ABox* (refer to Section 3.4), performing evaluation using *Knowledge Base Reasoner* and storing the evaluation results *RDF Store*. The purpose of *RDF Store* is to keep the evaluation results and to provide possibility for retrieving them through semantic queries. In that

[1] http://www.hermit-reasoner.com

case, *WUI Evaluation Component* would use *Ontology Processing Engine* to execute SPARQL Protocol and RDF Query Language (SPARQL[1]) queries in order to retrieve evaluation results from *Knowledge Base Reasoner*. SPARQL is a semantic query language for RDF databases used for manipulating and storing the data. Thus, the responsibility of *Ontology Processing Engine* is to fill individuals of *ABox* with actual values extracted from WUI being evaluated, whereas responsibility of *Knowledge Base Reasoner* is to check the consistency of the ontology modified with values added from WUI.

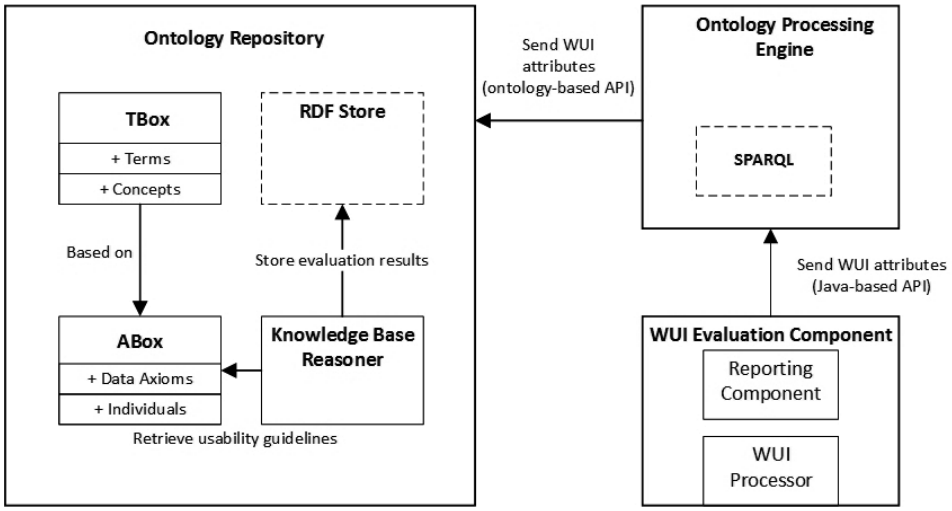


Figure 17. Architecture of the Guideliner using Knowledge Base Reasoner for evaluation.

The advantage of the latter approach is that *WUI Evaluation Component* is used only for extracting the WUI attributes (e.g. width of a button, text length of a link, etc.) and filling the individuals of the ontology with extracted properties. The whole evaluation logic is done by *Knowledge Base Reasoner*; the evaluation results, in turn, are stored separately from the evaluation logic. A major drawback of this approach is that in order to obtain human-readable evaluation results from *Knowledge Base Reasoner*, (e.g. element being evaluated, expected and actual value of measured characteristic and reason of violation) SPARQL queries must be used. The drawback of SPARQL queries is that the result of the queries should be processed again by *Reporting Component* to provide the evaluation result in required format. It means that all usability guidelines defined in ontology must be processed twice: first by *WUI Evaluation Component* to define what WUI characteristics should be extracted, and second by *Knowledge Base Reasoner* to perform the evaluation itself.

Overall, the most time-consuming part of retrieving WUI attributes is done on the side of *WUI Evaluation Component*, and then the retrieved WUI attributes are transferred to *Ontology Repository* that performs assertion of retrieved values and actual values of WUI attributes. In addition to the duplication of processing usability guidelines mentioned before, evaluation results are processed twice: firstly, by *Ontology Processing Engine* to retrieve them from ontology and transform to the Java objects, and, secondly by *Reporting Component* to transform evaluation results to the required format. Thus, the described approach requires redundant mappings and transformations between

[1] https://www.w3.org/TR/rdf-sparql-query/

components and requires creating additional SPARQL queries to retrieve evaluation results. For example, in order to create usability evaluation report, RDF Store should be queried to retrieve usability guidelines, evaluation results and then, additionally, transform all results to the required format. For the aforementioned reasons, the described architecture has not been selected for the *Guideliner* development.

To overcome the limitations of the previously analysed approach, it was decided to describe domain knowledge in ontology and exploit this knowledge described in the ontology to carry out evaluation in separate. With this approach, the central task of evaluation is performed by the *Usability Evaluation Engine* (Figure 16), which then must satisfy the next aspects to carry out this task:

- Be able to evaluate visual aspects of finally rendered WUI such as position of elements on the screen, foreground and background colour of elements, presence of scrolling and layout of elements on the page.
- Have full support for checking the consistency of HTML code including the comprehensive API for retrieving the elements by HTML tags and HTML attributes, and the possibility to trace the order of HTML elements.
- Have compatibility with various browsers, operating systems and platforms to verify that the conformance to usability guidelines is not affected by external factors like browser type or version, or device platform.

The aforementioned aspects clearly outline that WUI processing and evaluating is not a straightforward task as all those aspects should be considered. For this reason, the most logical approach is to perform the whole evaluation of WUI conformance to the guidelines as a part of *WUI Evaluation Component.*

Taking this approach clearly separates the responsibility of the individual components of the *Guideliner,* and also decreases the communication between them. Thereby, *Ontology Repository* is responsible only for storing usability guidelines; *Ontology Processing Engine* only for transforming those guidelines into Java objects, and *WUI Evaluation Component* for performing the evaluation of WUI conformance to usability guidelines and providing evaluation report in the required format.

The high-level architecture of the *Guideliner* presented in Figure 16 became the structural fundament for the physical implementation. *WUI Evaluation* component is Spring Boot[1] based Java application running in Tomcat[2] container. Java language was used because of a high proficiency of the author in that language (C# or other languages suitable for Web development that has libraries for integrating with ontology can be also used for that purpose). Spring Boot was selected as it is a de facto standard[3] for developing new Model View Controller application in the Java world. An advantage of Java is OWL API that provides API for processing ontology files; though OWL API also is ported to other languages such as C# and Python.

The focal responsibility of *Usability Evaluation Engine (a part of WUI Evaluation Component)* is to parse final rendered user interface and extract the characteristics of WUI components being evaluated (e.g. contrast, alternative text, etc.). There are two widely used WUI frameworks - Selenium WebDriver[4] and PhantomJS[5] – that provide a full-stack of instruments and operations needed for automated testing of user interfaces,

[1] http://spring.io/projects/spring-boot
[2] https://tomcat.apache.org
[3] https://www.developer.com/java/ejb/what-is-spring-boot.html
[4] https://www.seleniumhq.org/projects/webdriver
[5] http://phantomjs.org

and can potentially be used as a mechanism to automate the evaluation process of HTML-centric guidelines and visual aspects against usability guidelines.

PhantomJS is a headless WebKit scriptable browser providing a JavaScript API. It has a built-in native support for multiple web standards (including JSON, CSS selector and DOM handling), and runs perfectly on command line. PhantomJS is more suitable for sanity check or smoke tests than for extensive UI tests as it does not support cross-browser consistency and the accuracy of elements visual representation is not precise enough to be the solution for extensive automated WUI usability testing. The advantage of PhantomJS compared to Selenium WebDriver is less overhead and better performance (it uses headless browser that is around 2 – 3 times quicker than actual browser [120]). The cost it pays for that is the absence of compatibility with browsers and less advanced evaluation possibilities of visual aspects of WUI.

Selenium WebDriver, on the other hand, is a tool that provides a full-stack API for automated WUI testing and verification to ensure that WUI behaves in an expected way. Selenium WebDriver is distributed as a standalone library and has full support for most common programming languages like Java, C#, Python, Ruby, PHP and others. It is compatible with various browsers and their versions such as Firefox, Internet Explorer, Chrome etc. The main advantage of the Selenium WebDriver is the ability to execute tests on multiple browser platforms with high accuracy and full compatibility with HTML/CSS/JavaScript standards. Selenium WebDriver provides operations and commands for fetching the page, locating elements on the fetched pages, clicking elements, filling inputs, moving between windows and others, and has a clear API for finding HTML elements by the tag name, by class name, link text, XPath. Moreover, it allows determining the position, size and colour of elements on the page. Therefore, it provides the full-stack of instruments and tools that are needed for evaluating various usability guidelines automatically. The common way to integrate Selenium WebDriver into existing Java project is to include Selenium WebDriver as Java Archive (JAR). For instance, this can be done automatically by using a dependency management tool such as Maven[1].

In the current case, better performance and less overhead may be disregarded if the accuracy of evaluation of visual aspects of WUI is more comprehensive. The accuracy in evaluation of visual aspects has a critical impact on the overall quality of usability evaluation. For this reason, Selenium WebDriver was chosen as a base for the *Guideliner*. Because of Selenium WebDriver use, the *Guideliner* tool supports evaluating usability guidelines in various common browsers[2] like Mozilla Firefox (v. 47+), Google Chrome (v. 60+), Internet Explorer (v. 8+), Microsoft Edge (v. 12+) and Safari (v. 5.1+) running on Windows, Linux, Android and iOS platforms. Moreover, it is possible to simulate testing on different devices with various resolutions using for example Browserstack[3] – a service that runs Selenium tests on more than 2000 browsers and provides native browser experience.

Let us now focus back on the process of WUI conformance evaluation with the taken approach (Figure 18) in the context of WUI. The *WUI Evaluation Component* is capable of evaluating the properties of HTML elements such as length of element value, the existence of alternative text, value of element attribute etc. As a novelty compared to

[1] https://maven.apache.org

[2] Stated as of 2018

[3] https://www.browserstack.com

existing solutions, this component is also capable to evaluate visual characteristics of finally rendered WUI including the position of elements on the screen, foreground and background colour of elements, presence and the length of scrolling, layout and the order of elements – through the use of the Selenium Web Driver. All the capabilities are provided through adapters. Adapter-based approach for the tool development was selected as adapters are pluggable and simple WUI-element specific components that can be easily added if needed. The methods in adapters are divided into two steps: initially, adapter retrieves all elements from WUI and their corresponding attributes being evaluated using Selenium WebDriver; afterwards, it checks if retrieved elements comply with usability guidelines or not. Figure 18 shows that initially the *Guideliner User Interface* triggers evaluation by calling *UsabilityEvaluationController* which in turn calls *Ontology Processing Engine* to retrieve guidelines for evaluation and then triggers the evaluation by calling the *AdapterService, which* identifies the element on WUI (and its corresponding characteristics) that should be evaluated. After this, the *AdapterService* calls evaluation adapter (adapters are responsible for performing the evaluation of certain WUI components such *LinkAdapter* is responsible for evaluation link specific characteristics) providing the characteristics of the element as a parameter to the adapter (e.g. contrastMinValue = 4.5). The corresponding adapter retrieves the actual values of the WUI element characteristic being evaluated using Selenium WebDriver and asserts if the retrieved values are corresponding to the value defined in the usability guideline. For instance, if the length of the *Link* text is evaluated then the *LinkAdapter* asks Selenium WebDriver to find all *Links* and calculate the length of each *Link* text. Afterwards, *LinkAdapter* checks if the values returned by Selenium Web Driver correspond to the value defined in the usability guideline. If the link text of all *Links* corresponds to the length of the text in the usability guideline then a success response is generated, otherwise, a failure response is generated.

In addition to evaluation of HTML element properties, the *Guideliner* addresses the evaluation of visual usability guidelines that requires more functionality for the processing of visual elements than Selenium WebDriver proposes. Selenium does not provide clear API for evaluating all kind of visual characteristics, for this reason, the author extended Selenium API with custom methods (Figure 18 *Selenium WebDriver Custom Extension*) providing additional operations for evaluating scrolling, contrast rate, visual position, order of elements and the distance between elements on the page, not present in the original tool.

To exemplify the evaluation of visual usability guidelines, let us consider the following guideline from Nielsen Norman Group[1]: "*Horizontal scrolling is not allowed",* which cannot be evaluated by any of the existing tools for automated usability evaluation as it is not possible to verify the existence of horizontal scrolling only by processing HTML code. To evaluate the guideline, it is necessary to inspect the final rendered page and scroll it horizontally and check if it is allowed. The *Guideliner* is capable of evaluating such guideline using the Selenium *JavascriptExecutor* that helps to detect if scrolling is allowed and scroll the page horizontally if it is possible. Or, to evaluate the following guideline "*The distance between links should be at least 30 pixels for mobile devices"* defined by Google[2], the *Usability Evaluation Engine* retrieves all links on the web page, and then checks that the real distance is not less than the value defined in usability guideline.

[1] https://www.nngroup.com/articles/horizontal-scrolling

[2]https://developers.google.com/web/fundamentals/accessibility/accessible-styles#multi-device_responsive_design

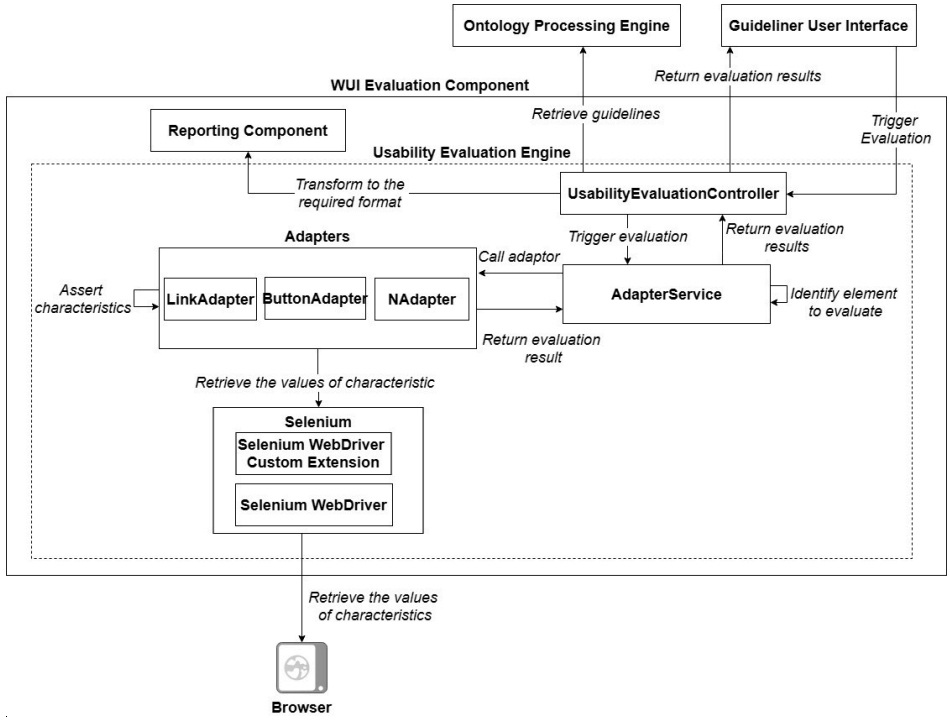


Figure 18. The process of evaluating WUI conformance to usability guidelines.

Through the use of Selenium WebDriver the *Usability Evaluation Engine* in the *Guideliner* tool supports evaluating visual usability guidelines as well as HTML-centric usability guidelines, and delivers a clear advantage over existing evaluation tools.

4.4.1 Usability Evaluation Report

An essential outcome of every usability evaluation carried out, including both manual and automated usability evaluation, is the report that presents the results of the evaluation. Properly structured report clearly outlines the elements violating guidelines, reason of violation and probable cause of the problem making it easier for developers to understand and fix the problem.

The reporting component of the *Guideliner* tool (Figure 16) provides two reporting APIs including an EARL Schemes based API[1] for exchanging evaluation results between other usability evaluation tools, and a REST API for communicating with *Guideliner User Interface*. EARL is a machine-readable format for expressing results for usability evaluation based on the designed vocabulary for expressing test results. EARL is concentrating on describing the results of usability evaluation; evaluation criteria, requirements and procedures are covered by EARL rather superficially.

EARL is intended for developers of Web accessibility evaluation and validation tools simplifying the exchange of test results between evaluation tools in an environment-agnostic way. EARL is the only industry option that addresses the problem of unification of usability evaluation results that is the reason why the *Guideliner* uses EARL for expressing test results. Every evaluation result is defined as an assertion in EARL.

[1] https://www.w3.org/TR/EARL10

An example of automated usability evaluation report in EARL generated by the *Guideliner* tool is presented in Appendix 5. The EARL report consists of the following information:

- Assertor – contains the information about the runner of the test: it can be the person performing the evaluation or the tool for automated usability evaluation. EARL supports five types of assertors: Software, Agent, Person, Organization and Group. Assertor of type Software is used in EARL reports as the evaluation is performed automatically by the tool.
- Test Subject – contains web content, software or response being evaluated. In the context of the *Guideliner*, Test Subject contains URL of web application being evaluated.
- Test Criterion – contains the specification such as set of guidelines, tests or any other testable elements that test subject is evaluated against. In the context of the *Guideliner*, Test Criterion (defined as a TestCase) presents the description of usability guideline.
- Test Result – contains the outcome of test including the information if the test failed or passed and the description of a test result.

The unified format of EARL is strictly defined and does not allow additional elements such as screenshots of the elements violating the guideline and the type of elements under evaluation (both elements are required and supported by the *Guideliner User Interface* described in Section 4.5) to be described. In general, it is common for all existing usability evaluation tools to have a custom format of usability evaluation reports that satisfies the needs of a particular tool, and so does *Guideliner*.

The main consumer of the *WUI Evaluation Component* is the *Guideliner User Interface* that uses it for triggering the evaluation process, retrieving guideline, and presenting evaluation results. EARL is designed only for representing evaluation results (for exchanging between the tools) and it is not intended for being a protocol for all kinds of communication (like retrieving the guidelines, creating new usability guidelines) between the components of the tool. Thus, in order to share the data between the *Guideliner User Interface* and *WUI Evaluation Component* an internal custom format called Guideliner Report Format (GRF) of usability evaluation report (based on EARL) was defined and used as a communication protocol between client and server side. Other external systems should use the *Guideliner* EARL endpoint for performing the evaluation with the *Guideliner*. GRF added the next properties to EARL: screenshot of failed element, text and description of the element violating the guideline. Table 2 presents the structure of the internal usability evaluation report containing a brief description of each element in report and its sample value. An extract of usability evaluation report based on this format is presented in Figure 19 containing the results of the tests run on https://www.etis.ee for the WUI conformance evaluation on the guideline presented by Google Mobile research group[1]: "*There should be no other links within 48 CSS pixels (roughly 7 mm)"*. The evaluation result is presented in JavaScript Object Notation (JSON) format as JSON is a native communication format for *Guideliner User Interface* (Section 4.5) – a common format used for data interchange between JavaScript-based applications. The *Guideliner* endpoint also allows using XML if needed. The example on Figure 19 shows that the evaluation failed (line 4) and contains the list of the elements that violate the guideline (lines 12 – 24) with the violation reason and the name of the violated element's

[1] https://developers.google.com/speed/docs/insights/SizeTapTargetsAppropriately

screenshot. Based on GRF, the *Guideliner User Interface* generates a visual report for human evaluators (Figure 22 discussed further in Section 4.5).

Table 2. Structure of the internal usability evaluation report format GRF.

Element Name	Element Description	Sample Value
elementType	Type of evaluated element	Navigation, Link, Paragraph
Result	Identifies if test passed or failed	SUCCESS, FAIL
Guideline	Guideline being evaluated	N/A
guideline.code	Guideline unique identifier in the ontology	14-9_OptimizeImages
guideline.name	Brief name of the guideline	All images on the screen should be optimized
guideline.description	Detailed description of the Guideline	The maximal size of the image should be 1 MB
failedElements	Elements violating guideline	N/A
failedElements.path	Path for downloading screenshot of element violating the guideline	screenshot42.jpg
failedElements.description	Reason for failure	Actual size of images is 2 MB, expected is less than 500 Kbytes.
failedElements.text	Text of the element violating guideline	Press to continue

To sum it up, the *Guideliner* Reporting Component provides different ways of formatting evaluation results including EARL for (automated) exchanging evaluation results between other usability evaluation tools, and REST API for communicating with the *Guideliner User Interface over the GRF*, delivering additional features compared to the EARL format.

4.5 Interfacing with the Guideliner

Having discussed all other components of the *Guideliner* tool (Figure 16), it is now time to turn to the *Guideliner User Interface* – a web user interface for managing usability guidelines and triggering the whole evaluation process. The intended users of the *Guideliner* are technical (including developers and quality assurance specialists) and business (including analysts and product owners) users. Technical users can use the *Guideliner Core REST API* to trigger the evaluation process and receive evaluation results. In general, business users do not have required technical background to use REST API as they are mostly used to visual interfaces. For this reason, a separate web application was designed that makes use of the *Guideliner Core REST API*. This REST API provides direct access to the *Guideliner Core* to invoke evaluation process; in both cases, the process of WUI evaluation is absolutely the same on regardless of how this process is invoked, i.e., via visual user interface or using the direct call. The structure of the Guideliner REST API

and Guideliner Report Format GRF that the REST API supports have been discussed previously in Section 4.4.1 in more details.

```
[
  {
    "elementType": "LINK",
    "result": "FAIL",
    "failedElements": [
      {
        "type": "LINK",
        "text": "A+",
        "description": "Link with text A+ is very close to A-",
    "pathToElement": "screen1503742712142.5.png"
      },
      {
        "type": "LINK",
        "text": "A-",
        "description": "Link with text A- is very close to Est",
        "pathToElement": "screen1503742712142.6.png"
      },
      {
        "type": "LINK",
        "text": "Est",
        "description": "Link with text Est
        is very close to link with text Events",
        "pathToElement": "screen1503742712142.7.png"
      }
    ],
    "guideline": {
      "code": "21-01_DistanceBetweenLinksShouldBeEnough",
      "name": "There should be no other Links within 48
      CSS pixels (roughly 7 mm)",
      "description": "There should still be no other
      tap targets within 7mm (32 CSS pixels),
      both horizontally and vertically, so that a user's finger
      pressing on one tap target will not inadvertently
      touch another tap target."
    }
  }
]
```

Figure 19. An example of usability evaluation report in JSON format for the guideline checking the distance between elements. Test run on https://www.etis.ee.

The Guideliner User Interface is a single-page web application (Figure 20) that has been built based on Angular JavaScript framework using Bootstrap – stylesheets containing an extensive list of components for designing client web applications. Alternatively, any other front end JavaScript framework could have been selected such as ReactJS. In fact, ReactJS and Angular are the most widely used JavaScript frameworks according to Stackoverflow Developer Survey Results, both technologies being powerful and flexible. As far as there are no obvious advantages in both technologies the author decided in favour of Angular because of the extensive experience and better expertise in Angular web application development.

Instead of using pure JavaScript, TypeScript[1] was used as a language for *Guideliner User Interface* development. TypeScript is typed language that compiles into JavaScript. It is a common approach to write web applications using Typescript as the Angular framework itself is written in TypeScript meaning that there are no limitations to using Typescript instead of pure JavaScript.

[1] https://www.typescriptlang.org/docs/home.html

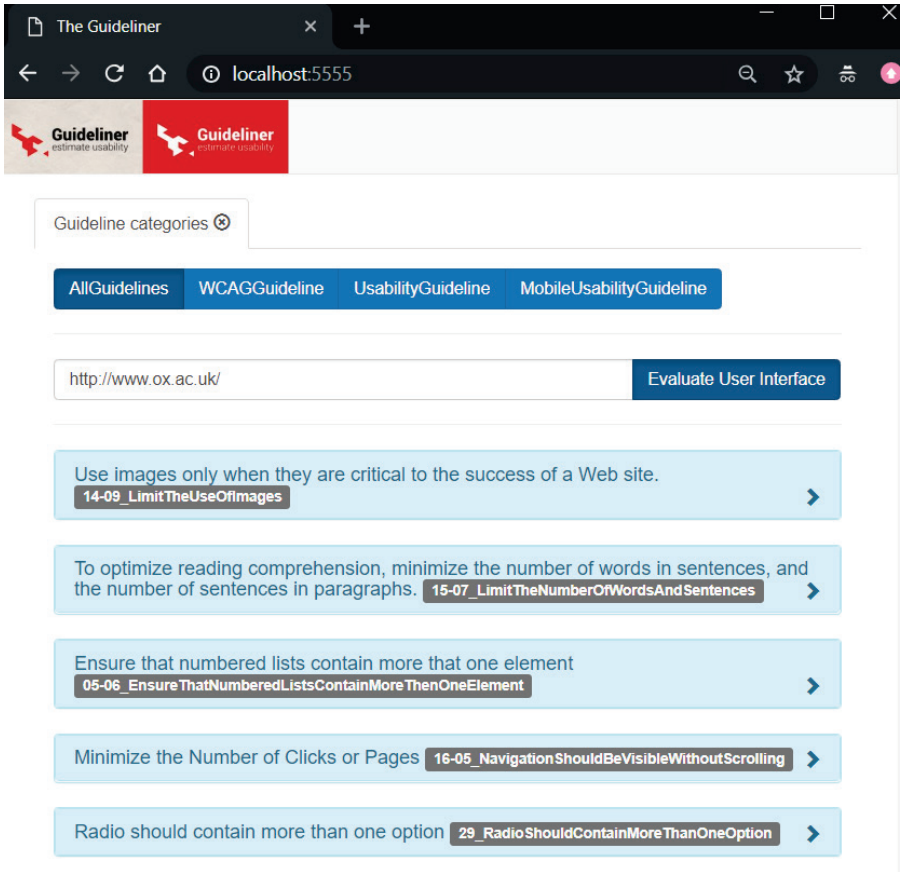


Figure 20. Screenshot of the Guideliner User Interface.

The user interface application has been developed based on Angular Seed[1] (a skeleton for Angular based applications served together with development and testing tools) providing fast, reliable and extensible starter for the development of Angular projects. It includes multiple useful features like Ahead-of-Time compilation, sample unit tests and others. This is important, as these features enable immediate validating of the correctness of the code. All aforementioned features increase the speed and quality of the *Guideliner User Interface* development.

The *Guideliner User Interface* allows the person performing WUI evaluation to select the category of guidelines for evaluation, trigger evaluation process and analyse evaluation results (Figure 20). The usability evaluation process is initiated from the *Guideliner User Interface* when user triggers the evaluation; Figure 21 illustrates the steps of the usability evaluation process. Initially, a user selects the category of usability guidelines (e.g. Mobile usability or WCAG guidelines) and the URL of WUI to be evaluated. Figure 20 presents a screenshot of the view containing the sets of guidelines to be evaluated such as mobile usability guidelines, WCAG guidelines and usability guidelines. After selecting the category of usability guidelines to be evaluated, the guidelines corresponding to the selected category appear. Each guideline contains title, description and success criteria. Then, user enters the URL of the web application to be

[1] https://github.com/angular/angular2-seed

evaluated and initiates the evaluation process by clicking *Evaluate User Interface* button. On the click, the *Guideliner User Interface* calls *WUI Evaluation Component* via REST providing the category of usability guidelines being evaluated and URL as input parameters. Based on the category provided *Usability Evaluation Component* calls *Ontology Processing Engine* to retrieve corresponding usability guidelines to be evaluated. Afterwards, *WUI Evaluation Component* performs the evaluation, and then calls *Reporting Component* to generate the usability evaluation report in JSON format that is sent back to the *Guideliner User Interface*; the *Guideliner User Interface* visualizes and structures the report to make it easier to read, highlighting the required information including the failed and passed usability guidelines (Figure 22). Violated guidelines are highlighted in red colour; passed guidelines in green colour. The progress bar above shows the ratio of failed and passed tests. Evaluation results provide full information of evaluated guideline including name, code, and description.

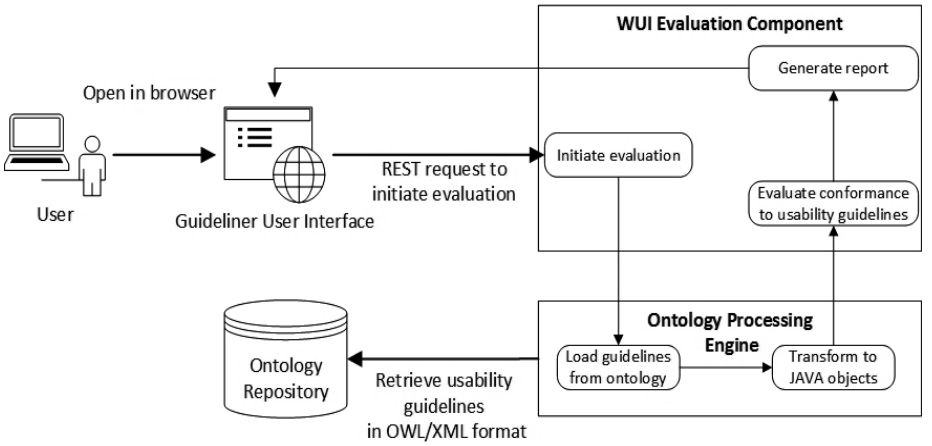


Figure 21. Steps of usability evaluation with the Guideliner.

Clicking the link *Open Failure Report* (Figure 22) opens a dialogue window containing a detailed view about detected problems: a screenshot, element text and the type (e.g. link, button, input) of the element violating the guideline. Figure 23 provides an example of the dialogue window that contains a human-readable explanation of the failure reason including expected and actual value. Explanation of violation informs WUI developers about the usability issue, emphasizing which characteristics should be changed and how much it should be changed to satisfy the usability guideline. The screenshot is optional for the evaluation report as it is not always possible to depict the element violating the guidelines. In particular, WUI elements that violate guidelines that cover the consistency and validity of HTML code in general do not contain images (but they contain HTML code snippet with incorrect element). On the contrary, WUI elements that violate guidelines evaluating the visual characteristic of WUI always contain a screenshot of the failed element making it easier to understand the reason for failure. HTML code snippet may not be useful when visual usability guidelines are violated as the violation can be caused by CSS styles or may be caused by multiple HTML elements.

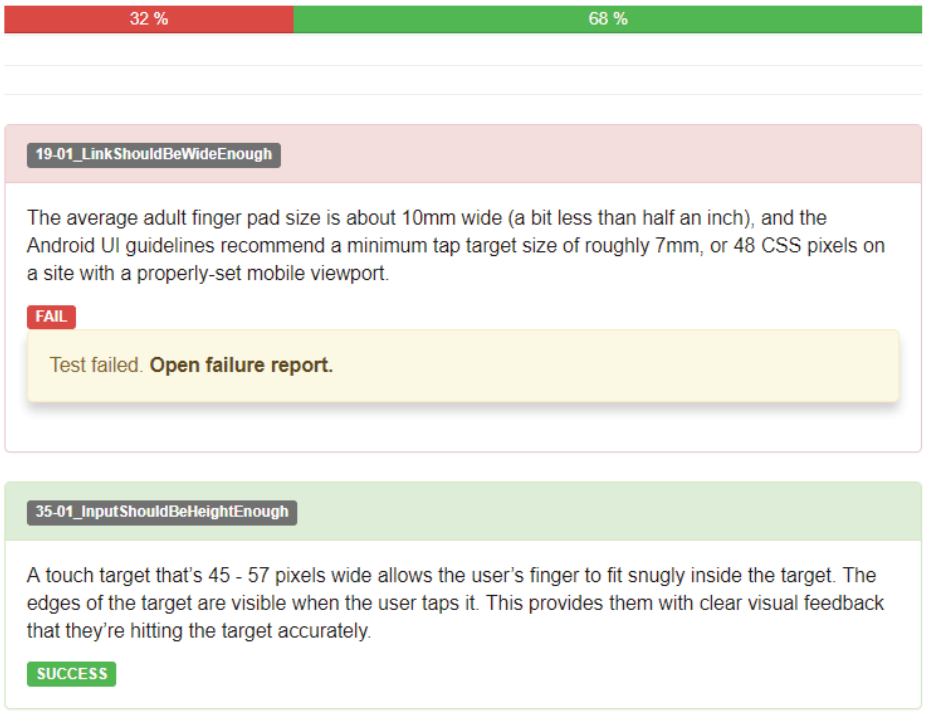


Figure 22. Screenshot from the Guideliner User Interface – an excerpt contains an example of a passed and a failed usability evaluation result.

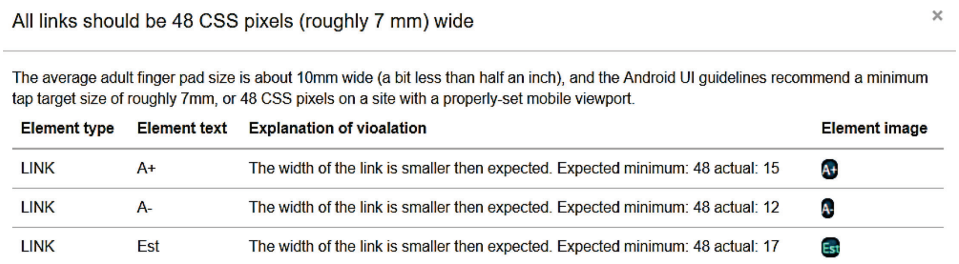


Figure 23. Screenshot from the Guideliner User Interface – a dialog presenting additional information about a failed guideline.

4.6 Performance Testing and Optimization of the Guideliner Tool

In order to understand what operations can potentially slow down the evaluation process or even crash the *Guideliner* tool, performance testing, which is a process of measuring and calculating performance metrics of implemented software [121], was carried out. Also, performance evaluation allows to detect operations that are not thread-safe and affect parallel evaluation of different elements of the same WUI. The next aspects were considered to evaluate the performance of the *Guideliner*: resource intensive operations, response time of the system in case of parallel requests, and high load on components.

To evaluate the performance of the *Guideliner* XRebel[1] was used. XRebel is a tool for analysing the performance metrics of the application including slow methods, resource problems, hidden exceptions, slow database queries and other aspects affecting the performance of the application. The advantage of XRebel over other existing solutions (e.g. AppDynamics[2], Plumbr[3]) for monitoring performance is that it detects performance issues already during development whereas other solutions are concentrating on finding performance defects during the testing phase. XRebel analyses the flow of system and detects the operations that consume the most resources (e.g. time, memory). Afterwards, XRebel returns these operations as candidates for performance issues. XRebel was temporarily integrated into the *Guideliner* with a purpose to identify the performance issues (Figure 16).

The purpose of the performance testing was to answer the following questions:

1. What are the operations that slow down the evaluation process?
2. Can the time consumed for usability evaluation by the Guideliner be decreased?

In order to analyse the performance of the *Guideliner*, ten WUIs from different areas, different target users, complexity and design were selected. The purpose of the selection was to include heavy-weight applications that contain multiple images, advertisements and interaction components as well as light-weight applications with just a few images and some text. The data collected in this experiment was also used for the author's research presented in [122] that focused on evaluation of the *Guideliner* capabilities. The following web applications and portals were selected for probation conducted 28.06.2018:

- Republic of Estonia Road Admission Portal[4], which has the same design template as all other ministry web pages of the Republic of Estonia.
- E-government Portal[5] being the primary gateway to the online governmental services in Estonia.
- Government Real Estate Portal[6] providing public sector real estate services. It was selected as its WUI does not contain many elements enabling to analyse the results for performance tests on WUI with smaller number of elements on the screen.
- Government information system management portal[7] containing repositories for public e-government services. It was selected to diversify WUI based on target users as the main visitors of that portal are the online service providers and the developers who are integrating with Estonian online services.
- Estonian Research Information System[8] providing information about Estonian researchers and their research activities and projects. The portal was selected as a representative of the academic field delivering mostly structured data.

[1] https://zeroturnaround.com/software/xrebel
[2] https://www.appdynamics.com
[3] https://plumbr.io
[4] https://www.mnt.ee/eng
[5] https://www.eesti.ee/en
[6] http://www.rkas.ee/en
[7] https://riha.eesti.ee/riha/main
[8] https://www.etis.ee/?lang=ENG

- Republic of Estonia Information System Authority[1] responsible for the development and maintenance of X-Road and www.eesti.ee portal. It was selected as it is one of the main public sector IT units of Estonia.
- Estonian News Portal Delfi[2] being one of the most visited online news portals in Estonia. It was selected because of the number of visits, the frequency of content update and the amount of content on the screen.
- National Aeronautics and Space Administration (NASA) web page[3] providing information about America's space agency. It was selected as it is one of the most popular science web sites that was awarded multiple times as a highly ranked scientific website[4].
- CNN news portal[5] is a popular news website with high daily visitors rate and update frequency.
- BBC news portal[6] providing latest news and headlines being in TOP 10 most visited web sited in the UK.

The selected WUIs were evaluated with the *Guideliner*. A series of experiments were carried out where for each of the selected website WUI, the evaluation was carried out six times: three times before the performance optimization and three times after the identified performance problems were resolved. The results of the evaluation between series varied in the range of five percent – as a result, the author concluded that because of the small deviation the number of experiments was sufficient, as the only external factor to influence the results was the Internet connection speed; the speed of WUI processing depends only on how fast the WUI is being loaded and how fast Selenium retrieves the elements for processing. All the experiments were carried out on the same platform, on the same machine, and under the same conditions. The Internet connection speed was stable during the whole experiment varying between 75 – 85 Mbps measured by a special script developed for that purpose, which was periodically calling (every ten seconds) SpeedOf.Me API[7] (service for measuring connection speed) and documenting the connection speed.

Overall, a series of 60 experiments were carried out. A special Java program was developed that was iterating over all URLs of websites selected for the experiment and was calling *Guideline Core REST API* to evaluate the conformance of WUI to usability guidelines. After each experiment, the total evaluation time and XRebel performance monitoring report were stored by developed Java program.

The performance evaluation reports produced by XRebel were parsed after each evaluation for further analysis of results. Figure 24 outlines the results of *Guideliner* evaluation, measured with the metric processed guidelines per second (*gps*) before and after the code optimization. After the initial evaluation, the code was optimized for the most time and resource intensive operations based on the XRebel evaluation results to decrease the execution time of those operations. The metric *gps* was chosen because it

[1] https://www.ria.ee/en
[2] http://www.delfi.ee
[3] https://www.nasa.gov
[4]https://solarsystem.nasa.gov/news/419/nasa-web-sites-social-media-honored-in-2018-webby-awards/
[5] http://edition.cnn.com
[6] https://www.bbc.co.uk
[7] https://speedof.me/api.html

clearly outlines the *Guideliner* processing speed in the unit of time, indicating the average processing throughput.

The evaluation results returned by XRebel pointed out eleven methods in the source code of the *Guideliner* that required more time for execution in comparison with other executed methods during the evaluation. The following methods of the *Guideliner* implementation showed extremely poor results during the evaluation:

1. Operations for making screenshots of the elements violating usability guidelines (method: *Screenshoter.makeScreenshot*; average execution time: 700 milliseconds). To optimize the operation, the screenshot of the whole WUI was taken before the evaluation and cached; in case the violated element was detected the cached screenshot was used to make the image of the violated element. Overall, the time of making a screenshot has decreased in average 50% after optimization.
2. Opening browser window for each evaluated usability guideline (method: *WebDriver.get(url)*; average execution time: 2100 milliseconds). To solve that problem, the browser windows was opened before the WUI evaluation and was closed only after the full evaluation was finished.
3. Operations for evaluating the distance between elements (methods: *ButtonAdaptor.evaluateDistance()*, *LinkAdaptor.evaluateDistance()*; with average execution time: 300 milliseconds). To decrease the evaluation time, the coordinates of elements were cached locally and not retrieved from Selenium API every time
4. Operation for evaluating contrast rate (method: *ContrastEstimator.estimate()*; average execution time: 200 milliseconds). The contrast rate was calculated based on the time-consuming image-processing methods. Instead, the HTML and CSS based method was used being around 20% more efficient with the same evaluation accuracy.

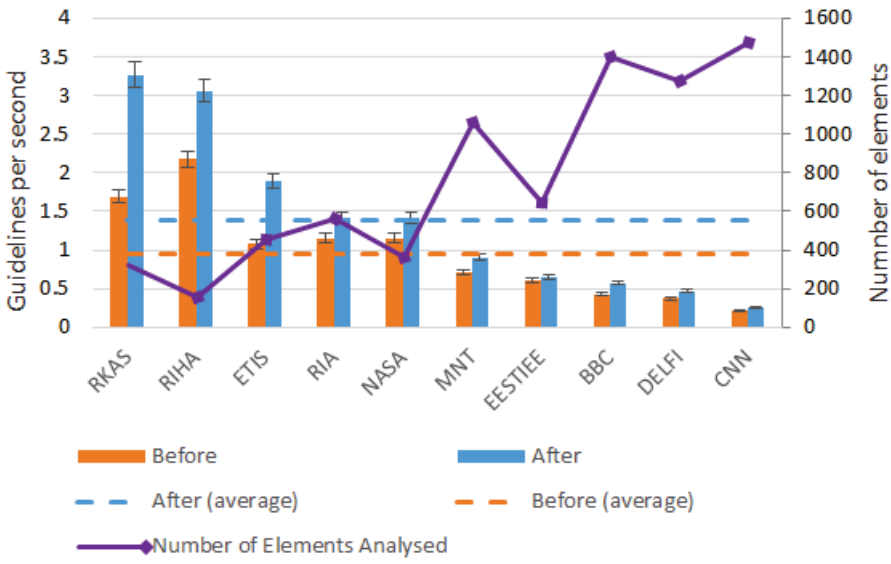


Figure 24. Guideliner performance testing results before and after code optimization.

After the performance problems were fixed, the same experiment was conducted once again (blue columns in Figure 24 represent the evaluation results). The results showed that the response time of certain methods decreased by around 50%.

For example, operation for taking and storing the screenshot of element violating the guidelines had taken around 700 milliseconds. After the screenshot making logic was reviewed and moved to the asynchronous thread, screenshot making time has decreased to around 350 milliseconds. Overall, as expected the average processing *gps* increased after the performance optimization as shown in Figure 24 (refer to blue column). In particular, the tangible shift in absolute numbers of *gps* was measured for WUI with smaller number of elements. For example, RKAS *gps* increased from 1.6 to 3.3 *gps* whereas CNN *gps* increases from 0.2 to 0.3 *gps*. Nevertheless, the increase in percentage or relative numbers is homogeneous between all WUI.

While exploring the correspondence, the correspondence between *gps* and the number of HTML elements on WUI was detected. Obviously, the evaluation time depends on the content size, amount and size of graphics, size of CSS and JavaScript files, and overall amount of HTML elements on WUI. As such, it was observed, that the *gps* rate was higher for WUI with smaller number of HTML elements on the screen – for RKAS (contains 318 HTML elements) 1.68 *gps*, whereas for CNN the same indicator was 0.22 (contains 1478 HTML elements). In this experiment, the number of elements on the page was measured with Selenium WebDriver. As every HTML element of the WUI being evaluated needs to be checked, the amount of different HTML elements is in clear correlation with the total evaluation time (Figure 24). Such trend becomes even more considerable when visual usability guidelines are evaluated for the reason that the evaluation of the element size, distance between element or the contrast of elements are more resource-intensive than evaluation of HTML-specific guidelines that are based on parsing of HTML code.

Overall, the performance testing indicated certain methods in the source code of the *Guideliner* that required more time for execution in comparison with other executed methods during the evaluation. The performance of identified resource-consuming methods was improved based on the performance test results and re-evaluated with repeated performance tests. Also, the correspondence between *gps* and the number of HTML elements on WUI was detected: it was observed that the *gps* rate was higher for WUI with smaller number of HTML elements on the screen. The next step is to compare the *Guideliner* with other tools based on wider range of characteristics.

4.7 Guideliner and Other Similar Tools

Software verification is an important part of software engineering, which is responsible for guaranteeing safe and reliable performance of the software systems that the economy or society relies on [123]. From the perspective of the *Guideliner*, software verification process proves that the *Guideliner* is capable of evaluating different types of web applications (e.g. entertainment portals, public sector portals etc.) effectively regardless of the WUI technology. Despite the fact that the *Guideliner* supports HTML/CSS/JavaScript based WUI, the frameworks used for WUI development and the methods used for composing and formalising the structure of WUI HTML/JavaScript/CSS code can potentially affect the accuracy of the evaluation process. In general, there are two types of software verification: dynamic (experimentation) and static verification. The latter covers static verification methods such as following code conventions, patterns and software metrics. Dynamic code verification, in its turn, is performed during software runtime verifying that the software behaves in an expected way.

Current Section concentrates on dynamic verification of the *Guideliner* tool with the purpose to find deviations in its behaviour when the conformance to usability guidelines

for different WUIs is evaluated. Different types of deviations can be detected such as slow response time, errors occurred during evaluation or missing fields in response (e.g. missing status of evaluation, missing error description or missing name of element that violates the guideline). The dynamic verification of the *Guideliner* was carried out to answer the following questions:

- How efficient in terms of guidelines being evaluated, and time needed for evaluation is the *Guideliner* for automatic evaluation of WUI conformance to usability guidelines in comparison to other similar tools?
- How frequently the usability guidelines captured by the usability ontology for the *Guideliner* are violated in comparison with other similar tools?
- How homogeneous are the *Guideliner* evaluation results in terms of detected usability issues in comparison to other similar tools?

For *Guideliner* verification, ten top world universities, and ten top European universities based on the QS World University ranking[1], as well as six Estonian public universities were selected for the experiment. The QS World University ranking evaluates the popularity of universities according to various metrics including their academic reputation, employer reputation, faculty ratio and citation per faculty. University web pages were selected for the experiment as they generally serve a wide range of users including people from different countries and also people with disabilities; university web pages are used by students, academic personnel, researchers, and general public. In addition, European public universities are affected by EU legislation acts that encourage EU web applications to follow WCAG guidelines (in the USA, in its turn, Section 508 is recommended to be followed). Also, university web pages are quite popular by visit rate − for example the amount of all visits (user sessions) to Harvard University web page is in average 30 million[2] per month. In total 21 universities were filtered out for the experiment as some European universities were also within the list of top world universities.

The selected university web pages were tested on three different levels: start page, a subpage, and a page with a form that has at least five input fields. In general, users navigate to a web page from the start page that is why the usability of start page has a critical impact on the impression users get and was one of the page types included in the experiment. A subpage was selected randomly – the only criterion was that this page should not contain an error message. In general, subpages contain the same header, footer and navigation layout (that is why subpages have similar problems as start pages) but they also contain information (e.g. longer text articles of information) that people are searching for. For that reason, subpages need to follow also different usability guidelines (e.g. content-centric guidelines) that may not be the case for the start page. In case of a page with a form, conformance to form-specific usability guidelines must be evaluated. The page with a form can have a simpler structure than the start page but forms contain input fields that should be properly structured to guarantee users are able to fill in the forms correctly. Overall, the motivation of the page selection scheme was to include different types of pages for the evaluation set as to cover various usability guidelines. This schema of page selection enables to validate the behaviour of evaluation tools with different types of pages and to compare the number of usability problems detected by tools in different use cases.

[1] https://www.topuniversities.com/qs-world-university-rankings/methodology
[2] https://www.similarweb.com/website/harvard.edu

The complexity of web applications has not been considered before the experiment. For example, a web page containing long text and small number of elements can potentially violate fewer usability guidelines because of design simplicity. The test data selection is enough for the analysis as the purpose of evaluation is to compare the *Guideliner* with other tools (not to compare university WUI with each other) from the perspective of performance, detected violations, homogeneity of results etc. Overall, 63 web pages were selected for the evaluation (the complete list of these pages is presented in Appendix 6).

In addition to the *Guideliner*, six publicly available tools for automated usability evaluation were selected for the verification experiment: two freeware evaluation tools (Wave and AChecker) and four commercial tools (Powermapper, Tenon, DynoMapper and TotalValidator). The selection of tools was based on the next criteria:

1. The tool must support accessibility guidelines;
2. The tool must contain at least 50 guidelines;
3. The tool must be downloaded or used online without additional build steps;
4. The selection of tools must contain both freeware as well as commercial tools;
5. The tool must have English user interface.

Support for the accessibility guidelines is crucial as they improve the quality of WUI for user including users with disabilities. Accessibility standards contain around 70 accessibility guidelines, and around 50 of them are considered as more critical (e.g. WCAG Level AAA guidelines are not considered critical to follow) – for that reason criterion of 50 guidelines was used. Another important aspect of criteria is that the selected tool should be available online without additional build steps as additional build steps could potentially decrease the quality of build, and as a result the quality of the experiment. Both freeware and commercial tools were selected for the experiment to diversify the selection and to take into the comparison of tools from different areas. The English user interface is an inevitable requirement as the purpose was to compare the tools that can be used world widely rather than the tools made for a specific country. The *Guideliner* is a freeware tool that meets all aforementioned requirements.

In Section 4.1 two different lists of tools for automated usability evaluation were presented: W3C and the list presented by Timbi-Sisalima et al. in. Despite the fact that there are many different tools for usability evaluation presented by the aforementioned lists, most of them did not satisfy the selection criteria, missing either one or many features required. All tools listed in Section 4.1 support accessibility guidelines, though the number of supported guidelines did not satisfy the evaluation criteria. Some tools were not accessible online; some tools were quite unstable throwing multiple exceptions. Also, few tools were not internationalized for the English user interface. For instance, Examinator[1] presented in [115] did not have English version of UI; Nu Checker[2] did not support enough accessibility guidelines. Also, commercial tools were included in the current evaluation as some of them provided trial version for the short period of time that was enough to conduct the evaluation. As a result, six tools out of 116 were selected for the experiment: AChecker, Wave, TotalValidator, PowerMapper, DynoMapper and Tenon. The selection was partly based on the Timbi-Sisalima et al. [115] who filtered out most efficient tools from the W3C list that are feasible for automated usability tests. Also, the author decided to include Wave that evaluates contrast rate of elements. AChecker,

[1] www.examinator.ws

[2] https://validator.w3.org/nu/

TotalValidator, Tenon PowerMapper and DynoMapper were selected randomly from the list as most of selected tools provided the same functionality according to their documentation.

Wave and AChecker were selected as being widely used freeware tools for automated usability evaluation. For instance, Wave was used for the evaluation of Estonian public sector web applications in 2013 and 2015 [28] carried out by Ministry of Economic Affairs and Communication. AChecker has been used for evaluating WUI usability of India university web pages [124]. TotalValidator was selected because of rich functionality and clear evaluation reports. PowerMapper is a commercial tool that evaluates the widest variety of usability aspects including Search Engine Optimization (SEO) and compatibility with different browsers. This tool is used for example by Bank of America, eBay and Microsoft. Tenon is the only selected commercial tool that concentrates mostly on accessibility guidelines (used by Kaplan and The Paciello Group). DynaMapper differs from the other selected usability evaluation tools by providing the possibility to create web cards and to evaluate the whole web application and not just a single page. The chosen set of the tools covers various aspects of WUI usability evaluation (such as accessibility, usability, SEO optimization) and allows to compare the *Guideliner* to these tools based on the amount and types of guidelines, format of evaluation results and interfacing options. The analysis of these selected tools is not the aim of this study; the tools will be used as a benchmark toolset for the *Guideliner* verification.

The comparison of the guidelines supported by the tools used for the Guideliner tool verification is presented in Table 3. All tools support HTML-specific accessibility guidelines (e.g. described by WCAG and Section 508). There are only a few tools that support visual usability guidelines (e.g. Wave supports only visual guidelines for checking the contrast ratio). From the perspective of the number of the guidelines supported, Guideliner supports 98 guidelines that is less than PowerMapper (150 guidelines), AChecker (152 guidelines) and TotalValidator (115 guidelines) and more than Wave (80 guidelines), Tenon (74 guidelines) and DynoMapper (89 guidelines). Tools contain more guidelines than WCAG (contains around 70 guidelines) as many guidelines in WCAG are mapped into multiple automated guidelines. For example, WCAG guideline[1] "H44: Using label elements to associate text labels with form controls" is divided into 18 automated guidelines in case of AChecker as every input element (e.g. text, checkbox, password or file inputs) forms its own guideline.

Appendix 7 provides a detailed comparison of characteristics of the selected benchmark tools. Most of the tools do not support adding custom usability guidelines; only AChecker and Guideliner are extendable with new guidelines. The Guideliner is the only tool that allows evaluating the conformance to mobile usability guidelines. Other tools are limited only to desktop guidelines. The potential reason can be that screen reader is mostly used on desktop devices rather than on mobile devices that is why mobile platforms are not in the scope of the tools.

The author compared the tools based on the metrics presented by Timbi-Sisalima et al. in [115] in order to show that all tools satisfy the criteria of selection and that all of them have similar functionality in terms of the number and types of supported guidelines. The results of comparison showed that Tenon and DynoMapper support similar set of metrics. Wave, TotalValidator and AChecker support the same metrics but both of them are not commercial products giving additional point in the presented

[1] https://www.w3.org/TR/WCAG20-TECHS/H44.html

matrix. The same number of points also gets PoweMapper (despite that it is commercial product) supporting usability guidelines. The *Guideliner,* in addition, supports visual usability guidelines and allows defining custom usability guidelines. Appendix 8 shows the matrix of association of the tools with the metrics they support.

Usability tests with the selected benchmarking tools and the *Guideliner* were conducted during the period from 12.12.2017 to 03.05.2018 using all seven aforementioned tools for automated usability evaluation. A total of 441 usability tests were conducted. All tests were executed manually providing the URL of the web page for the evaluation as a parameter. The tools provided evaluation results in different format such as CSV, PDF, pictograms or as a list. For further analysis and to unify the evaluation results they were aggregated to Microsoft Excel.

Table 3. Comparison of the guidelines supported by the tools used for the Guideliner verification.

Tool	Supported Types of Guidelines	Number of Supported Types of Guidelines	Number of visual usability guidelines supported	Number of guidelines covered by tool
Wave	ARIA HTML Section 508 WCAG 2.0	4	3	80
Powermapper	HTML/HTML5 Section 508 Usability.gov WCAG 2.0 SEO	5	5	150
AChecker	BITV 1.0 Section 508 Stanca Act WCAG	4	n/a	152
Tenon	BITV 1.0 Section 508 Stanca Act WCAG	4	n/a	74
DynoMapper	Section 508 WCAG 2.0	2	n/a	89
TotalValidator	HTML Section 508 WCAG 2.0	3	n/a	115
Guideliner	Usability.gov Visual usability WCAG 2.0	3	43	98

Overall, less than 4% of tests (17 tests out of 441) of the experiment finished with an exception during the experiment (refer to Appendix 9 Total sum of detected violations of tool for automated usability evaluation column failed tests). The primary reason for these failures was preconfigured Robot Exclusion Standard (robots.txt) in the root directory of the evaluated websites that some tools followed for the evaluation (e.g. Powermapper); the causes of other failures were not identifiable.

Now, turning to the verification experiment goals, the first question to be answered during the verification experiment was the evaluation efficiency in terms of the guidelines processing speed. The evaluation efficiency was measured by two metrics: the number of evaluated guidelines per second (*gps*), and the number of guideline violations captured per second (*vps*). The metrics *gps* and *vps* were introduced to enable the comparison of the results of different tools on an equal basis. The metric *gps* shows how many guidelines the tool evaluates in the unit of time. The value of the *gps* is that it directly measures the tool performance as the more guidelines the tool evaluates during the unit of time, the better performance the tool has. The metric *vps* was used to measure the number of violations tools detects in the unit of time. To collect the data for both metrics, the number of distinct guideline violations, number of total violations and evaluation time were recorded during every experiment conducted to evaluate a specific page from the page set with a selected tool. The metric guidelines per second of each tool *gps(t) (t* stands for the tool*)* was calculated based on Equation 1 where *N(t,g)* (*t* stands for tool and *g* stands for guideline) is the total number of the guidelines that tool supports and *T(t)* (*t* stands for tool) is WUI evaluation time of the tool. Table 4 shows the results for *gps* for each tool and indicates how the *Guideliner* performs compared to the benchmark tools. For example, the average performance of the *Guideliner* compared to AChecker on Start Page is 17%, which means that the Guideliner is almost five times slower (4.6 times) than the AChecker to which it is compared to. The reason for that is the *Guideliner's* support for visual usability guidelines that are more resource intensive than HTML-based guidelines.

$$gps(t) = \frac{N(t,g)}{T(t)} \quad (1)$$

Table 4. Results of gps of the Guideliner compared to the benchmark tools.

Page type	**Average *gps* with Standard Deviation**			**Guideliner Performance %**		
Tool	Start page	Sub-page	Form	Start page	Sub-page	Form
AChecker	3.9 ± 6.3	5.2 ± 17.1	5.4 ± 7.8	17	17	22
DynoMapper	1.3 ± 3.3	1.1 ± 3.8	1.3 ± 4.4	53	84	89
Powermapper	6.0 ± 17.5	6.9 ± 18.3	4.7 ± 3.1	11	13	25
Tenon	6.9 ± 5.8	10.5 ± 18.7	12.9 ± 34.5	10	9	9
TotalValidator	6.4 ± 4.3	7.8 ± 5	9.1 ± 5.8	11	11	13
Wave	11.7 ± 21.5	13.2 ± 23.1	7.5 ± 3.2	6	7	16
Guideliner	0.7 ± 0.6	0.9 ± 0.8	1.2 ± 0.9	100	100	100
Average	6.0 ± 9.8	7.5 ± 14.3	6.8 ± 9.8	11	12	17

The results show that most of the tools evaluate in average 5 – 10 guidelines per second, with an exception of DynoMapper which is able to evaluate in average 1 – 2 guidelines per second. Also, the results show wide variation in standard deviations of average *gps.* The reason for the wide variation in standard deviations is a big difference in evaluation times. For example, the minimum evaluation time for PowerMapper was 4 seconds whereas the maximum was 88. Nevertheless, for tools like TotalValidator and the *Guideliner* standard deviation was smaller. There is not a clear answer why the

evaluation time was diverse between different pages for the same tool as the author does not have access to the source code of other tools.

The average *gps* for the *Guideliner* remained modest compared to other tools. The reason relies on its support to visual and mobile usability guidelines that require more resource-intensive operations (such as evaluation the distance, height, and width of elements) and take more time during each evaluation. This ability, unfortunately, slows down the evaluation speed. The *Guideliner gps* improves around 70% if checking of visual and mobile usability guidelines is omitted.

Let us now analyse the results of the metric *vps*. The metric violations per second *vps(t) (t* stands for the tool*)* was calculated based on Equation 2 where *N(t,v)* (*t* stands for tool and *v* stands for the number of violations detected on WUI) is the total number of detected violations and *T(t)* (*t* stands for tool) is WUI evaluation time of the tool. Average number of violations on the start page, sub-page and page with a form were documented. Table 5 shows the results of *vps* for each tool and indicates how the *Guideliner* performs compared to the benchmark tools. For example, the average *Guideliner* violation detection performance compared to DynoMapper on the start page is 425% which means that the *Guideliner* detects around four times (4.25) more violations per second than DynoMapper to which it is compared to. The results show that most of the tools detect in average 1 – 5 violations per second, with the exception of Wave and Tenon which are able to detect in average 5 – 10 violations per second. The average *vps* for the *Guideliner* was average between all tools showing higher performance than AChecker, DynoMapper and PowerMapper and lower performance then Tenon, TotalValidator and Wave.

$$vps(t) = \frac{N(t,v)}{T(t)} \tag{2}$$

Table 5. Results of vps of the Guideliner compared to the benchmark tools.

Page Type	**Average *vps* with Standard Deviation**			**Guideliner Performance %**		
Tool	Start page	Sub-page	Form	Start page	Sub-page	Form
AChecker	1.1 ± 0.9	1.1 ± 1.0	2.1 ± 1.8	265	270	162
DynoMapper	0.7 ± 0.6	0.6 ± 0.5	1.0 ± 1.5	425	530	331
Powermapper	2.4 ± 1.7	1.3 ± 0.8	1.5 ± 1.1	126	240	226
Tenon	5.1 ± 4.9	6.8 ± 7.7	6.5 ± 4.8	60	44	53
TotalValidator	3.2 ± 1.9	3.2 ± 3.3	12.9 ± 39.2	96	96	26
Wave	10.6 ± 6.4	8.8 ± 8.9	8.4 ± 5.5	29	34	41
Guideliner	3.0 ± 2.2	3.0 ± 2.2	3.4 ± 3.6	100	100	100
Average	3.9 ± 2.7	3.6 ± 3.7	5.4 ±9	78	81	63

The second question the experiment sought answer to was to evaluate the frequency of violation of the supported usability guidelines. The frequency of violation of the guidelines was measured by metric: frequency of violation (*fov*). The metric frequency of violation *fov(t)* (*t* – stands for the tool) was calculated based on Equation 3 where *N(t,d)* (*t* stands for tool and *d* stands for the distinct detected usability guidelines) shows the number of distinct violations of usability guidelines and *G(t)* (*t* stands for the tool) shows

the total number of supported guidelines. The metric was measured as the total amount of guidelines supported by the tool may be misleading because there can be the guidelines that are always passed by most WUI. The metric *fov* was introduced to enable the comparison of the results of different tools on an equal basis. Table 6 shows the results for *fov* analysis for each tool and indicates how the *Guideliner* performs compared to the benchmark tools. For example, the average *Guideliner* performance compared to AChecker on the start page is 539% which means that the relation of total number of distinct violations of usability guidelines to the total number of guidelines is 5.39 time higher for the *Guideliner* than AChecker to which it is compared to. The results show the average violation frequency between all the tools is around seven. The average *fov* for the *Guideliner* exceeds the average of all other tools around 2.6 times. Overall, the aforementioned results mean that *Guideliner* violation frequency is highest between all benchmark tools. Such results were achieved by including visual common and mobile usability guidelines that were violated more often than accessibility guidelines.

$$fov(t) = \frac{N(t,d)}{G(t)} \tag{3}$$

Table 6. Results of fov of the Guideliner compared to the benchmark tools.

Page Type	**Average *fov* with Standard Deviation**			**Guideliner Performance %**		
Tool	Start page	Sub-page	Form	Start page	Sub-page	Form
AChecker	2.9 ± 2.4	2.7 ± 2.3	3.6 ± 2.9	539	605	626
DynoMapper	9.8 ± 6	10.2 ± 9.2	12.2 ± 8.9	158	176	164
Powermapper	8.8 ± 7.3	5.5 ± 4.7	6.8 ± 6.2	283	195	302
Tenon	5.0 ± 3.1	4.8 ± 2.9	5.3 ± 3.2	364	344	345
TotalValidator	4.5 ± 3.2	3.7 ± 3.4	10.0 ± 30.1	192	379	451
Wave	10.4 ± 6.5	9.4 ± 13.6	7.5 ± 4.4	257	167	178
Guideliner	17.3 ± 3.9	16.7 ± 3.8	19.2 ± 6.0	100	100	100
Average	6.9 ± 4.8	6.0 ± 6.0	7.6 ± 9.3	255	250	276

The third purpose of the experiment was to compare how homogeneous are the *Guideliner* evaluation results in comparison to the selected benchmark tools in terms of the detected usability issues. Appendix 9 shows the raw data of number of violations grouped by university WUI and the tool. As this data is not comparable due to different tools having a different set of guidelines they are based on, the author normalized the data by introducing the concept of violation coverage which indicates the correspondence of distinct guideline violations to the total amount of guidelines supported by the tool. Firstly, the violation coverage of the tool for each university WUI *VC(u,t)* (*u* stands for university WUI and *t* stands for the tool) was calculated based on the Equation 4 where *N(u,t)* is the number of distinct violation detected by the tool for each university WUI and *N(g)* (*g* stands for the guideline) is the number of guidelines the tool supports. Secondly, the minimum $VC_{min}(t)$, maximum $VC_{max}(t)$ and average $VC_{avg}(t)$ violation coverages per tool were calculated finding the minimum, maximum, average value within the range of *VC(u, t)* of each tool (refer to Equation 5-7).

$$VC(u,t) = \frac{N(u,t)}{N(g)}; \quad (4)$$

$$VC_{min}(t) = min([VC(u,t)]); \quad (5)$$

$$VC_{max}(t) = max([VC(u,t)]); \quad (6)$$

$$VC_{avg}(t) = average([VC(u,t)]); \quad (7)$$

Table 7 shows the results of the analysis of minimum, maximum and average violation coverages per tool and indicates how the *Guideliner* performs compared to the benchmark tools. The results in Table 7 show that the *Guideliner* has highest violation coverage within all the tools. In addition, the *Guideliner* minimum and maximum violation coverages are higher than the same metric for other tools. Appendix 9 Table Minimum, Maximum and Average Violation Coverages complements the data presented in Table 7 by adding additional granulation by university WUI enabling to compare the *Guideliner* performance in the scope of each WUI in the experiment. The results clearly outline that the *Guideliner* minimum coverage is higher for all WUI than the minimum over all tools, while the maximum performance is more than zero in most cases meaning that the *Guideliner* showed the highest maximum coverage between all tools. The average *Guideliner* coverage results are also not homogeneous with average results of all tools exceeding the average coverage results between all tools multiple times. Overall, the maximum, minimum and average results of the *Guideliner* violation coverage show mostly in all cases better performance than the result of the same metrics of other tools. Aforementioned results mean that the *Guideliner* having fewer guidelines described than some other tools (e.g. PowerMapper, AChecker) is still able to detect more violations. The reason for such performance is that the *Guideliner* contains usability guidelines that are not supported by other tools but at the same time that are violated as often as accessibility guidelines.

Table 7. Results of vc of the Guideliner compared to the benchmark tools.

	Violation coverage			**Guideliner vs Benchmark tool**		
	Min	**Max**	**Average**	**Guideliner min %**	**Guideliner max %**	**Guideliner average %**
Wave	0.02	0.28	0.09 ±0.06	522	88	194
PowerMapper	0.02	0.18	0.07 ±0.04	544	143	252
Achecker	0.01	0.11	0.03 ±0.02	1240	229	582
DynoMapper	0.10	0.20	0.09 ±0.05	108	124	198
Tenon	0.01	0.09	0.05 ±0.03	805	266	350
TotalValidator	0.01	0.29	0.09 ±0.05	1251	86	342
Guideliner	0.11	0.25	0.18 ±0.03	100	100	100

Overall, the verification evaluation of the *Guideliner* tool proved that the *Guideliner* is functionality-wide mature tool that can be used for automated usability evaluation as an alternative to the existing software. The comparison of characteristics of the tools showed that the *Guideliner* is the only tool that in addition to HTML-specific guidelines evaluates also visual usability guidelines. On the one hand, from the perspective of

evaluation speed, the *Guideliner* falls below the average level because of its supports for visual and mobile usability guidelines that require more resource intensive operations. On the other hand, verification evaluation demonstrated that support for visual usability guidelines allow the *Guideliner* to detect more violations on WUI than other tools. Thus, the verification evaluation proved that the *Guideliner* is comparable to the alternative tools for automated usability evaluation showing higher than average results for violation coverage and frequency of violation. The experiments presented in current Section were obtained in the process of tight cooperation and supervision of a master student the author was supervising [125].

4.8 Limitations of the Guideliner

Multiple research works advice to combine various usability evaluation approaches pointing out that there is no "silver bullet" solution capable of evaluating every aspect of WUI [46] [67]. For that reason, usability evaluation should not rely only on one method. To design a highly usable WUI the combination of multiple evaluation methods (such as user testing, card sorting, heuristic evaluation etc.) should be used. The proposed *Guideliner* tool is capable of evaluating the conformance of WUI to a set of predefined usability guidelines. Nevertheless, it cannot entirely substitute manual evaluations and user tests as it is practically impossible to simulate human-specific characteristics as human sense as stated in [49]. In comparison with other similar tools (refer to Section 4.7 for more details), the speed of evaluating usability guidelines is considered as the main *Guideliner* limitation (as it showed poorer results than all other tested tools).

The *Guideliner* supports HTML 4 and 5, CSS 3 (providing backwards compatibility with previous versions) and JavaScript-based WUIs. It does not require additional adoptions allowing evaluation of any WUI without extra configurations. The only technological limitation is that WUIs requiring specific environments to run (e.g. Flash and Java Applets) are not supported by the tool. The reason for this is that the *Guideliner* uses Selenium WebDriver to process WUI (Figure 16). Selenium WebDriver supports only HTML, JavaScript and CSS based WUIs. To run a Flash web application, the Adobe Flash Player[1] is required containing the embedded environment for injecting Flash Web applications into the browser. Java Applets, on the other hand, require Java Runtime Environment (JRE) to run an applet or web application. In fact, both technologies are deprecating in favour of HTML 5 due to their age and security issues, and both require additional plugins in browsers to be installed.

The *Guideliner* is in an active development (available on Github[2]), and to make it widely used, the source code should be refactored and additional predefined usability guidelines from e-commerce, banking, search engine optimization (SEO) domains added to *Ontology Repository,* making the *Guideliner* more valuable for e-commerce. All aforementioned improvements are considered as topics for further research.

[1] https://get.adobe.com/flashplayer/
[2] https://github.com/guideliner

4.9 Chapter Summary

Automated usability evaluation is an emerging approach to optimise labour-intensive and time-consuming process of manual usability evaluation. It is achieved through the use of tools capable of automated evaluation of WUI against the conformance to usability guidelines.

In this chapter, the *Guideliner* – a tool for automated evaluation of WUI conformance to usability guidelines was proposed. The *Guideliner* tool uses usability ontology (introduced in Section 3.4) as a source of usability guidelines. Ontology processing engine is used to transform usability guidelines described in ontology into corresponding Java objects. The Selenium-based WUI evaluation component checks the conformance of WUI to the usability guidelines and returns the report as a result of evaluation. The solution proposed in this Chapter has been published in [12].

The main contribution of this Chapter is a method for usability evaluation prevailing over analogue solutions such as Mauve, AChecker etc. The novelty of proposed method is that it is overcoming the limitations of existing solutions (that are limited only to the evaluation of HTML-specific usability guidelines) evaluating both HTML centric accessibility guidelines as well as visual usability guidelines that are evaluated on finally rendered WUI designed especially for mobile platform as well as guidelines specific for desktop devices only. The proposed solution is capable of evaluating most HTML, CSS and JavaScript-based web applications; the only limitation is that WUIs requiring specific environments to run (e.g. Flash and Java Applets) are not supported by the method. The *Guideliner* contains 20 mobile usability guidelines, 23 common usability guidelines (applicable both to mobile and desktop devices) and 55 accessibility guidelines. Altogether the Guideliner covers 98 usability guidelines; support for more guidelines can be obtained by describing more guidelines in the usability ontology.

In order to evaluate the *Guideliner* and compare its performance to other usability evaluation tools, a total of 441 usability tests were conducted. The experiment results showed that the *Guideliner* detected more usability issues than other tools. Also, the comparison of tools' characteristics showed that the *Guideliner* is the only tool that in addition to HTML-specific guidelines evaluates also visual usability guidelines. Because of the support for resource-intensive visual usability guidelines the *Guideliner* falls below the average from the perspective of evaluation speed. Overall, the comparison of the *Guideliner* tool to the benchmark tools proved that the *Guideliner* provides sufficient functionality for automated usability evaluation showing better results for some of the verification criteria.

5 Usability Evaluation within WUI Development

Usability evaluation is a method for identifying specific problems with usability of products [9]. As a process, it is laborious and costly in terms of human resources and time. If some of this work could be done automatically in early development phases of WUI using knowledge and best practices from web usability domain, then usability problems could be identified as they occur during the development process and solved immediately – saving time and cost on the necessity to re-develop an ill-designed WUI with poor usability. Automatic usability evaluation during WUI development would economize time and cost of using human experts and testers wherever possible, and contribute to and stimulate production of high-quality user interfaces.

This chapter addresses the problem of how to effectively evaluate web user interfaces usability and cross-platform compatibility during their development, in particular within software implementation phase with minimal cost and time. Two case studies will be presented.

5.1 Introduction

Web user interfaces are very fastidious to the modifications as even a small change in some style or colour scheme, alteration of layout (e.g. decreasing the position between the content and corresponding title), or addition of elements can cause new usability issues (for instance, changing the colour of a link text, making it lighter or darker, can potentially lead to severe usability problems regarding contrast). According to WCAG guidelines documented in [32], the contrast ratio between the letters and the background that is immediately behind the letter should be kept above 4.5. Violating this guideline leads to lower usability for target users including people with (eye-) disabilities. Each modification during development can potentially introduce new issues of usability, and in the worst case severely lower the usability of a WUI. Discovering poor usability as early as possible – during development – makes fixes less costly than finding and fixing them in later phases of development (e.g. after launch/going live). To bring usability evaluation into WUI development requires an approach that will provide subsequent feedback to WUI developers. The latter is one of the aims of the *Guideliner* tool presented in this thesis.

A typical WUI development process consists of the next six iterative phases: planning, analysis, design, implementation, testing and maintenance [17]. Planning, analysis and design phases are responsible for gathering the right requirements, prioritizing them and formalising for the WUI development. Certain inspection evaluation methods such as heuristic evaluation and cognitive walkthrough can be applied already during the design phase when the initial prototypes and mock-ups are evaluated by usability experts. WUI code writing occurs during the implementation phase when developers establish or modify WUI source code according to the requirements set in the planning phase. Afterwards, during the testing phase, it is ensured that the result of the development is exactly the same as it was planned, and that the WUI after modifications is usable and user-friendly. In general, inspection and empirical usability evaluation occur during the testing phase when WUI is finalised for the release. Nevertheless, fixing usability problems during the testing phase is more time-consuming than correcting the same issues during the implementation phase as research has proven [126] [127].

Multiple research works have concentrated on investigating the cost of changes and the cost of fixing the bugs in software during different phases of software development

[126] [127] [128]. Boehm [126] published statistics about the cost of software changes on different software phases based on the TRW Automotive and IBM projects. The results showed that the cost of change or bug fixing increases with every consecutive phase of software development. To fix an error or defect in requirements is relatively cheap in comparison to fixing the same error during acceptance testing or during software maintenance phase. Of course, in case of larger software projects, the difference in cost is more considerable than for smaller software projects. Similar conclusions were also presented by McConnel [127], who claims that finding the defects during the initial phases of software development decrease the cost of project around 10 – 100 times in comparison with the situation when they are detected and fixed during the acceptance testing. Tassey [128] investigated the economic impact of an inadequate infrastructure of software testing in the U.S. The results of the study showed that most of the bugs (40% of all bugs) are detected during testing and fixing the defects found during the testing is around two times more expensive (Figure 25: p – 8p show relative average cost per bug) than fixing defects found during the implementation phase of development. These findings serve as a motivation for concentrating on automated usability evaluation particularly during the coding or implementation phase of WUI development that decreases the average cost per bug in average two times in comparison with the testing phase.

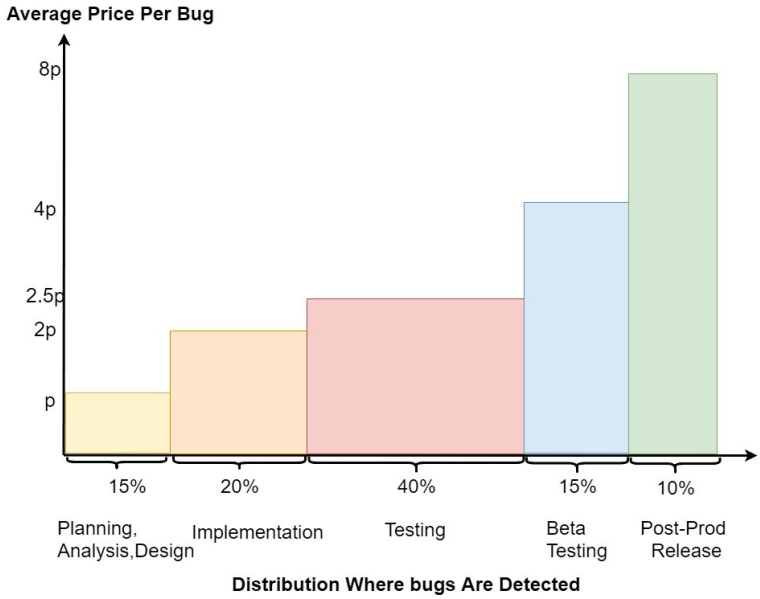


Figure 25. Software testing costs shown by where bugs are detected [129].

Commonly, WUI usability is not tested with each minor change separately but rather checked when all functionality for a release is finalised. It introduces a gap between the moment usability defect is detected and the moment when it was introduced by developers into the product. The motivation of the current Chapter is to address the latter gap and propose a method of evaluating WUI conformance to usability guidelines during the implementation phase of WUI development process. The real benefit of automated usability evaluation during the implementation phase is increased quality of WUI achieved by immediate feedback to developers on found usability issues. Usability

evaluation during the implementation phase allows to identify usability defects and issues as they occur allowing to save time and cost on necessity to re-develop the WUI. Also, this would save time and cost of using human experts and testers wherever possible, stimulating development of high-quality user interfaces. Automated detection of usability issues during the implementation phase enables to focus during the testing phase on the issues that cannot be automatically detected.

While Chapter 4 focused on presenting the *Guideliner* – a tool for automated usability evaluation using the *Guideliner User Interface* component for initiating the process of automated evaluation, this Chapter concentrates on exploiting the *Guideliner Core* within a development IDE for automated usability evaluation during the implementation phase of WUI development.

The rest of this chapter is organized as follows. Firstly, usability evaluation during different phases is discussed. Then, the *Guideliner* integration into implementation phase of WUI development and Continuous Integration Process are presented. Finally, use cases of the *Guideliner* integration into RIA (Republic of Estonia Information System Authority) WUI development process and to the Kühne + Nagel IT Service Centre AS intranet web application development process are demonstrated.

5.2 Usability Evaluation during WUI Development Process

A typical software development process today is an iterative process (e.g. Scrum[1]) consisting of planning, analysis, design, implementation, testing and maintenance [17]. User Interface development also follows the aforementioned phases (Figure 26) containing specific activities for (W)UI development during each phase:

1. Planning, Analysis, Design phases – determining the goals of the iteration or the whole project and allocating the resources needed to achieve the solution. In general, the output of these phases is a set of stories to be implemented. The initial WUI prototype is usually be created and evaluated during these phases.
2. Implementation – part of software development process when the software, including (W)UI, is being coded according to the specification defined; it can also include writing automatic developer tests.
3. Testing – ensures that changes introduced during the implementation phase of (W)UI development meet their objectives, and that newly developed features do not decrease usability of WUI. Functional WUI testing is not in scope of current thesis as current thesis is concentrating only on usability evaluation during testing phase of WUI development.
4. Maintenance – contains operations for maintaining the application in a production environment including post-implementation reviews.

The duration and complexity of each phase can be different depending on the selected development methodology. In case of an agile methodology, there are shorter and flexible iterations during the development cycle than in case of a waterfall model when the development process flows in one direction long and requirements cannot evolve dramatically during WUI development.

[1] https://www.scrum.org/resources/what-is-scrum

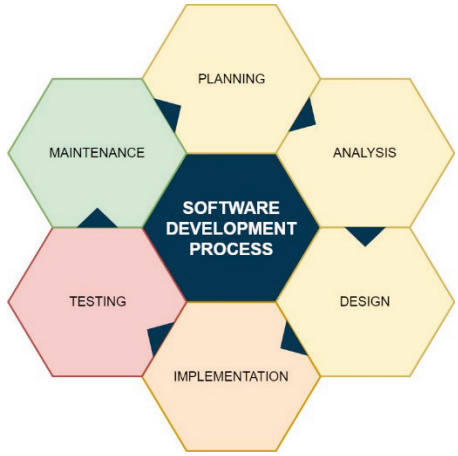


Figure 26. Phases of software development process.

Most existing methods for usability evaluation fall into design and testing phases. For example, heuristic evaluation, cognitive walkthrough and user tests can be initially conducted on mock ups or even paper prototypes to receive initial feedback on the preliminary design. Afterwards, when WUI has been developed according to the mock ups, in the testing phase WUI is evaluated using various methods including heuristic evaluation, user tests, eye tracking, expert review etc. The only usability evaluation occurring during the implementation phase is WUI developer review when the developer evaluates usability (of developed functionality) based on his knowledge in the domain of usability evaluation [122].

In general, during the implementation phase of WUI development, developers write code that satisfies the requirements; they add new elements to WUI or modify existing elements on the screen according to business requirements. In addition to modifying the code of WUI, developers cover the functionality with automated unit and functional WUI tests. WUI functional tests (black-box testing which checks the software behaves as defined in specification) facilitate the evaluation of application quality automatically providing the feedback about the status of the application.

In general, WUI developers do not have enough expertise and competency to evaluate usability of WUI as their main responsibility is functional development of WUI, e.g. dividing design into components, writing front-end code and covering code with tests. The general lack of competence in usability domain is the reason why usability evaluation should also be automated. The automation of usability evaluation should be done in a way so even developers without a good usability evaluation experience would understand the cause of detected problems and the way problems should be fixed as eventually, developers are the main consumers of automation of usability evaluation during the implementation phase. Descriptive usability evaluation result will help developers to avoid introducing the same usability defects again as they understand the reason of problem.

Typical WUI development paradigm requires usability testing to be conducted by usability experts and quality assurance (QA) specialists to be included in testing phase. (as shown in Figure 27). Usability testing can be conducted automatically using tools designed for automated usability evaluation or manually involving usability experts and potential users. The principal shortcoming of usability evaluation during testing phase is

that usability defects identified should be reported and sent back to developer to be fixed. This, however, introduces a delay between the moment when WUI was changed and the moment the developer receives feedback on errors, increasing total time of development and as a result also cost of the fix and the project.

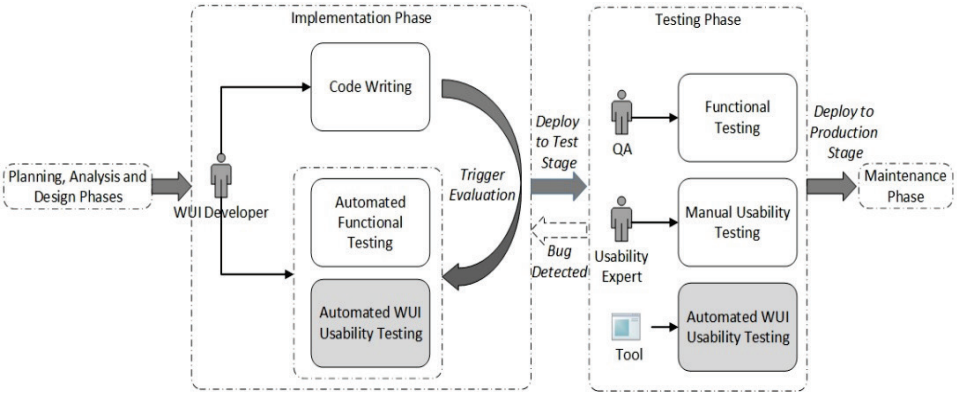


Figure 27. Phases of WUI development process highlighting the missing activity (Automated WUI Usability Testing) during the implementation phase.

The current Chapter addresses the aforementioned delay in the feedback on usability evaluation, proposing a solution that enables automated usability evaluation already during the implementation phase of WUI development. Figure 27 highlights the missing activity *Automated WUI Usability Testing* during the implementation phase of WUI development process. An approach to evaluate usability during the implementation phase will be introduced in Section 5.3 with more details.

As stated earlier, there is a gap between the moment usability defect is detected and the moment it was introduced into product. The reason why automatic usability evaluation is not part of the implementation phase is the lack of proper tools for automated usability evaluation during this implementation phase. There are existing solutions that can be used during testing but there are obstacles (e.g. these tools require special environment to run and they cannot be executed together with unit or functional tests) preventing them to be used during implementation phase of WUI development. Implementation phase of WUI development is code-centric while testing phase of WUI is concentrating on WUI testing using various methods and tools that increase the efficiency and accuracy of evaluation such as user tests and tools for automatic usability evaluation. Existing tools for automated usability evaluation fall into the category of tools suitable for testing phase being development process-centric rather than code-centric.

Overall, usability is evaluated rather during the testing phase than during the implementation phase of WUI development process. The following sections are concentrating on problems of implementation-time usability evaluation.

5.3 Guideliner Integration into WUI Implementation Phase

The benefit of conducting usability evaluation during the implementation phase of WUI development is that detected defects of usability can be fixed immediately – saving time and cost on the necessity to re-develop a WUI with poor usability. The purpose of current Section is to enhance the *Guideliner* with the possibility of immediate usability evaluation

during implementation phase of WUI development. A part of this enhancement is achieved by enabling integration of the *Guideliner* tool with integrated development environments (IDE).

The main beneficiaries of implementation-time automatic usability evaluation are WUI developers as the feedback provided by evaluation results informs the developers on how the changes they introduced comply with usability guidelines. Existing solutions for implementation-time code evaluation (e.g. WUI unit tests (Karma[1], Jasmin[2]) or functional tests (PhantomJS, Selenium)) are architecturally designed as environment-agnostic pluggable components that can be used as a part of IDE.

Originally, the *Guideliner* was designed as a separate web application (Section 4.3) addressing testing phase of WUI development. In order to bring it into the use already in the implementation phase, the *Guideliner* needs to be integrated with an IDE and thereby should be distributed as a pluggable component. As an outcome of further development of the *Guideliner* tool, the primary deliverable for WUI developers community is an artifact distributed as a dependency for dependency management tool (e.g. Maven[3], Gradle[4] or Ivy[5]), which provides an API for executing usability tests on the local or remote machine; and contains *Guideliner Core* compiled and packaged as a Java Archive (JAR) required by dependency management tools. The advantage of distributing the *Guideliner* via the dependency management tool is that it can be easily plugged into an existing project.

Introducing the *Guideliner* into the implementation phase requires some prior configuration of the *Guideliner* by WUI developers so that usability tests are executed together with other WUI tests. Figure 28 shows high level architecture of the *Guideliner* (described in Section 4.3) complemented with *Testing Module* that provides Java API which makes it possible to use the *Guideliner* also within an IDE. The *Testing Module* contains *Integration Testing Component* that provides an abstraction for creating and executing automated usability tests. *Testing Module* and *Guideliner Core* form together Java Archive (JAR) that is integrated into IDE. Overall, the described architecture of the *Guideliner* enables triggering evaluation process from IDE by the mean of JAVA API.

The purpose of the automated usability tests enabled by *Guideliner* as well as automated WUI unit and functional tests is to check certain characteristics of WUI. Test asserts the actual values of evaluated characteristics (in the context of usability evaluation, characteristics are the properties of WUI elements such as the colour, size, font of the elements) with the expected ones. There are a few frameworks available for asserting the actual results with expected ones. The most popular testing frameworks are JUnit[6] and TestNG[7]. JUnit has been used as a framework for composing automated usability evaluation tests for the research described in this Chapter because of the author's proficiency in that framework. In fact, JUnit and TestNG serve the same purpose and have similar syntax. JUnit can be easily replaced with any other alternative as the developed library is test framework agnostic.

[1] https://karma-runner.github.io/2.0/index.html
[2] https://jasmine.github.io
[3]https://maven.apache.org
[4] https://gradle.org
[5] http://ant.apache.org/ivy
[6] http://junit.org/junit4
[7] http://testng.org/doc

JUnit tests can be run from the command line as well as from an IDE. There are many IDEs available either freeware like Eclipse[1] and NetBeans[2], and commercial software such as IntelliJ IDEA[3] (it has freeware version but with limited functionality). Every aforementioned IDE provides sufficient functionality for showing clear and concise visual output of evaluation results of automated tests. The IntelliJ IDEA has been used for the research described in this Chapter because of the author's experience with it, nevertheless, it can be substituted by any alternative IDE with no effect on productivity.

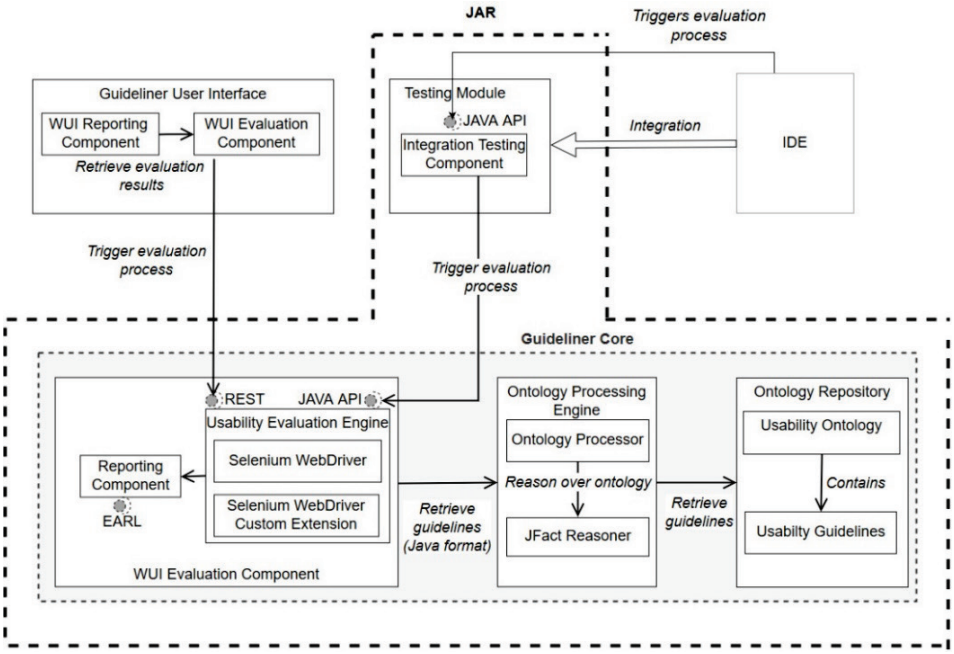


Figure 28. High level architecture of the Guideliner complemented with Testing Component.

The evaluation of WUI can be triggered directly from IDE visual user interface or command line (e.g. using Maven or Gradle commands). There are two options available how to configure the *Guildeliner* usability tests: reuse a predefined set of usability guidelines or define a new set. Besides selecting the guidelines for evaluation, the *Guideliner* also allows configuring the tests with additional parameters: *webURL* (URL of web application to be evaluated), *browserType* (type of web browser such as Chrome) and *browserWindowSize*. The aforementioned parameters make it possible to perform the evaluation of WUI using different browsers with various browser windows size. When the parameters are set, the evaluation can be started. An example of IntelliJ IDEA evaluation results is presented in Figure 29 showing the full integration of the Guideliner tests into the IDE. The Test results shown in Figure 29 contain the testing method being executed – *testMobileUsabilityGuidelines* and in brackets corresponding usability guideline evaluated and the time evaluation took. The green circle in front of each usability evaluation result means that the test passed, the orange circle highlights failed

[1] https://www.eclipse.org
[2] https://netbeans.org
[3] https://www.jetbrains.com/idea

tests. The tests can be executed all together, or only a selection of tests carried out. The integration provides value especially for developers as they can run all the task (programming and testing) conveniently in the same environment, without the need to switch software. With this approach, WUI usability tests can be initiated easily with one click in IDE.

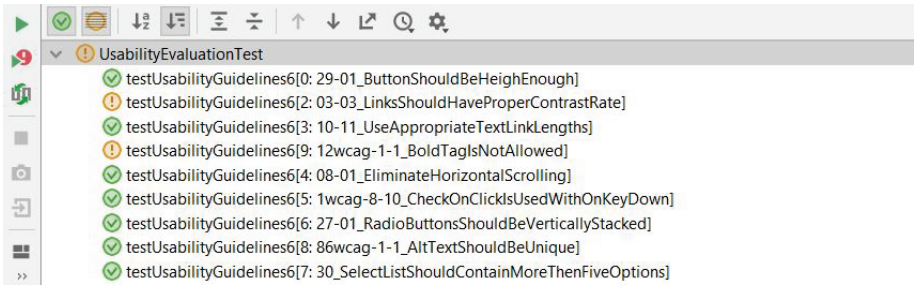


Figure 29. Automated Usability Evaluation Results for Estonian Research Information System https://www.etis.ee (extract from IntelliJ IDEA).

In order to obtain the descriptive reason for failure described in the evaluation results, the failed usability test should be selected and clicked. After selecting the failed usability test, a new window appears (Figure 30) containing all WUI elements that violate selected usability guideline. The view contains the next information: text of the element that violates the guidelines, type of the element, screenshot name and the violation reason containing the expected value of characteristic being evaluated and the actual value extracted from WUI. This detailed information helps developers to identify elements of WUI that violates usability guidelines, and to understand the cause of these test failures, allowing to introduce necessary fixes in the implementation phase already.

```
Element Text: Uudised
Element type: LINK
Element path: screen1542738048741.0.png
Violation reason Element with text Uudised does not have required contrast of 4.5

Element Text: Vanemad uudised
Element type: LINK
Element path: screen1542738048928.1.png
Violation reason Element with text Vanemad uudised   does not have required contr

Element Text: Priit Tuvike
Element type: LINK
Element path: screen1542738049085.2.png
Violation reason Element with text Priit Tuvike does not have required contrast o
```

Figure 30. An additional description provided for failed usability guideline for https://www.etis.ee (extract from IntelliJ IDEA).

Overall, to integrate the *Guideliner* into implementation phase of WUI development process, the *Guideliner* has been further developed to provide an API for executing usability tests on the local or remote machine either executed from command line or from IDE. Integration of automated usability evaluation into IDE allows initiating automated usability tests from the same environment where developers design other tests such as WUI functional tests. With this development, usability defects can be detected during the implementation phase of WUI development that makes the fixes less time and resource consuming.

5.4 Guideliner in Continuous Software Integration Process

Continuous Integration (CI) as defined by Fowler and Foemmel in [129] is "*a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible*". The importance of CI is that it reduces the cost and the risk of delivering software with defects supporting more frequent incremental build of an application. CI is established based on the development timeline [130]. In general, all activities of CI process like building an application and running automated tests (including usability evaluation tests) are configured and run automatically every time developer commits the code to code repository. Under this approach, every modification of WUI is automatically evaluated against a set of usability guidelines informing the developer in case some usability guidelines are violated. One of the purposes of current thesis is to automate usability evaluation that is why fitting the *Guideliner* into the CI process is beneficial as it checks the conformance of WUI to usability guidelines during the WUI deployment to the staging environment. While Section 5.3 introduced *Guideliner Testing Component* that is compatible with IDE, the purpose of current Section is to show how the *Guideliner Testing Component* can be reused during the CI process.

A sample development pipeline of CI is presented in Figure 31 consisting of *Version Control Server*, *Integration Server* responsible for building the application and running automated tests, and *Application Server* where the software is deployed. In order to execute *Guideliner*-based usability tests together with other automated tests, the project should be built first. The automated tests include WUI code unit tests, integration tests testing the behaviour of a component or the integration between a set of components, and functional WUI tests verifying that WUI behaves in a way it is expected. Using the *Guideliner* as a part of the CI pipeline complements the list of automated tests with automated usability tests that check the conformance of WUI to predefined usability guidelines.

The internal process of WUI evaluation workflow with the *Guideliner* as part of CI containing main evaluation activities is described on Figure 32. Initially, developers modify code of WUI and commit modified files to the remote code repository. Then, integration server component that is listening for modifications of the code captures the code modification event. Afterwards, integration server checks out code of application, builds the project and triggers the process of automatic evaluation checking whether WUI is accessible from the browser and whether there is a predefined set of guidelines to be processed. Then, the evaluation of WUI is performed according to the predefined usability guidelines. Finally, depending on the evaluation results, the evaluation status is set to 'passed', if WUI conforms to all usability guidelines, and the application is deployed to the stage environment. In all other cases, the evaluation status is set to 'failed' and the report providing failed usability tests and corresponding reasons for failures are provided to the developer running the tests.

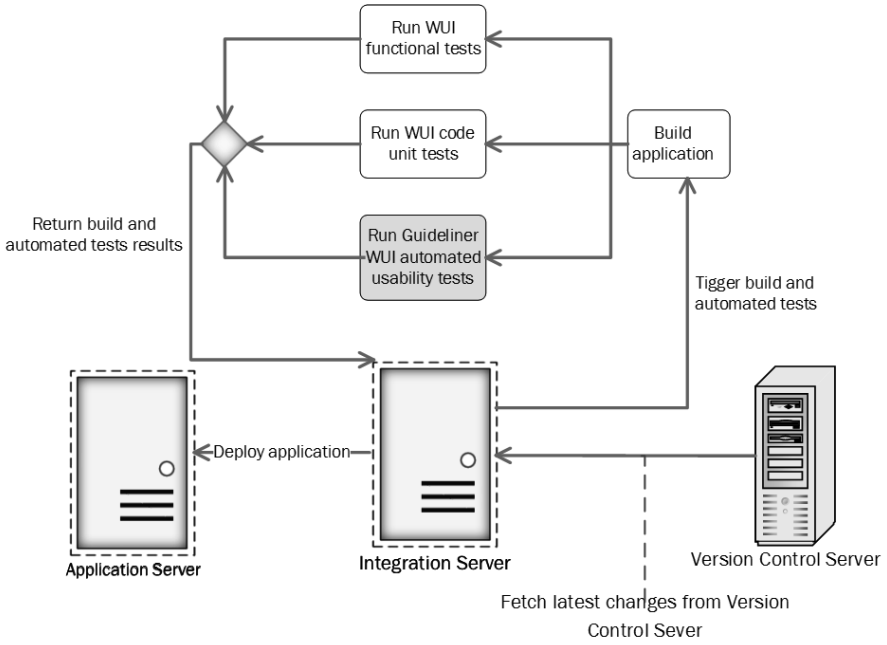


Figure 31. A sample development pipeline of continuous integration.

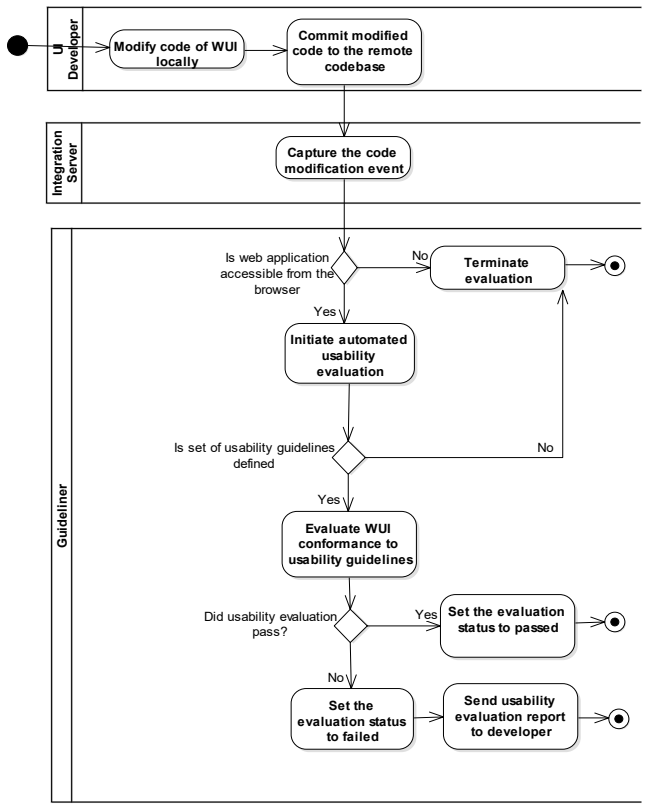


Figure 32. Activity diagram for WUI evaluation workflow.

To sum up, the *Guideliner* automated WUI usability tests can be integrated into CI process together with other types of automated tests such as WUI unit and functional tests. Applying the *Guideliner*-based usability testing during CI process verifies that every WUI modification deployed to the staging environment conforms to set usability guidelines.

5.5 Integration Use Cases

In order to validate the proposed method of implementation-time automatic usability evaluation of WUI, and to verify that it is suitable under real development conditions, the *Guideliner* was introduced to the development and CI processes at the Estonian Information Systems Authority (RIA) and Kühne + Nagel IT Service Centre AS (K+N), both interested in improving their development processes.

5.5.1 Integration into RIA WUI Development Process

Estonian Information Systems Authority (RIA) coordinates the development and administration of the Estonian State Portal (ESP)[1] – the gateway to eEstonia. ESP contains approximately 100 subpages including various pages for citizens and entrepreneurs. The WUI of ESP is extremely critical to be flawless as it provides services for all inhabitants of Estonia. Following web usability guidelines for critical portals is important as it is used by people with different experience and skills in using various WUIs, and also by people with disabilities.

The primary motivation for integrating automated usability and accessibility evaluation to RIA's development process from the author's point of view is to ensure that the *Guideliner* can be a beneficial tool within real development environment. From the perspective of RIA, the motivation is carried by the necessity to ensure the conformance of ESP's WUI to WCAG. ESP follows Directive (EU) 2016/2102 of the European Parliament and of the Council of 26 October 2016 [131] on the accessibility of websites and mobile applications of public sector bodies. The main purpose of the directive is to encourage all governmental applications to be more accessible by people with disabilities increasing the importance of following WCAG standard guidelines.

RIA is responsible for developing and maintaining the governmental gateway portal ESP, also known as eesti.ee portal. The portal is a primary gateway to public information and service in Estonia. It is a secure Internet environment that the residents of Estonia use to access the state's information systems and portals. The main reason why the Estonian State Portal was selected for the research is because it is the most popular public governmental portal with more than 40 mln. visits per year [2]. Figure 33 demonstrates the view of the ESP home page containing the most popular sections of the web application. The WUI of the ESP is based on HTML, CSS and JavaScript technologies and is designed using responsive web design approach. It means that the positions of elements, their size and design vary across the devices ensuring high usability on every device.

Despite the fact that there are multiple existing solutions for checking the conformance to WCAG standard (e.g. AChecker[2], Mauve[3] and TotalValidator[4]), there is no standard commonly available solution that can suit entirely RIA's development

[1] https://www.eesti.ee/en
[2] https://achecker.ca/checker/index.php
[3] http://mauve.isti.cnr.it
[4] https://www.totalvalidator.com

process model. For instance, AChecker, Mauve and TotalValidator are distributed as standalone web applications requiring a separate environment to run (e.g. AChecker requires Apache server with preinstalled PHP environment). RIA uses TotalValidator for periodical inspection of WUI compliance to usability guidelines. TotalValidator is distributed as a browser extension and usability experts use it during the pre-release testing to make sure that new functionality does not violate WCAG guidelines. Nevertheless, RIA's WUI development team moves towards full automation of the development process and they find that their current approach to automated usability evaluation (manual execution of usability tests with TotalValidator) does not comply with the RIA development process requiring that the tool for automated usability evaluation is built and run as a part of continuous integration process. RIA development team's vision is that usability evaluation is executed automatically every time developers commit changes to the source code of an application. Inability to fully automate usability evaluation is the main disadvantage of the current RIA setup.

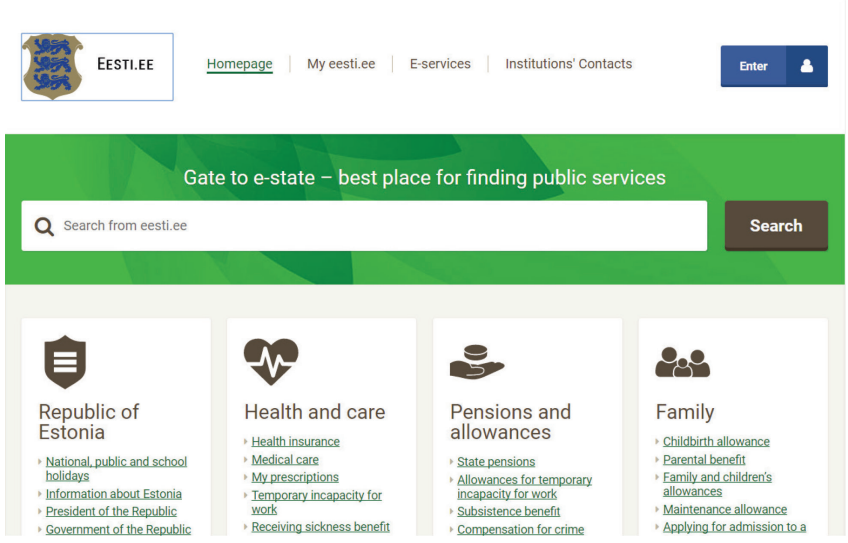


Figure 33. Screenshot of the Estonian State Portal eesti.ee containing the most popular sections of the web application (accessed 27.03.2018).

The *Guideliner* as an implementation-time usability evaluation tool addresses the discussed disadvantage and enables to fully automate usability evaluation process through integration into development and CI process described in Sections 5.3 and 5.4.

RIA carries out different types of usability tests in different phases of WUI development process. Figure 34 presents the usability evaluations conducted during different RIA development phases. Plurastic walkthrough, Heuristic evaluation and User tests are conducted manually during the Design phase; White box testing performed by developer is the only type of usability testing performed during the Implementation phase; Testing Phase is covered by TotalValidator accessibility tests and Expert usability review; during the Maintenance phase Google Analytics and Plumbr interaction based tests are used. The *Guideliner* usability tests marked with dashed-line rectangles on Figure 34 complement the Implementation and Testing phases of RIA WUI usability tests with automated usability Tests. The *Guideliner* usability tests during the Implementation phase allow executing usability tests on the developer's local environment. In addition, the *Guideliner* is integrated into the testing phase of WUI development performing

usability evaluation of WUI after each deployment on staging environment. These improvements are aimed to detect usability defects before manual usability evaluation.

RIA is a governmental company and it does not have its own development team. All changes made to the ESP are done by partner companies. The development process with partner companies (RIA has multiple different partners) can be different between the partners but in common it is organised in an agile way consisting of iterations that include planning, implementation, testing and releasing. In order to decide how to integrate the *Guideliner* into RIA's development process, RIA development process was reviewed in cooperation with RIA UX architects and WUI testing team. The development process is optimized with CI that automates the process of application building, running automated WUI tests and releasing the application to the staging environment). Figure 35 presents the Implementation phase of the WUI development process. According to RIA development process, developers develop WUI on local machines. After the development is finished, they commit modified code to the remote code repository, afterwards, the changes are merged to the shared branch and the Integration Server is notified about changes in code base. Then, the integration server checks out application source code, initiates the build, and performs WUI code unit tests. When all functionality for release is finalised by partner companies, RIA's testing teams perform release testing.

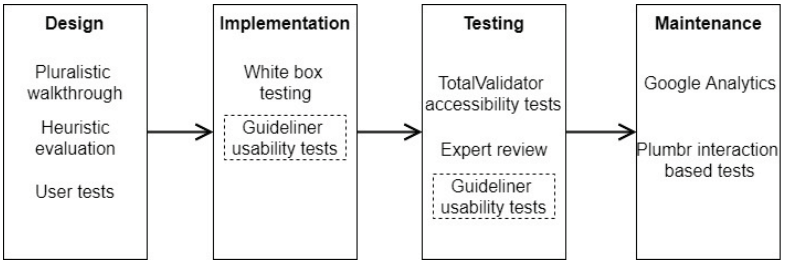


Figure 34. Usability tests carried out in different phases of the RIA WUI development process, and the fitting of the Guideliner tests into it.

Overall, based on the analysis of the development process, the process was suitable for the *Guideliner* to be integrated as a command line library into the implementation phase of the project together with other tests including function WUI tests and unit tests. Figure 35 highlights the step *Run the Guideliner automated usability tests* that was introduced to RIA's WUI development process with the addition of the *Guideliner*. The new step allows automatically to check the conformance of WUI to usability guidelines on the local environment as well as on the integration server.

One of the purposes of the *Guideliner* integration into the RIA development process was to investigate whether the *Guideliner* can be integrated into the existing project without major affection to the existing development process. The results of the integration to RIA's WUI development process proved that the *Guideliner* can be integrated as an additional activity (run automated usability tests) without any major modification of existing process activities.

The *Guideliner* usability tests integration into RIA development process was possible as RIA already used Selenium-based WUI functional tests and had infrastructure ready for such tests. For the projects that do not contain Selenium-based tests, the infrastructure (e.g. browser for Selenium tests) needs also to be configured, though,

the configuration is quite trivial. In addition to infrastructure configuration, integration server requires also alteration to run the *Guideliner* usability tests.

Another purpose of the *Guideliner* integration was to estimate its viability. As was mentioned before, RIA has contracts with partner companies to handle ESP development, which makes estimating the *Guideliner* economic benefit for RIA not so straightforward. At the moment of the *Guideliner* integration, RIA was in the process of changing its partner companies and organising a new public procurement. The change of development partner did not allow to conduct a long-time experiment, and it was decided to evaluate the economic benefit indirectly in cooperation with RIA's team (UX architect and Testing Lead) relying on their expertise and previous practice. RIA verifies the usability of developed WUI functionality when the partner company has finished the implementation and handed it over for evaluation. In case usability problems are detected, the development partner company is informed about the problems they need to address and provide fixes for.

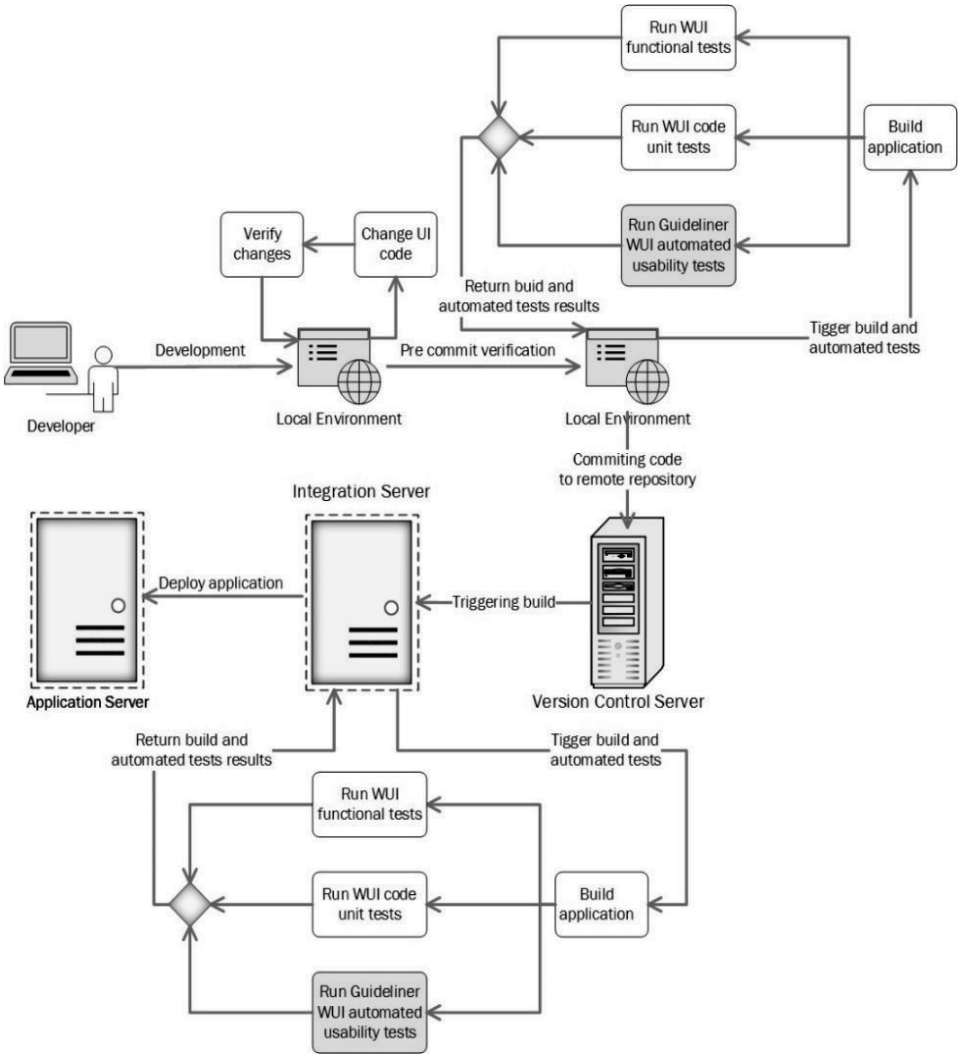


Figure 35. Implementation phase of WUI development process at RIA.

In order to evaluate the *Guideliner* integration benefit indirectly, the publications addressing bug fix price on different software development phases were analysed. The authors of different research works [128] [132] [133] agree that around 35% - 55% of the total software costs goes on testing, and that fixing the bug in implementation phase is in average 50% cheaper than detecting and fixing the same bug during the implementation phase. For that reason, these authors suggest to build testing process in a way that enables to detect defects as soon as possible. Before the *Guideliner* was integrated to RIA it took nine steps to detect and to fix a usability problem as Table 8 demonstrates (commit the code, make pull request, merge the code, deploy to staging environment, perform the usability testing, document the problem, send it back to developer, create branch and fix the problem). Together with RIA experts, the thesis author defined a weight of each step in RIA development process that shows the relative human resource consumption on each of the 9 steps. For instance, the weight of 45% means that it takes 45% of the time to finish the step Perform usability testing step. The most time-consuming step is usability evaluation as it is done manually and requires review of functionality from the perspective of usability, whereas most other steps require minor human effort. After the *Guideliner* integration, three steps that were executed manually (steps 5, 6, 7) are now fully automated and do not require human expert input anymore. As a result, the total weight of the next three steps: perform usability testing, document the problem and send it back to the developer decreased from 60% to 20%. The decrease was prompted by the improvements in time estimation that dropped by 40%. Based on the expert knowledge of RIA experts, it was concluded that the integration of the *Guideliner* could decrease the evaluation time around 40%.

Table 8. Steps to detect and fix usability problem in RIA.

#	Step	Weight	Type
1	Commit the code	5%	manual
2	Make pull request	5%	automatic
3	Merge the code	10%	manual
4	Deploy to staging environment	5%	automatic
5	Perform usability testing	45%	manual
6	Document the problem	10%	manual
7	Send it back to developer	5%	manual
8	Create branch	5%	automatic
9	Fix the problem	10%	manual

Figure 36 shows the average relative price of fixing the problem before and after the *Guideliner* integration according to the methodology presented before. It also highlights the shift in time of possible defect detection. Such improvement is accompanied by the decrease in the number of steps to be performed to detect and to fix a usability defect.

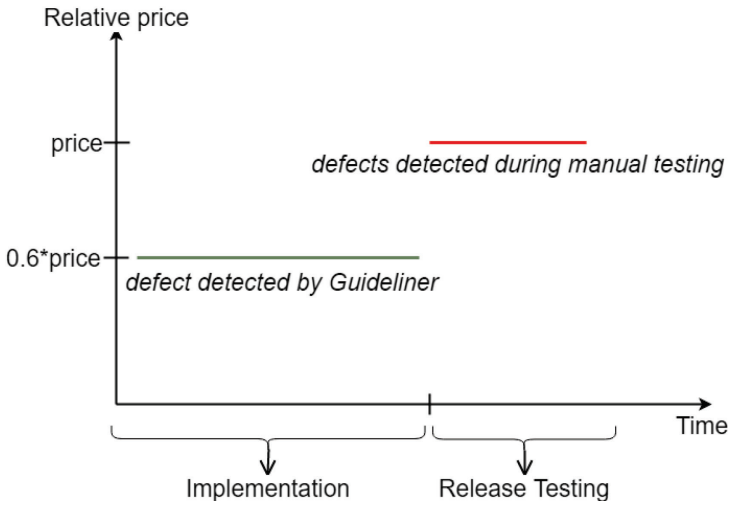


Figure 36. Average relative price for fixing the bugs found by the Guideliner during the implementation phase and during the release testing phase.

The economic benefit for RIA's development partner companies is that the usability defects detected by the *Guideliner* during the implementation phase will be fixed with smaller amount of steps (according to the estimations presented in [128] the price of a fix decreases up to 50%) than detecting the same problems during the testing phase. With such improvement, RIA usability testing team, in its turn, can concentrate on detecting more severe usability problems that cannot be automatically detected and require manual labour, delegating simple ones to the *Guideliner*.

Overall, the *Guideliner* automated usability tests were integrated into RIA implementation and testing phases of WUI development. The *Guideliner* integration obviated the need for manual usability evaluation with TotalValidator used before the *Guideliner* integration, and that was triggered manually as it was installed as a browser plugin. With the integration of the *Guideliner*, the automated usability evaluation is fully automated along with WUI functional and WUI code unit tests. The analysis of integration and its evaluation with RIA experts showed that the potential time, and as a result, the cost of fixing the bug detected with the *Guideliner* during the implementation phase could in average be 40% less expensive than if found manually during the testing phase. Next, the cooperation with RIA is continuing with encouraging RIA partner software development companies (companies that develop WUI for RIA applications) to integrate the *Guideliner* into their development process to assure that their code conforms to the accessibility and usability guidelines.

The integration of *Guideliner* into RIA's development process presented in current Section is a result obtained in the process of cooperation and supervision of a master student the author was supervising [134].

5.5.2 Integration into Kühne + Nagel WUI Development Process

Kuehne + Nagel (K+N) was the second organization into whose development process the *Guideliner* was introduced to test its viability. Kuehne + Nagel is a global transport and logistics company with headquarters in Switzerland. The company is focusing on providing IT-based logistics solutions for sea freight and air freight forwarding, contract logistics, and overland businesses. Kuehne + Nagel established an IT development centre

in Tallinn in 2012 with a name Kühne + Nagel IT Service Centre AS, and signed a cooperation agreement with Tallinn University of Technology with the purpose to develop new solutions in the IT area.

The purpose of the *Guideliner* integration into Kühne + Nagel IT Service Centre AS WUI development process was to ensure that the *Guideliner* provides enough functionality for integration into real WUI development process. In addition, the goal of the integration was to analyse how the *Guideliner* integration would economize the time spent on usability evaluation.

The web application under study in case of Kühne + Nagel IT Service Centre AS is the Customer Identification Program (CIP) that provides the functionality for managing customers of the company. The purpose of the web application is to deliver the information about company customers to all concerned departments. CIP is an internal application that must be used strictly inside the company's internal network that is why the discussions about various features, views and data the application contains is strictly limited herein. One of the main reasons why CIP was selected for the *Guideliner* integration into the WUI development process is to test and verify the *Guideliner* for the project at the beginning of a development phase. At the time of the *Guideliner* integration, CIP WUI development was at the initial stage suiting entirely the purpose of the integration.

The WUI of CIP is based on the technologies fully supported by the *Guideliner* – HTML, CSS and JavaScript technologies being developed with Angular 4 JavaScript Framework[1]. The custom application specific CSS style theme was developed based on Bootstrap 4[2]. Figure 37 shows the screenshot of CIP view made on Test Environment containing detailed information of a selected customer. At the time of evaluation, the application contained more than 15 various views.

As far as CIP is a business critical web application, all main WUI business workflows are covered with automated functional tests triggered and executed automatically after each commit verifying that the business workflow has not been affected by the changes to the WUI code. The evaluation of usability is not automated but rather manually reviewed by product owner, QA and developers when the feature is released to the staging environment; there is no evaluation of every modification of WUI conducted – this is the gap that the *Guideliner* addresses in K+N case providing the possibility to evaluate usability automatically after each modification of WUI. CIP has already existing functional Selenium based tests that are triggered and executed after every commit to the WUI code base. The existing testing model is extendable and provides an easy way to integrate the *Guideliner*.

K+N WUI development process from the technical perspective is similar to the RIA process discussed in Section 5.5.1. According to CIP WUI development process (Figure 35) at K+N, developers write code on their local machines, afterwards, they run WUI functional tests locally before committing modified code to the remote repository and if tests pass they commit changes to the remote repository. Then, the built is triggered on integration server where the application is built and tests are executed automatically. Afterwards, the application is deployed to the application server where QA verifies the changes done by developers. Usability testing occurs on application server after the whole release functionality is finalized.

[1] https://angular.io/

[2] https://getbootstrap.com

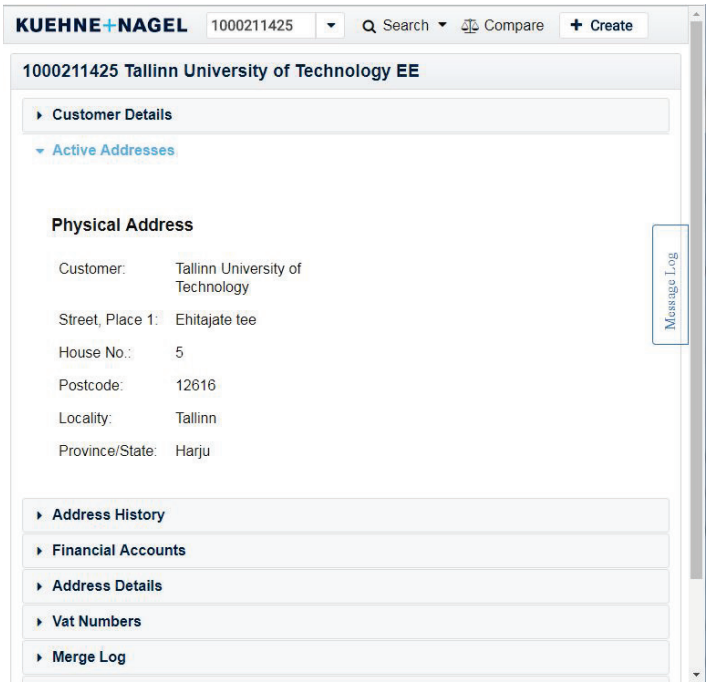


Figure 37. Screenshot of Customer Identification Program (CIP) containing detailed information about the customer (accessed 27.06.2017).

As was mentioned before, the CIP existing testing model that consists of automated code unit tests and WUI functional tests is extendable. The testing model was extended with additional type of tests – the *Guideliner* usability tests. The purpose of the *Guideliner* usability tests is to make sure that code modifications did not break the conformance of WUI to usability guidelines. Also, CIP WUI development process (see Figure 35) was affected – before deployment of CIP WUI to staging environment the conformance of WUI to usability guidelines is verified automatically: in case usability guidelines are violated, application will not be deployed to staging environment.

After the *Guideliner* was integrated into the development process, four developers from a team, who were mostly concentrating on the WUI development process, participated in the experiment. Developers were obliged to execute the *Guideliner* usability tests every time before they commit the code to the remote repository. The experiment lasted for 20 working days. During that period 17 tasks for WUI changes were resolved; 89 commits were done to the remote repository.

Developers were obliged to run the *Guideliner* usability tests before committing to the remote repository. The *Guideliner* detected 129 violations during the experiment for 17 tasks. All violations detected by the *Guideliner* in CIP were fixed. In rare cases, usability violations were detected by *Guideliner* on the staging environment but only in cases when developers did not run usability tests locally. That was the case when developers were fixing WUI bugs and wanted to deploy the fix as quickly as possible missing the execution of functional and usability tests. The integration of the *Guideliner* also encouraged team members to propose new CIP specific usability guidelines for the *Guideliner*. Proposed usability guidelines were based on their expert knowledge of detecting similar usability issues within different views of CIP.

CIP team follows Scrum-based process with two-week Sprint intervals. At the end of the Sprint, all tasks resolved during the Sprint were deployed to the production environment. After the experiment was finished, the benefits of the *Guideliner* usability tests were analysed from the perspective of time saved for detecting and fixing usability guidelines. Before the *Guideliner* was introduced to the development process, the process of handling usability defect consisted of the next steps done by QA: assign task, understand the change, detect the problem, document the problem, send it back to the developer, and the next steps done by developer: understand the problem, create the branch, make code review, deploy to test and inform QA). After the *Guideliner* was integrated, the usability defects were detected automatically after the code was committed to the remote repository before they reached QA, and developers were notified immediately about the introduced usability defects.

A QA engineer was asked to perform ten manual usability evaluations and to calculate the time required to detect and document found usability defects. The average time to detect and document usability defects was in average 27 minutes (Table 9); on average, two violations were detected during each evaluation. The same metrics (evaluation time and the number of detected violations) were also measured with the *Guideliner*. As the *Guideliner* performs the evaluation automatically, the average execution time of the *Guideliner* usability tests was in average 10 minutes (Table 9). The average number of detected usability guidelines by the *Guideliner* was also two. The usability issues detected by the *Guideliner* and by QA engineer were mostly overlapping with very few exceptions.

The experiment showed that the *Guideliner* detected certain usability defects that were missed by QA engineer. For example, the *Guideliner* reported the violation of usability guideline that defines the minimum distance between elements on WUI even when the distance between elements was few pixels smaller than allowed; QA engineer missed the violation as it is not straightforward to detect such violation manually. The experiment also showed that in certain cases the usability defects reported by QA engineer were not detected by the *Guideliner* as these guidelines were not defined in the *Guideliner*. For example, QA engineer detected that some labels of the buttons were not consistent between applications. It is complex to detect such violation automatically as the labels of the buttons depend on the context where they are used.

Based on the calculation presented before, the *Guideliner* detects usability defects 60% faster than an average human evaluator. Moreover, the *Guideliner* detects the problems before QA starts the process of usability evaluation meaning that QA gets more time for other activities, such as severe interaction-based usability problems that cannot be detected automatically by usability evaluation tools.

Overall, the purpose of the *Guideliner* integration was to ensure that it is suitable for the project that follows short (two-week) cycle software development process. The main difference with RIA's case is that RIA covered only technical owner role, and the development was done by partner companies. In contrary, CIP team covers both roles: technical owner of the project as well as a team that develops the project. Covering both roles allowed measuring the minimal time spent on usability problem detection before and after the *Guideliner* integration. Measurement results showed that after the *Guideliner* was integrated the time spent on usability problem detection decreased by approximately 60%. Both, RIA and KN integration use cases showed that the *Guideliner* integration into WUI development process enables immediate feedback for the developers decreasing the usability evaluation time around 40% - 60%.

Table 9. Result of QA manual usability evaluation.

Attempt nr.	Manual Evaluation Time (minutes)	Number of Usability Defects Reported	Automated Evaluation Time (minutes)	Number of Usability Defects Reported by Guideliner
1	23.50	4	11.20	5
2	22.45	0	8.78	1
3	25.90	0	9.45	0
4	25.49	2	13.98	1
5	24.34	3	11.45	4
6	27.83	0	10.09	0
7	32.56	3	8.68	4
8	30.63	4	11.58	2
9	31.34	0	11.43	0
10	28.45	5	8.34	4

5.6 Chapter Summary

Due to rapidly changing business requirements, WUI design and structure are almost always in continuous development to meet new requirements. Changes to WUI should be done with extreme caution as every even minor change of WUI can potentially lead to severe usability problems [12]. That is why immediate evaluation of WUI conformance to usability guidelines and subsequent feedback to WUI developer becomes another critical demand in WUI development and in particular during the implementation and coding phase of the WUI. Usability evaluation during the implementation phase of WUI development addresses the gap between the moment usability defect has been detected and the moment when it has been introduced by developer. Immediate usability evaluation is important because finding usability problems early in the implementation phase makes the fix less costly than found later.

This chapter focused on executing automatic WUI usability evaluation enabled by the *Guideliner* during WUI implementation phase as a part of its development process. Usage of the *Guideliner* was included into development process steps and the *Guideliner Core* was integrated into integrated development environment extending the capabilities of the original *Guideliner.* The approach was evaluated on two organizations and their software development processes – Estonian Information Systems Authority and Kühne + Nagel IT Service Centre AS. The primary motivation of the *Guideliner* evaluation was to ensure that *Guideliner* can be a beneficial tool within real development environment and that it suits for different WUI development processes. For this purpose, the *Guideliner* integration into the WUI software development process of aforementioned organisations was evaluated. The analysis of integration into RIA WUI development process showed that the cost of fixing the bug detected with the *Guideliner* during the implementation phase is in average 40% less than if found during the testing phase. Also, the measurement results showed that after the *Guideliner* was integrated the cost of detection of usability issues for KN has decreased by 60%.

Overall, the purpose of the Chapter was to integrate the *Guideliner* into IDE enabling automated usability evaluation already during the implementation phase of WUI development. The results of the experiment showed that the benefit of the *Guideliner* integration into implementation phase of WUI development is immediate feedback for the developers on the detected usability defects that decrease the cost of fixing usability defects.

The use cases of integration the *Guideliner* in real WUI development processes confirmed the effectiveness and usefulness of it. Using the *Guideliner* for implementation-time usability evaluation is not limited to any development process but can be exploited in any HTML-based WUI development. The solution proposed in this Chapter has been published in [13].

6 Conclusions and Future Work

„The details are not the details. They make the design. “
Sir Charles O. Eames[1]

The proliferation of the Internet, development of hardware and software, and the advances in technologies for content delivery has led to a situation where the Web can be accessed not only from desktop computers but on a multitude of different platforms including laptops, tablets and smartphones, all of which have become an indispensable part of our lives. However, this has also raised challenges for web user interface usability evaluation, encouraging developers and researchers to find new and cost-efficient approaches for immediate implementation-time usability evaluation, including computer-aided usability evaluation

This thesis has focused on modelling custom usability guidelines for automatic usability evaluation and on the automatic evaluation of WUI conformance to usability guidelines during both WUI implementation and WUI testing phases.

6.1 Conclusions

Current dissertation brings forward automated WUI usability evaluation. The dissertation analyses the methods used to formalise usability guidelines and proposes usability ontology to overcome the limitations of existing approaches such as inability to define visual usability guidelines. Another contribution of the thesis is a method for usability evaluation prevailing over analogue solutions. The novelty of proposed method is that it is overcoming the limitations of existing solutions (that are limited only to the evaluation of HTML-specific usability guidelines) evaluating both HTML centric accessibility guidelines as well as visual usability guidelines that are evaluated on finally rendered WUI designed especially for mobile platform as well as guidelines specific for desktop devices only. Finally, the thesis concentrates on automated usability evaluation during the implementation phase of WUI development addressing the gap between the moment usability defect is detected and the moment when it was introduced by developer. Immediate usability evaluation is important because finding usability problems early in the implementation phase makes the fix less costly than found later. The primary outcome of the dissertation is the tool for automated web user interface usability evaluation called the *Guideliner.* The *Guideliner* incorporates research outcomes of current thesis including established usability ontology for capturing usability domain knowledge, a novel method for automatic usability evaluation of visual usability guidelines as well as HTML-centric guidelines, and implementation time usability evaluation.

Current thesis concentrates on unresolved problems in usability evaluation such as the inability to evaluate visual characteristics of WUI and insufficient support for implementation-time usability evaluation. Aforementioned shortcomings are the motivation for current research that distinguishes the research presented in current thesis from the research works of other researchers in this area who are mostly

[1] Charles Ormond Eames (1907–1978), American designer who made significant historical contributions to the development of modern architecture and furniture.

concentrated on the automated evaluation of WUI conformance to HTML-centric guidelines during the testing phase of WUI development.

The main contributions of the thesis are as follows:

- Establishment of the structure of and the usability ontology to capture usability domain knowledge and to formalise usability guidelines in machine-processable and human-readable format (RT3). The usability ontology contains predefined set of usability guidelines and also allows defining additional custom usability guidelines based on the established usability domain knowledge, thus it is extendible.
- A novel method for evaluating visual characteristics of WUI based on usability guidelines is introduced (RT4). The novelty of that method relies on that it evaluates WUI conformance to usability guidelines on the finally rendered result (after page has finished its loading and all scripts have been run). In addition to evaluating HTML code based guidelines, it also allows evaluating WUI visual characteristics such as position of elements on the screen, the distance between the elements, presence, the length of scrolling and the contrast rate of elements.
- A method for immediate evaluation of WUI conformance to usability guidelines during the implementation phase of WUI development (RT2). It addresses the problem of how to effectively and efficiently evaluate web user interfaces usability during their development, in particular within the implementation phase with minimal cost and time. Such enhancement allows fixing usability defects early in the development making the fix cheaper and less time-consuming than found later.
- A tool for automated usability evaluation called the *Guideliner*, which is based on the aforementioned established methods (RT1, RT2). The *Guideliner* uses usability ontology as a source of usability guidelines; it evaluates finally rendered WUI assessing HTML-based usability guidelines and guidelines covering visual characteristics of WUI, and is applicable both at the final established WUI evaluation phase (traditional approach), and for evaluating WUI conformance to set usability criteria during WUI implementation phase (novel method introduced in the thesis). The distinctiveness of the *Guideliner* consists of implementation-time automated usability evaluation and in automated evaluation of multiple visual characteristics of WUI.

The *Guideliner* is applicable to any HTML, CSS and JavaScript-based WUI and does not depend on any WUI development process. In order to prove its efficiency for real development environment, the *Guideliner* has been applied to the public sector portal and to the internal web application. The results of the integration confirmed the viability and the effectiveness of the *Guideliner* in terms of implementation-time usability evaluation.

There are no restrictions to exploit the *Guideliner* in other areas such as online banking, e-commerce applications and social network web applications; this is possible due to the extendibility of the *Guideliner* to define any web application specific usability guidelines using designed usability ontology.

The economic benefit gained from the *Guideliner* for automated usability evaluation is considerable. In general, cost of change or bug fixing increases with every following phase of software development. The *Guideliner* delivers the ability to detect usability

defects during implementation phase of WUI development whereas traditionally usability is evaluated during the testing phase. Implementation-time usability evaluation allows fixing usability issues immediately after detection reducing the cost of fixing.

Another benefit gained from the *Guideliner* is automated evaluation of visual usability guidelines of finally rendered WUI that increases the coverage of usability problems handled by automated usability evaluation. It means that usability evaluation performed during testing phase can be concentrated on more severe usability problems that cannot be automatically detected such as certain interaction-based usability problems delegating simpler ones to the *Guideliner*.

To the best of the author's knowledge, the new methods presented and discussed in current thesis were not presented at the time usability evaluation studies were started.

By its contribution, the dissertation has addressed two problems: implementation-time efficient automated evaluation of WUI conformance to HTML centric accessibility guidelines as well as to visual usability guidelines (that are evaluated on finally rendered WUI). Another topic addressed in dissertation is usability ontology as a unified way of formalising usability domain knowledge through various aspects, being both human- and machine-readable.

6.2 Future Work

The primary research area of the thesis is automated usability evaluation. Various directions can be taken to advance the research described in this dissertation further:

First, extend existing predefined sets of usability guidelines with e-commerce, online banking and social network specific usability guidelines. With such development, the tool will be applicable to more WUIs.

Second, add additional predefined usability guidelines from e-commerce, banking, search engine optimization and image processing domains. For example, it is possible to evaluate the correctness of alternative text using image content analysis and detection services. Such enhancement will increase the variety of usability characteristics that can be evaluated automatically.

Third, continue cooperation with RIA to encourage RIA partner software development companies (companies that develop WUI for RIA applications) to integrate the *Guideliner* into their development process to assure that their code conforms to the accessibility and usability guidelines.

Fourth, increase the popularity of the *Guideliner* in the area of usability evaluation by introducing it at conferences and workshops, presentations) for WUI developers, and promoting it on GitHub.

Fifth, the formation of the *Guideliner* community by increasing the number of contributors and end users. As a result, more people will be involved in the *Guideliner* development and its exploitation.

Sixth, continuation of academic research in the field of automated usability evaluation by involving master students in the further development of the *Guideliner* and presenting the outcomes on scientific conferences.

The research presented in this dissertation has established a basis for further studies on automated implementation–time usability evaluation as well as automated evaluation of WUI visual characteristics.

References

[1] Ofcom, “Adults’ media use and attitudes report,” Ofcom, 2014.

[2] “The State Portal eesti.ee in numbers (2016),” 2016. [Online]. Available: https://www.eesti.ee/eng/topics/business/riigiportaali_abi/partnerile_1/eesti_ee_2016_aasta_statistika. [Accessed 26 97 2018].

[3] A. McCartney, M. Funk, J. Mercay and H. Marmy, “System and method for dynamically publishing XML-compliant documents,” Google Patents, 2002.

[4] A. Andrews, J. Offutt and R. Alexander, “Testing web applications by modeling with FSMs,” *Software & Systems Modeling,* vol. 4, no. 3, pp. 326-345, 2005.

[5] Y.-F. Li, P. Das and D. Dowe, “Two decades of Web application testing—A survey of recent advances,” *Information Systems,* vol. 43, pp. 20-54, 2014.

[6] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke and S. Raghavan, “Searching the web,” *ACM Transactions on Internet Technology (TOIT),* vol. 1, no. 1, pp. 2-43, 2001.

[7] E. DIn, “Ergonomic requirements for office work with visual display terminals (VDTs),” *International Organization for Standardization,* vol. 11, 1998.

[8] J. Nielsen and H. Loranger, Prioritizing web usability, Pearson Education, 2006.

[9] J. Dumas, J. Dumas and J. Redish, A practical guide to usability testing, Intellect books, 1999.

[10] A. G. Schiavone and F. Paterno, “An extensible environment for guideline-based accessibility evaluation of dynamic Web applications,” *Universal access in the information society,* vol. 14, no. 1, pp. 111-132, 2015.

[11] G. Gay and C. Q. Li, “AChecker: open, interactive, customizable, web accessibility checking,” in *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, Raleigh, 2010.

[12] J. Marenkov, T. Robal and A. Kalja, “A Framework for Improving Web Application User Interfaces Through Immediate Evaluation.,” in *Databases and Information Systems IX*, Amsterdam, 2016.

[13] J. Marenkov, T. Robal and A. Kalja, “A Tool for Design-Time Usability Evaluation of Web User Interfaces,” in *Advances in Databases and Information Systems*, Cham, 2017.

[14] J. Marenkov, T. Robal and A. Kalja, “A Study on Immediate Automatic Usability Evaluation of Web Application User Interfaces,” in *International Baltic Conference on Databases and Information Systems*, Cham, 2016.

[15] A. Dingli and J. Mifsud, “Useful: A framework to mainstream web site usability through automated evaluation,” *International Journal of Human Computer Interaction (IJHCI),* vol. 2, no. 1, p. 10, 2011.

[16] A. Dingli and S. Cassar, “An intelligent framework for website usability,” *Advances in Human-Computer Interaction,* vol. 2014, p. 5, 2014.

[17] A. Zuo, J. Yang and X. Chen, “Research of agile software development based on formal methods,” in *International Conference on Multimedia Information Networking and Security*, Nanjing, 2010.

[18] J. Nielsen, Usability engineering, Elsevier, 1994.

[19] A. Abran, A. Khelifi, W. Suryn and A. Seffah, "Usability meanings and interpretations in ISO standards," *Software quality journal,* vol. 11, no. 4, pp. 325-338, 2003.

[20] J. Fisher, J. Bentley, R. Turner and A. Craig, "SME myths: if we put up a website customers will come to us-why usability is important," in *eIntegration in action: 18th Bled eConference, Bled, Slovenia, June 6-8, 2005*, Bled, 2005.

[21] C. Flavián, G. Miguel and G. Raquel , "The role played by perceived usability, satisfaction and consumer trust on website loyalty," *Information & management,* vol. 43, no. 1, pp. 1-14, 2006.

[22] D.-H. Byun and G. Finnie, "Evaluating usability, user satisfaction and intention to revisit for successful e-government websites," *Electronic government, an international journal,* vol. 8, no. 1, pp. 1-19, 2010.

[23] C. Downing and C. Liu, "Assessing web site usability in retail electronic commerce," in *Proceedings of the 35th Annual Computer Software and Applications Conference (COMPSAC)*, Munich, 2011.

[24] "Accessibility, Usability, and Inclusion: Related Aspects of a Web for All," W3C, 06 05 2016. [Online]. Available: https://www.w3.org/WAI/intro/usable. [Accessed 23 06 2018].

[25] F. Adebesin, P. Kotze and H. Gelderblom, "The complementary role of two evaluation methods in the usability and accessibility evaluation of a non-standard system," in *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, Bela Bela, 2010.

[26] "Web Accessibility," European Commision, 2010. [Online]. Available: http://ec.europa.eu/ipg/standards/accessibility/index_en.htm. [Accessed 09 05 2018].

[27] "Interoperability of the State Information System," Ministry of Economic Affairs and Communication.

[28] "Avaliku sektori veebilehtede vastavus WCAG 2.0 nõuetele 2015. aastal," Majandus- ja Kommunikatsiooniministeerium, 2015.

[29] S. Smith and J. Mosier, Guidelines for designing user interface software, Bedford: Mitre Corporation, 1986.

[30] P. Kara, A. Schade and J. Nielsen, "Intranet Usability Guidelines: Findings from User Testing of 42 Intranets," 2014.

[31] R. Budiu and J. Nielsen, User experience for mobile applications and websites: design guidelines for improving the usability of mobile sites and apps, DMIT, 2015.

[32] "Mobile Web Best Practices 1.0," W3C, 29 07 2008. [Online]. Available: https://www.w3.org/TR/mobile-bp/. [Accessed 06 05 2018].

[33] A. Wessels, M. Purvis and S. Rahman, "Usability of Web Interfaces on Mobile Devices," in *Proceedings of the Eighth International Conference on Information Technology: New Generations*, Las Vegas, 2011.

[34] J. Marenkov, T. Robal and A. Kalja, “A study on effective knowledge reuse in multi-platform web applications user interfaces,” in *Management of Engineering and Technology (PICMET)*, Portland, 2015.

[35] W. Quesenbery, “Building a better style guide,” in *Proceedings of UPA*, 2001.

[36] D. Kim, “An investigation of the effect of online consumer trust on expectation, satisfaction, and post-expectation,” *Information Systems and E-Business Management,* vol. 10, no. 2, pp. 219-240, 2012.

[37] M. Abdeldayem, “A study of customer satisfaction with online shopping: evidence from the UAE,” *International Journal of Advanced Media and Communication,* vol. 4, no. 3, pp. 235-257, 2010.

[38] L. Kantner, D. H. Sova and S. Rosenbaum, “Alternative methods for field usability research,” in *Proceedings of the 21st annual international conference on Documentation*, New York, 2003.

[39] M. Vuolle, M. Tiainen, T. Kallio, T. Vainio, M. Kulju and H. Wigelius, “Developing a questionnaire for measuring mobile business service experience,” in *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, Amsterdam, 2008.

[40] B. Leporini, F. Paterno and A. Scorcia, “Flexible tool support for accessibility evaluation,” *Interacting with computers,* vol. 18, no. 6, pp. 869-890, 2006.

[41] C. Cuadrat Seix, M. S. Veloso and J. J. R. Soler, “Towards the validation of a method for quantitative mobile usability testing based on desktop eyetracking,” in *Proceedings of the 13th International Conference on Interaccion Persona-Ordenador*, New York, 2012.

[42] M. Ekşioğlu, K. Esin, Ç. Burak, S. Murat and O. Selen, “Heuristic evaluation and usability testing: case study,” in *International Conference on Internationalization, Design and Global Development*, Berlin, 2011.

[43] Y.-J. Lee and C.-J. Lin, “Evaluation and satisfaction survey on the interface usability of online publishing software,” *Mathematical Problems in Engineering,* vol. 2014, no. 10, p. 10, 2014.

[44] G. Brône, B. Oben and T. Goedemé, “Towards a more effective method for analyzing mobile eye-tracking data: integrating gaze data with object recognition algorithms,” in *Proceedings of the 1st international workshop on pervasive eye tracking & mobile eye-based interaction*, New Yor, 2011.

[45] S. Cheng, “The research framework of eye-tracking based mobile device usability evaluation,” in *Proceedings of the 1st international workshop on pervasive eye tracking & mobile eye-based interaction*, New York, 2011.

[46] M. Ivory and M. Hearst, “The state of the art in automating usability evaluation of user interfaces,” *ACM Computing Surveys (CSUR),* vol. 33, no. 4, pp. 470-516, 2001.

[47] J. O. Bak, K. Nguyen, P. Risgaard and J. Stage, “Obstacles to usability evaluation in practice: a survey of software development organizations,” in *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*, Lund, 2008.

[48] A. Häkli, Introducing user-centered design in a small-size software development organization, Helsinki: Helsinki University of Technology, 2005.

[49] K. Norman and E. Panizzi, “Levels of automation and user participation in usability testing,” *Interacting with computers,* vol. 18, no. 2, pp. 246-264, 2005.

[50] M. H. Blackmon, P. Polson, M. Kitajima and C. Lewis, “Cognitive walkthrough for the web,” in *Proceedings of the SIGCHI conference on human factors in computing systems*, Minneapolis, 2002.

[51] T. Hollingsed and D. Novick, “Usability inspection methods after 15 years of research and practice,” in *Proceedings of the 25th annual ACM international conference on Design of communication*, El Paso, 2007.

[52] J. Nielsen and R. Molich, “Heuristic evaluation of user interfaces,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, Seattle, 1990.

[53] H. Al-Khalifa, B. Al-Twaim and B. AlHarbi, “A heuristic checklist for usability evaluation of Saudi government mobile applications,” in *Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services*, Singapore, 2016.

[54] O. Al Osaimi and A. AlSumait, “Usability Guidelines for Arab E-government Websites,” *World Acad. Sci. Eng. Technol. Int. J. Soc. Behav. Educ. Econ. Bus. Ind.,* vol. 7, no. 12, pp. 3159-3162, 2013.

[55] N. Al-Wabil, “Usability of mobile applications in Saudi higher education: An exploratory study,” in *International Conference on Human-Computer Interaction*, Cham, 2015.

[56] A. de Lima Salgado, S. S. Rodrigues and R. P. Fortes, “Evolving Heuristic Evaluation for multiple contexts and audiences: Perspectives from a mapping study,” in *Proceedings of the 34th ACM International Conference on the Design of Communication*, New York, 2016.

[57] E. De Kock, J. Van Biljon and M. Pretorius, “Usability evaluation methods: Mind the gaps,” in *Proceedings of the 2009 annual research conference of the south african institute of computer scientists and information technologists*, Vanderbijlpark, 2009.

[58] L. Kuparinen, J. Silvennoinen and H. Isomäki, “Introducing usability heuristics for mobile map applications,” in *Proceedings of the 26th International Cartographic Conference*, Dresden, 2013.

[59] R. Yáñez Gómez, D. Cascado Caballero and J.-L. Sevillano, “Heuristic evaluation on mobile interfaces: A new checklist,” *The Scientific World Journal,* vol. 2014, 2014.

[60] F. Patterno and A. G. Schiavone, “The role of tool support in public policies and accessibility,” *IX Interactions,* vol. 22, no. 3, pp. 60-63, 2015.

[61] M. Arrue, M. Vigo and J. Abascal, “Including heterogeneous web accessibility guidelines in the development process,” in *Engineering Interactive Systems*, Berlin, Springer, 2008, pp. 620-637.

[62] N. S. M. Yusop, J. Grundy and R. Vasa, “Reporting usability defects: do reporters report what software developers need?,” in *Proceedings of the 20th international conference on evaluation and assessment in software engineering*, Limerick, 2016.

[63] A. Følstad, L. Effie and H. Kasper , “Analysis in practical usability evaluation: a survey study,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Austin, 2012.

[64] A. Johnston and M. Pickrell, “Designing for technicians working in the field: 8 usability heuristics for mobile application design,” in *Proceedings of the 28th Australian Conference on Computer-Human Interaction*, Launceston, 2016.

[65] M. McCloskey, H. Loranger and J. Nielsen, Teenagers (ages 13-17) on the Web, Nielsen Norman Group, 2013.

[66] R. Atterer, M. Wnuk and A. Schmidt, “Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction,” in *Proceedings of the 15th international conference on World Wide Web*, Edinburgh, 2006.

[67] G. Saadawi, E. Legowski, O. Medvedeva, G. Chavan and R. Crowley, “A method for automated detection of usability problems from client user interface events,” in *AMIA Annual Symposium Proceedings*, 2005.

[68] J. Grigera, A. Garrido and J. M. Rivero, “A tool for detecting bad usability smells in an automatic way,” in *International Conference on Web Engineering*, Cham, 2014.

[69] K. Kappel, M. Tomitsch, T. Koltringer and T. Grechenig, “Developing user interface guidelines for DVD menus,” in *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, Montréal, 2006.

[70] J. Coelho, C. Duarte, P. Biswas and P. Langdon, “Developing accessible TV applications,” in *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*, Dundee, 2011.

[71] V. Moshnyaga, “Guidelines for developers of smart systems,” in *8th International Conference on Intelligent Systems (IS)*, Sofia, 2016.

[72] A. Sanchez, O. Starostenko, E. Castillo and M. Gonzalez, “Generation of usable interfaces for mobile devices,” in *Proceedings of the 2005 Latin American conference on Human-computer interaction*, Cuernavaca, 2005.

[73] M. Evans, “Challenges in developing research-based Web design guidelines,” *IEEE transactions on professional communication,* vol. 43, no. 3, pp. 302-312, 2000.

[74] A. Hussain and E. Ferneley, “Usability metric for mobile application: a goal question metric (GQM) approach,” in *Proceedings of the 10th international Conference on information integration and Web-Based Applications & Services*, New York, 2008.

[75] G. Conti and E. Sobiesk, “Malicious interface design: exploiting the user,” in *Proceedings of the 19th international conference on World wide web*, Raleigh, 2010.

[76] C. M. Bailey and C. D. Seals, “Evaluation of Web Usability Guidelines for Teens,” in *Proceedings of the SouthEast Conference*, Kennesaw, 2017.

[77] O. Machado and M. Pimentel, “Heuristics for the assessment of interfaces of mobile devices,” in *Proceedings of the 19th Brazilian symposium on Multimedia and the web*, Salvador, 2013.

[78] B. Shneiderman and M. Leavitt, Research-based web design and usability guidelines, U.S. Deptartment of Health and Human Ser, 2006.

[79] "Ecommerce User Experience," Nielsen Norman Group, 2016.

[80] L. Chittaro, "Designing visual user interfaces for mobile applications," in *Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems*, Pisa, 2011.

[81] R. Alsalamen and G. Shahin, "Screen size effect on usability of Arabic forms for smartphones," in *International Conference on New Trends in Computing Sciences*, Amman, 2017.

[82] A. de Lima Salgado and A. P. Freire, "Heuristic evaluation of mobile usability: A mapping study," in *International Conference on Human-Computer Interaction*, Cham, 2014.

[83] R. Inostroza, C. Rusu, S. Roncagliolo and V. Rusu, "Usability heuristics for touchscreen-based mobile devices: update," in *Proceedings of the 2013 Chilean Conference on Human-Computer Interaction*, Temuco, 2013.

[84] J. Varsaluoma, "Scenarios in the heuristic evaluation of mobile devices: emphasizing the context of use," in *International Conference on Human Centered Design*, Berlin, 2009.

[85] E. Bertini, T. Catarci, S. Gabrielli, S. Kimani and G. Santucci, "Appropriating heuristic evaluation for mobile computing," *International Journal of Mobile Human Computer Interaction,* vol. 1, no. 1, pp. 20-41, 2009.

[86] A. Bruun and J. Stage, "New approaches to usability evaluation in software development," *Journal of Systems and Software,* vol. 105, no. C, pp. 40-53, 2015.

[87] M. Bakaev, T. Mamysheva and M. Gaedke, "Current trends in automating usability evaluation of websites: Can you manage what you can't measure?," in *11th International Forum on Strategic Technology*, Novosibirsk, 2016.

[88] Y.-C. Lin, C.-H. Yeh and C.-C. Wei, "How will the use of graphics affect visual aesthetics? A user-centered approach for web page design," *International Journal of Human-Computer Studies,* vol. 71, no. 3, pp. 217-227, 2013.

[89] D. Kieras, "Model-based evaluation," *The Human-Computer Interaction: Development Process,* pp. 294-310, 2009.

[90] F. Au, S. Baker, I. Warren and G. Dobbie, "Automated usability testing framework," in *Proceedings of the ninth conference on Australasian user interface*, Wollongong, 2008.

[91] S. Dixit and V. Padmadas, "Automated Usability Evaluation of Web Applications," in *Proceedings of the International Congress on Information and Communication Technology*, Singapore, 2016.

[92] L. Goncalves, L. Vasconcelos, E. Munson and L. Baldochi, "Supporting adaptation of web applications to the mobile environment with automated usability evaluation," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, Pisa, 2016.

[93] M. Speicher, A. Both and M. Gaedke, "Ensuring web interface quality through usability-based split testing," in *International Conference on Web Engineering*, Cham, 2014.

[94] N. Fernandes, N. Kaklanis, K. Votis, D. Tzovaras and L. Carricco, “An analysis of personalized web accessibility,” in *Proceedings of the 11th Web for All Conference*, Seoul, 2014.

[95] A. Beirekdar, J. Vanderdonckt and M. Noirhomme-Fraiture, “A framework and a language for usability automatic evaluation of web sites by static analysis of HTML source code,” in *Computer-Aided Design of User Interfaces III*, Dordrecht, Springer, 2002, pp. 337-348.

[96] Y. Takata, T. Nakamura and H. Seki`, “Accessibility verification of WWW documents by an automatic guideline verification tool,” in *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, Big Island, 2004.

[97] J. Ladry, P. Palanque, E. Barboni and D. Navarre, “Model-based usability evaluation and analysis of interactive techniques,” in *Proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010)*, Aachen, 2010.

[98] S. Humayoun and Y. Dubinsky, “MobiGolog: formal task modelling for testing user gestures interaction in mobile applications,” in *Proceedings of the 1st International Conference on Mobile Software Engineering and Systems*, Hyderabad, 2014.

[99] N. Liyanage and K. Vidanage, “Site-ability: A website usability measurement tool,” in *Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, Negombo, 2016.

[100] W. A. R. Isa, M. Yusoff and D. A. A. Nordin, “Evaluating the usability of homestay websites in malaysia using automated tools,” in *International Conference on Soft Computing in Data Science*, Singapore, 2015.

[101] R. Abdullah and K. T. Wei, “Usability measurement of Malaysia online news websites,” *International Journal of Computer Science and Network Security,* vol. 8, no. 5, pp. 159-165, 2008.

[102] E. Friess, “Discourse variations between usability tests and usability reports,” *Journal of Usability Studies,* vol. 6, no. 3, pp. 102-116, 2011.

[103] “Web Content Accessibility Guidelines,” W3C, 22 05 2018. [Online]. Available: http://www.w3.org/WAI/intro/wcag . [Accessed 14 06 2018].

[104] “Web Content Accessibility Guidelines (WCAG) 2.0,” W3C, 11 12 2008. [Online]. Available: https://www.w3.org/TR/WCAG20/. [Accessed 28 07 2018].

[105] “Understanding SC 2.4.2,” W3C, 2016. [Online]. Available: https://www.w3.org/TR/UNDERSTANDING-WCAG20/navigation-mechanisms-title.html . [Accessed 26 07 2018].

[106] R. Bailey, C. Barnum, J. Bosley, B. Chaparro and J. Dumas, Research-Based Web Design & Usability Guidelines, Washington, 2006.

[107] M. Klein, D. Fensel, F. Van Harmelen and I. Horrocks, “The relation between ontologies and XML schemas,” *Electronic Trans. on Artificial Intelligence,* vol. 6, no. 4, 2001.

[108] T. Gruber, “Toward principles for the design of ontologies used for knowledge sharing?,” *International journal of human-computer studies,* vol. 43, no. 5-6, pp. 907-928, 1995.

[109] L. Yu, A developer's guide to the semantic Web, Springer Science & Business Media, 2011.

[110] J. Xiong, C. Farenc and M. Winckler, "Towards an ontology-based approach for dealing with web guidelines," in *International Conference on Web Information Systems Engineering*, Berlin, 2008.

[111] J. Borges, I. Morales and N. Rodríguez, "Guidelines for designing usable world wide web pages," in *Proceedings of the CHI '96 Conference Companion on Human Factors in Computing Systems*, Vancouver, 1996.

[112] "Mobile Overview Report," scientiamobile, 2018.

[113] T. Robal, J. Marenkov and A. Kalja, "Ontology design for automatic evaluation of web user interface usability," in *Management of Engineering and Technology (PICMET)*, Portland, 2017.

[114] F. Lizano, M. Sandoval, A. Bruun and J. Stage, "Is usability evaluation important: the perspective of novice software developers," in *Proceedings of the 27th International BCS Human Computer Interaction Conference*, London, 2013.

[115] C. Timbi-Sisalima, C. Amor, S. Oton, J. Hilera and J. Aguado-Delgado, "Comparative Analysis of Online Web Accessibility Evaluation Tools," in *Information Systems Development: Complexity in Information Systems Development (ISD2016 Proceedings)*, Katowice, 2016.

[116] W. Hürsch and C. Videira Lopes, Separation of Concerns, 1995.

[117] "OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax," W3C, 11 12 2012. [Online]. Available: https://www.w3.org/TR/owl2-syntax/. [Accessed 05 05 2018].

[118] M. Horridge and S. Bechhofer, "The owl api: A java api for owl ontologies," *Semantic Web,* vol. 2, no. 1, pp. 11-21, 2011.

[119] D. Tsarkov and I. Horrocks, "FaCT++ description logic reasoner: System description," in *International Joint Conference on Automated Reasoning*, Heidelberg, 2006.

[120] N. Snellman, A. Ashraf and I. Porres, "Towards automatic performance and scalability testing of rich internet applications in the cloud," in *37th EUROMICRO Conference on Software Engineering and Advanced Applications*, Oulu, 2011.

[121] P. Claus and M. Boŝković, "Model-driven performance evaluation for service engineering," *Emerging Web Services Technology,* vol. 2, pp. 171-185, 2008.

[122] J. Marenkov, T. Robal and A. Kalja, "Design-Time Web Usability Evaluation with Guideliner," *Complex Systems Informatics and Modeling Quarterly,* no. 15, pp. 90-109, 2018.

[123] D. Beyer, "Status report on software verification," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Berlin, 2014.

[124] A. Ismail and K. Kuppusamy, "Accessibility of Indian universities' homepages: An exploratory study," *Journal of King Saud University-Computer and Information Sciences,* vol. 30, no. 2, pp. 268-278, 2016.

[125] K. Kert, Automatic Website Usability Evaluation, Tools and their Comparison, Tallinn: Tallinna Tehnikaülikool, 2018.

[126] B. Boehm, Software engineering economics, Prentice-hall Englewood Cliffs, 1981.

[127] S. McConnell, Code complete, Pearson Education, 2004.

[128] G. Tassey, “The economic impacts of inadequate infrastructure for software testing,” U.S Department of Commerce, 2002.

[129] M. Fowler and M. Foemmel, “Continuous integration,” *Thought-Works,* vol. 122, p. 14, 2006.

[130] J. Humble and D. Farley, Continuous delivery: reliable software releases through build, test, and deployment automation, Pearson Education, 2010.

[131] “Directive (EU) 2016/2102 of the European Parliament and of the Council of 26 October 2016 on the accessibility of the websites and mobile applications of public sector bodies (Text with EEA relevance),” 2016.

[132] D. Levine, J. Mason and J. Hadley, “Software development and quality assurance,” *Computer security handbook,* pp. 1-39, 2012.

[133] J. Tan, K. Otto and K. Wood, “Relative impact of early versus late design decisions in systems development,” *Design Science,* vol. 3, pp. 777-780, 2017.

[134] R. Roben, Automation of Usability Evaluation Based on the Example of Republic of Estonia Information System Authority, Tallinn: Tallinn University of Technology, 2018.

Acknowledgements

I would like to express my appreciation to all people who advised and helped me during PhD studies.

More specifically, I wish to thank my supervisor doctor Tarmo Robal for giving advice, support, cooperation during my PhD studies, and for giving me opportunity and encouraging me to bring the research work to completion.

I would like to offer special thanks to professor Ahto Kalja (deceased 22.10.2018), who, although no longer with us, continues to inspire by his example and dedication to the academic world he served over the course of his career.

I would like to thank everyone I worked together with during my PhD studies, especially the whole team at the Department of Computer Systems. I would like to show my appreciation to Dr. Margus Kruus for an excellent cooperation in dealing with administrative issues.

I am grateful to Dr. Hele-Mai Haav from Department of Software Science at Tallinn University of Technology for her consultation on research topics.

Also, I would like to thank professor Maarja Kruusmaa for providing valuable feedback about my research during regular evaluations organized at the School of Information Technologies.

I would like to thank Kristi Kert and Raido Roben for conducting usability experiments and inspiring with some ideas for future research.

Additionally, I would like to thank development and testing team of the Information System Authority of the Republic of Estonia for their valuable ideas and practical suggestions for the *Guideliner* development.

I would also like to express thanks to all my friends and colleagues unmentioned here.

Finally, I would like to thank my family for their understanding and support during my PhD studies. Thank you!

Abstract

Automatic Implementation-Time Usability Evaluation for Web User Interfaces

The evaluation of newly established or revamped web user interfaces and assurance that they conform to usability guidelines should not be resource-consuming and laborious task that many companies are attracted to skip. Instead, it should be a cost- and time-efficient step of development applied as a custom practice, especially when software gets developed according to agile methods consisting of short and flexible iterations with frequent releases to production. Yet, usability evaluation is still not optimized in a way that it can be automated excluding the involvement of usability experts into the evaluation. Automated cost- and time-efficient usability evaluation is one of the driving forces of this thesis.

Overall, current dissertation is focusing on the problem of automated usability evaluation during the implementation phase of WUI development, including visual characteristics of WUIs – a feature missing in tools available today.

Firstly, usability ontology to capture usability domain knowledge and to formalise usability guidelines in machine-processable and human-readable format was established. Usability ontology contains predefined set of usability guidelines and also allows defining additional custom usability guidelines based on the established usability domain knowledge.

Secondly, a method for evaluating visual characteristics of WUI based on usability guidelines is introduced. The novelty of that method is that it evaluates WUI conformance to visual as well as HTML-centric usability guidelines on the finally rendered result (after page finished its loading and all scripts have been applied).

Thirdly, a tool for automated usability evaluation called the *Guideliner* was designed. The *Guideliner* evaluates WUI conformance to usability guidelines during implementation phase of WUI development. It addresses the problem of how to effectively evaluate web user interfaces usability during their development, in particular within the implementation phase with minimal cost and time. The *Guideliner* evaluates finally rendered WUI assessing HTML-based usability guidelines and guidelines covering visual characteristics of WUI, and is applicable both at the final established WUI evaluation phase (traditional approach) and for evaluating WUI conformance to set usability criteria during WUI implementation phase (novel method introduced in the thesis).

The methods introduced in the thesis can be exploited for any HTML, JavaScript, and CSS based web user interface without restrictions. Usability evaluations conducted on various local and international web applications have proved that proposed methods are efficient and complete enough to evaluate usability of majority of web user interfaces.

Lühikokkuvõte

Veebi kasutajaliidese kasutatavuse automaatne hindamine realisatsioonifaasis

Erinevate nutiseadmete ja veebiplatvormide rohkus tänapäeval on toonud endaga kaasa mitmeid muutusi veebirakendustes ja nende loomises. Üha enam pööratakse tähelepanu ja pannakse arenduse käigus rõhku kasutajate erinevatele vajadustele, et tagada kasutajate rahulolu, ja pikas perspektiivis tagada veebirakenduse edu ja kestus. Kasutatavuse reeglite jälgimine veebi kasutajaliideste disainimisel on üks olulisematest edu võtmetest ja ka nõuetest kasutajaliidestele. Samas on kasutatavuse testimine kogu arendusprotsessi jooksul küllalt kulukas. Osa kasutatavuse testimisest, mida tehakse inimese poolt on aga võimalik teostada automaatselt arvutiprogrammide abiga erinevates tarkvara arendusfaasides. Kasutatavuse automaatse kontrollimise eelis on tema odavus, ja kui seda läbi viia tarkvara arendusfaasis, võimaldab see täiendavalt anda otsest tagasisidet arendajatele leitud kasutatavuse probleemide kohta kohe peale kasutajaliidese koodi loomist või muutmist. Selline lähenemine omakorda vähendab vea ülesleidmise aega ja kokkuvõttes ka kasutatavuse vigade parandamise maksumust.

Käesolev doktoritöö keskendub veebi kasutajaliideste kasutatavuse automaatse testimise võimalustele ja probleemidele, sealhulgas uurides kasutatavuse automaatse hindamise võimalikkust tarkvara arendusfaasis. Teiseks uudsuseks olemasolevate vahendite kõrval on kasutatavuse visuaalsete aspektide automaatne hindamine, mis on pikalt olnud antud valdkonnas probleemiks puudulike tarkvaralahenduste tõttu.

Töö käigus on loodud raamistik, mis muudab oluliselt kasutajaliidese arendusprotsessi, kuna kasutajaliideste automaatne valideerimine kasutatavuse reeglite suhtes toimub kasutajaliideste varajases arendusetapis. Selline parandus lubab kasutatavuse vigu tuvastada ja parandada varakult tarkvara loomise käigus, mis omakorda vähendab paranduse kulukust võrreldes vea avastamisega mõnes hilisemas arendusetapis, näiteks lõpptestimisel.

Teiseks tähtsaks saavutuseks on kasutatavuse reeglite esitamise viis, kasutades selleks ontoloogiat, mis võimaldab kasutatavuse reegleid esitleda nii inimloetavalt kui ka masintöödeldaval kujul. Töö käigus on loodud ontoloogia disain ja ka kasutatavuse ontoloogia automaatselt hinnatavatele reeglitele.

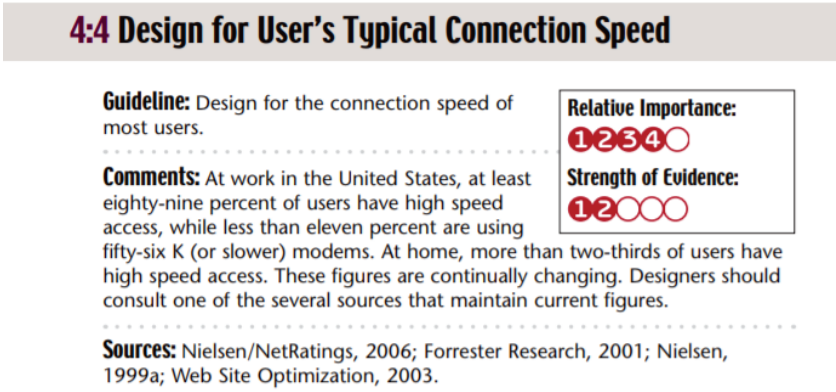
Kolmandaks saavutuseks loodud raamistiku juures on visuaalsete karakteristikute hindamise meetod. Meetodi uudsus seisneb selles, et hindamisprogramm suudab hinnata kasutajaliidese vastavust kasutatavuse reeglitele lõppkasutajale esitatava kuva suhtes, st. kui kõik stiililaadid ja skriptid lehe kujundamiseks on oma töö lõpetanud. Lisaks HTML koodi spetsiifilistele reeglitele oskab loodud hindamisprogramm Guideliner hinnata kasutajaliidese visuaalseid omadusi nagu näiteks elemendi positsioon lehel, kaugus erinevate elementide vahel, lehekülje kerimise olemasolu aga ka elementide kontrastsust tausta suhtes.

Doktoritöö käigus loodud raamistikku saab kasutada erinevatel veebitehnoloogiatel nagu HTML, JavaScript ja CSS põhinevatel kasutajaliidestel. Läbiviidud testid tõestasid, et välja töötatud metoodika ja arendatud hindamisvahendi Guideliner abil on edukalt võimalik hinnata veebilehe vastavust kasutatavuse reeglitele ja tuvastada kasutatavuse probleeme.

Appendixes

Appendix 1 Example of Usability Guideline Structure

Example 1.
Example of usability guideline definition and presentation derived from Research-Based Web Design & Usability Guidelines presented in [78].



4:4 Design for User's Typical Connection Speed

Guideline: Design for the connection speed of most users.

Relative Importance: 1 2 3 4 ○

Strength of Evidence: 1 2 ○ ○ ○

Comments: At work in the United States, at least eighty-nine percent of users have high speed access, while less than eleven percent are using fifty-six K (or slower) modems. At home, more than two-thirds of users have high speed access. These figures are continually changing. Designers should consult one of the several sources that maintain current figures.

Sources: Nielsen/NetRatings, 2006; Forrester Research, 2001; Nielsen, 1999a; Web Site Optimization, 2003.

Example 2.
Example of usability guideline presented by Jakob Nielsen in Top 10 Guidelines for Homepage Usability [1]

Make the Site's Purpose Clear: Explain Who You Are and What You Do

1. Include a one-sentence tagline

Start the page with a tagline that summarizes what the site or company does, especially if you're new or less than famous. Even well-known companies presumably hope to attract new customers and should tell first-time visitors about the site's purpose. It is especially important to have a good tagline if your company's general marketing slogan is bland and fails to tell users what they'll gain from visiting the site.

2. Write a window title with good visibility in search engines and bookmark lists

Begin the TITLE tag with the company name, followed by a brief description of the site. Don't start with words like "The" or "Welcome to" unless you want to be alphabetized under "T" or "W."

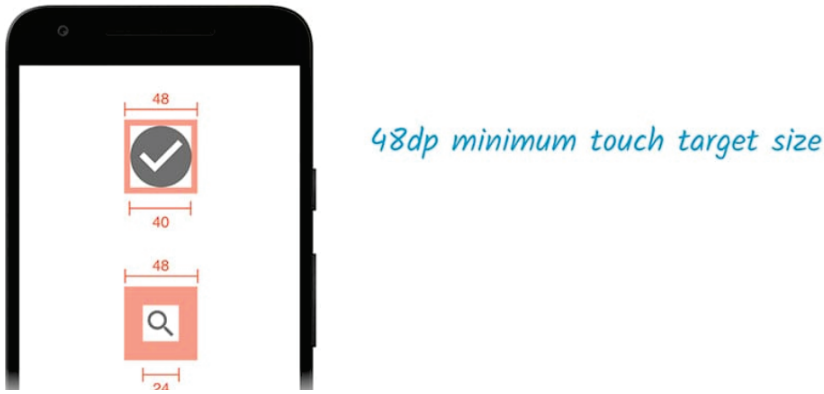
3. Group all corporate information in one distinct area

Finding out about the company is rarely a user's first task, but sometimes people do need details about who you are. Good corporate information is especially important if the site hopes to support recruiting, investor relations, or PR, but it can also serve to increase a new or lesser-known company's credibility. An "**About <company-name>**" section is the best way to link users to more in-depth information than can be presented on the homepage. (See also my report on usability guidelines for the design of "about us" areas of corporate websites.)

[1] https://www.nngroup.com/articles/top-ten-guidelines-for-homepage-usability/

Example 3.
Example of usability guidelines presented by Google[1]

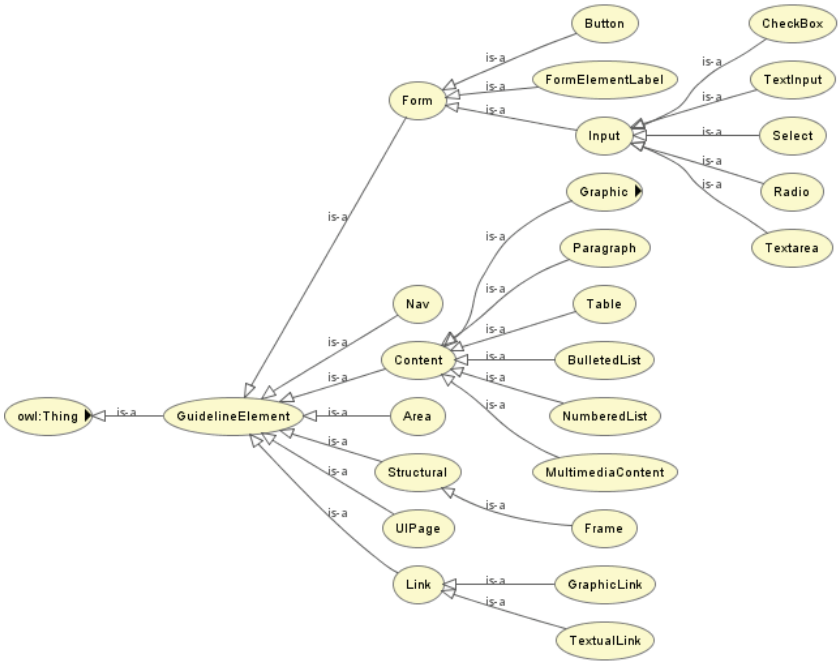
A minimum recommended touch target size is around 48 device independent pixels on a site with a properly set mobile viewport. For example, while an icon may only have a width and height of 24px, you can use additional padding to bring the tap target size up to 48px. The 48x48 pixel area corresponds to around 9mm, which is about the size of a person's finger pad area.



[1] https://developers.google.com/web/fundamentals/accessibility/accessible-styles

Appendix 2 Main Concepts of the Usability Ontology

Outline of established usability ontology describing the *GuidelineElement* concept and its descendant classes: excerpt from the Protégé ontology editor



Appendix 3 Usability Guidelines Used for the Research

To compose the list of usability guidelines for the research, the author analysed recommendations in scientific publications [29] [111], WCAG [103] and Section 508 guidelines[1], Research-Based Web Design [106] and Usability Guidelines from U.S. Dept. of Health and Human Services [78], and supplemented with recommendations from the Nielsen Norman Group [8] [31] [65]. Moreover, evidence-based user experiences and usability research works have been inspected [30] [31]. The appendix presents examples of usability guidelines that can be evaluated automatically grouped by the category (mobile usability guidelines, common usability and common accessibility guidelines).

Excerpt from mobile usability guidelines (4 of 20 guidelines are presented)

Identifier:	1
Guideline:	A label should be placed above the input field
Description:	Left-aligned or right-aligned labels decrease usability on mobile devices as small screen size leave very little space for the input field. Label located above the input allows to make input field screen-wide
Condition:	Check that label is located above the input
Platform:	Mobile
Category:	Form
Source:	Google HCI
Relative Importance:	4
Strength of Evidence:	4

Identifier:	2
Guideline:	A scroll is one directional
Description:	Ensure that the scrolling is one directional. It should be one directional in both landscape and portraits screen orientation. Mixing two scrolling on the same page increases the complexity and disorientation decreasing the overall satisfaction with mobile web application.
Condition:	Check that only one scroll horizontal or vertical is available at the same time
Platform:	Mobile
Category:	UI Page
Source:	Mobile Usability Research
Relative Importance:	5
Strength of Evidence:	5

Identifier:	19
Guideline:	All links should be 48 CSS pixels wide
Description:	The average size of finger pad is approximately 10 millimetres for adults. The minimal recommended size of tap target is about seven millimetres that are roughly equal to 48 CSS pixels.
Condition:	Check that links width is not less than 48px
Platform:	Mobile
Category:	Link

[1] https://www.section508.gov

Source:	Google HCI, Android HCI
Relative Importance:	4
Strength of Evidence:	4

Identifier:	20
Guideline:	Radio buttons should be vertically-stacked
Description:	Due to the limited screen size on mobile devices, it could be complicated to fit radio buttons vertically. Also, vertically-stacker radio button could be processed faster by users then located horizontally
Condition:	Check that there are no radio buttons on the same line
Platform:	Mobile
Category:	Radio Button
Source:	Venture Harbour HCI
Relative Importance:	5
Strength of Evidence:	4

Excerpt from common accessibility guidelines (4 of 55 guidelines are presented)

Identifier:	21
Guideline:	Language should be defined in HTML code
Description:	Primary language of the web page should be defined in HTML code. When language is defined assistive technologies can process it more precisely
Condition:	Lang attribute of HTML tag should exist and should not be empty
Platform:	All
Category:	Accessibility
Source:	WCAG 2.0
Relative Importance:	5
Strength of Evidence:	5

Identifier:	22
Guideline:	Text equivalents is defined for every image on the screen
Description:	All images have alternative text defined. People with visual or certain cognitive disabilities cause screen readers that process alternative text for describing the content of the image.
Condition:	Each img tag has an attribute alt defined
Platform:	All
Category:	Accessibility
Source:	WCAG 2.0
Relative Importance:	5
Strength of Evidence:	5

Identifier:	72
Guideline:	Link alternative text should be different from the link text itself
Description:	When alt attribute of link has the same value as a link text then screen readers read out the same text link text twice
Condition:	Check that link text and link alt attribute have different value
Platform:	All

Category:	Accessibility
Source:	WCAG 2.0
Relative Importance:	4
Strength of Evidence:	4

Identifier:	73
Guideline:	Every page has title attribute
Description:	The purpose of title attribute is to identify the location without the need to identify the whole content of the page
Condition:	Check that title tag is defined and is not empty
Platform:	All
Category:	Accessibility
Source:	WCAG 2.0
Relative Importance:	5
Strength of Evidence:	4

Excerpt from common usability guidelines (4 of 23 guidelines are presented)

Identifier:	74
Guideline:	Select lists should contain more than 7 options
Description:	In order to see results of select list, the user should click on the select list. So, in case there are less than 7 options available it is more beneficial to use radio buttons, so users immediately scan how many options they have.
Condition:	Check that select list has more than 7 option
Platform:	Desktop/Mobile
Category:	Select List
Source:	Baymard Institute UX Research
Relative Importance:	4
Strength of Evidence:	3

Identifier:	75
Guideline:	Radio button label should be clickable
Description:	Radio button target area should be enlarged with a label or words associated with it.
Condition:	Check that when radio button label is clicked, radio button becomes selected
Platform:	Desktop/Mobile
Category:	UI Page
Source:	UX Planet Research
Relative Importance:	4
Strength of Evidence:	4

Identifier:	97
Guideline:	Buttons background colour should have proper contrast rate
Description:	Buttons should not blend in with its surrounding content as, in common, they are used to perform certain actions and should be bright enough in comparison with surrounding content
Condition:	Check that the contrast rate of button and the content surrounding is more than 4:51
Platform:	Desktop/Mobile

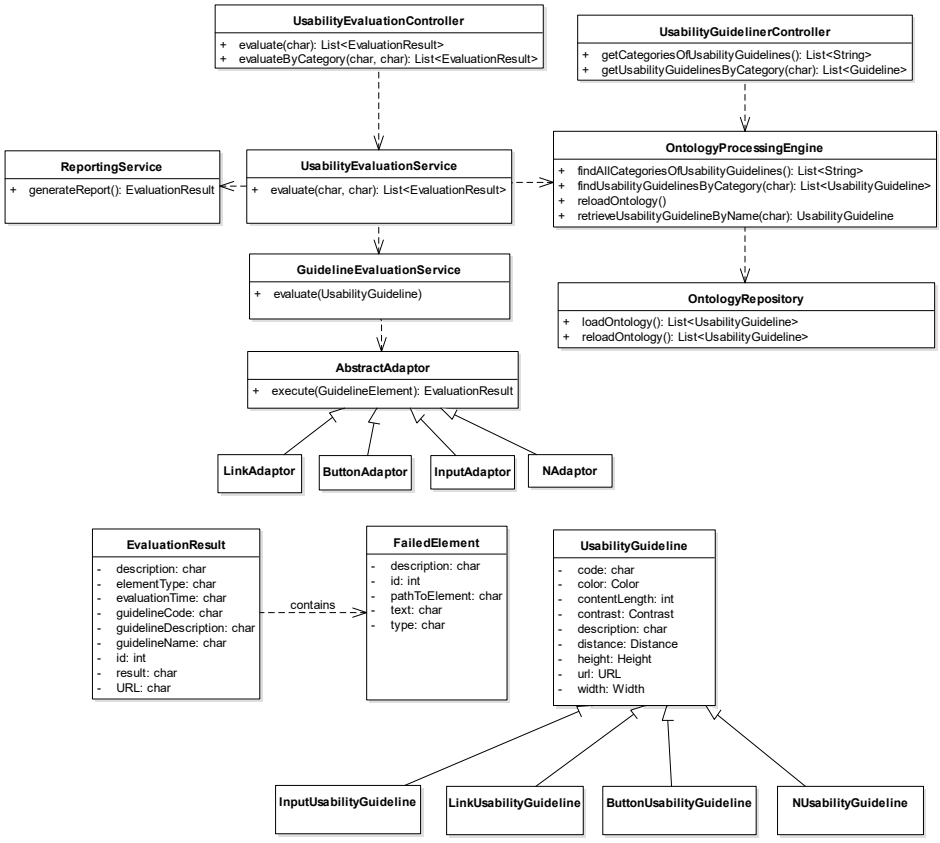
Category:	Button
Source:	Prototype usability research
Relative Importance:	4
Strength of Evidence:	4

Identifier:	98
Guideline:	Web UI response time should not be more that 3 seconds
Description:	Usability research show that around 40% of users abandon web site that takes more than 3 seconds to load.
Condition:	Check that with connection speed of 50Mbps web application load time is not more than 3 seconds
Platform:	Desktop/Mobile
Category:	UI Page
Source:	Nielsen Norman Group Research
Relative Importance:	4
Strength of Evidence:	4

Appendix 4 Guideliner Software Architecture

Class Diagram of the Guideliner

The class diagram shows the abstraction of the *Guideliner* architecture. It contains main classes of the *Guideliner Core* including Reporting Component service (*ReportingService*), Ontology Processing Engine services (*OntologyProcessingEngine*, *OntologyRepository*) and WUI Evaluation Component services (*UsabilityEvaluationService*, *GuidelineEvaluationService*, *AbstractAdapter* and its child classes). Also the domain objects: *UsabilityGuideline* and *EvaluationResult* are presented. The purpose of the class diagram is to show how the primary concepts of the *Guideliner* on the code level.



Appendix 5 Usability Evaluation Report Based on EARL

This appendix presents an example of automated usability evaluation report based on EARL. The EARL report (generated by the *Guideliner*) shows that WUI (https://www.etis.ee/Portal/Projects/Index?searchType=detailed) conformance was evaluated to two usability guidelines: 24-IdentifySelectedInput (evaluation passed) and 27-RadioButtonsShouldBeVerticallyStacked (evaluation failed).

```
<rdf:RDF xmlns:earl="http://www.w3.org/WAI/ER/EARL/nmg-strawman#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:dct="http://purl.org/dc/terms/"
xmlns:foaf="http://xmlns.com/foaf/0.1/">
    <earl:TestSubject rdf:ID="evaluatedWebPages">
        <dct:title xml:lang="en">Web Page being evaluated</dct:title>
        <dct:hasPart
rdf:resource="https://www.etis.ee/Portal/Projects/Index?searchType=detailed"/>
    </earl:TestSubject>
    <earl:Software rdf:about="http://validator.w3.org/about.html#"
rdf:ID="automatedUsabilityAssertor">
        <dct:title xml:lang="en">Automated Web Usability Evaluation Tool</dct:title>
        <dct:hasVersion>1.9.1</dct:hasVersion>
        <dct:description xml:lang="en">A tool for automated usability evaluation checking
the conformance of web UI to the predefined set of guidelines
        </dct:description>
    </earl:Software>
    <earl:Assertion rdf:ID="assertion1">
        <earl:assertedBy rdf:resource="#automatedUsabilityAssertor"/>
        <earl:subject rdf:resource="#evaluatedWebPages"/>
        <earl:testcase rdf:resource="#24-IdentifySelectedInputTestCase"/>
        <earl:result rdf:resource="#24-IdentifySelectedInputTestResult"/>
    </earl:Assertion>
    <earl:Assertion rdf:ID="assertion2">
        <earl:assertedBy rdf:resource="#automatedUsabilityAssertor"/>
        <earl:subject rdf:resource="#evaluatedWebPages"/>
        <earl:testcase rdf:resource="#27-RadioButtonsShouldBeVerticallyStackedTestCase"/>
        <earl:result rdf:resource="#27-RadioButtonsShouldBeVerticallyStackedTestResult"/>
    </earl:Assertion>
    <earl:TestCase rdf:about="http://www.w3.org/TR/WCAG20-TECHS/H36" rdf:ID="24-
01_IdentifySelectedInputTestCase">
        <dct:title xml:lang="en">(24-IdentifySelectedInput) Identify selected
inputs</dct:title>
        <dct:description xml:lang="en">User should be easily able to identify what has been
selected to make the experience better. Show the selected link by highlighting it with
different colour or something similar or viable
        </dct:description>
    </earl:TestCase>
    <earl:TestResult rdf:ID="24-IdentifySelectedInputTestResult">
        <earl:outcome rdf:resource="http://www.w3.org/WAI/ER/EARL/nmg-strawman#pass"/>
    </earl:TestResult>
    <earl:TestCase rdf:about="http://www.w3.org/TR/WCAG20-TECHS/H36" rdf:ID="27-
01_RadioButtonsShouldBeVerticallyStackedTestCase">
        <dct:title xml:lang="en">(27-RadioButtonsShouldBeVerticallyStacked) Radio buttons
should be vertically-stacked</dct:title>
        <dct:description xml:lang="en">Vertically-stacking radio buttons
            (and checkboxes) makes them faster to process compared to a horizontal layout.
        </dct:description>
    </earl:TestCase>
    <earl:TestResult rdf:ID="27-RadioButtonsShouldBeVerticallyStackedTestResult">
        <earl:outcome
                rdf:resource="http://www.w3.org/WAI/ER/EARL/nmg-strawman#fail" />
        <dc:description rdf:parseType="Literal">
            <div xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">
Radio button with labels Beginning, Fuzzy and Precise and vertically stacked.
            </div>
        </dc:description>
    </earl:TestResult>
</rdf:RDF>
```

Appendix 6 WUIs Used for Guideliner Verification

The appendix lists the websites and particular pages that were used for the *Guideliner* verification. The following notation is used below to identify start page (S), subpage (P), and a webpage with a form (F) for the websites involved in the study.

World universities (10)

1. Massachusetts Institute of Technology
S: http://web.mit.edu/
P: http://gradadmissions.mit.edu/
F: http://student.mit.edu/catalog/archive/fall/extsearch.cgi
2. Stanford University
S: https://www.stanford.edu/
P: https://www.stanford.edu/academics/
F: https://www.gsb.stanford.edu/faculty-research/centers-initiatives/ces/research/search-funds/primer
3. Harvard University
S: https://www.harvard.edu/
P: https://www.harvard.edu/students
F: https://connects.catalyst.harvard.edu/profiles/search/people
4. University of California
S: http://www.caltech.edu/
P: http://www.caltech.edu/content/undergrad-education
F: https://directory.caltech.edu/search/advanced_search
5. University of Cambridge
S: https://www.cam.ac.uk/
P: https://www.undergraduate.study.cam.ac.uk/applying
F: https://www.student-funding.cam.ac.uk
6. University of Oxford
S: http://www.ox.ac.uk/
P: https://www.ox.ac.uk/research/divisions?wssl=1
F: https://www.ox.ac.uk/funnelback/search?wssl=1
7. London's Global University
S: https://www.ucl.ac.uk/
P: http://www.ucl.ac.uk/research/excellence/funding
F: https://ucl-primo.hosted.exlibrisgroup.com/primo72explore/search
8. Imperial College London
S: https://www.imperial.ac.uk/
P: http://www.imperial.ac.uk/clinical-trials-unit/collaborations/
F: http://www.imperial.ac.uk/collegedirectory/
9. University of Chicago
S: https://www.uchicago.edu/
P: https://www.uchicago.edu/academics/
F: https://directory.uchicago.edu/
10. ETH Zurich – Swiss Federal Institute of Technology Zurich
S: https://www.ethz.ch/en.html
P: https://www.ethz.ch/en/doctorate/registration-admission.html
F: https://wohnen.ethz.ch/index.php?act=searchoffer

Europe universities (10)

1. University of Cambridge
S: https://www.cam.ac.uk/
P: https://www.undergraduate.study.cam.ac.uk/applying

F: https://www.student-funding.cam.ac.uk

2. University of Oxford

S: http://www.ox.ac.uk/

P: https://www.ox.ac.uk/research/divisions?wssl=1

F: https://www.ox.ac.uk/funnelback/search?wssl=1

3. London's Global Universities

S: https://www.ucl.ac.uk/

P: http://www.ucl.ac.uk/research/excellence/funding

F: https://ucl-primo.hosted.exlibrisgroup.com/primo-explore/search?vid=UCL_VU2&mode=advanced&sortby=rank

4. Imperial College London

S: https://www.imperial.ac.uk/

P: http://www.imperial.ac.uk/clinical-trials-unit/collaborations/

F: http://www.imperial.ac.uk/collegedirectory/

5. ETH Zurich – Swiss Federal Institute of Technology Zurich

S: https://www.ethz.ch/en.html

P: https://www.ethz.ch/en/doctorate/registration-admission.html

F: https://wohnen.ethz.ch/index.php?act=searchoffer

6. Technical University of Lausanne

S: https://www.epfl.ch/

P: https://studying.epfl.ch/student_desk

F: https://infoscience.epfl.ch/search?ln=en&p=test&f=title&ext=collection%3AARTICLE

7. University of Edinburgh

S: https://www.ed.ac.uk/

P: https://www.ed.ac.uk/staff

F: https://www.ed.ac.uk/student-funding/search-cholarships?field_study_level_tid=1&term_node_tid_depth=All&term_node_tid_depth_1=All

8. King's College London

S: https://www.kcl.ac.uk/

P: https://www.kcl.ac.uk/innovation/research/index.aspx

F: https://www.kcl.ac.uk/study/Search-results.aspx

9. The London School of Economics and Political Science

S: http://www.lse.ac.uk/

P: http://www.lse.ac.uk/accounting/Home.aspx

F: http://www.lse.ac.uk/student-life/accommodation/search-accommodation?from_serp=1

10. Paris University École normale supérieure (ENS)

S: http://www.ens.fr/

P: http://www.ens.fr/en/academics/admissions

F: http://www.ens.fr/en/les-laboratoires-sciences

Estonian universities (6)

1. Tallinn University of Technology

S: https://www.ttu.ee/

P: https://www.ttu.ee/teaduskond/infotehnoloogia-teaduskond/doktoriope-33/

F: https://www.ttu.ee/?id=30052

2. Tallinn University

S: https://www.tlu.ee/en

P: https://www.tlu.ee/en/research/Scholars

F: http://www.tlu.ee/en/Conference-Centre/Inquiry-for-organising-an-event

3. Estonian Academy of Arts

S: https://www.artun.ee/en/admissions/welcome

P: https://www.artun.ee/en/studies/

F: https://www.artun.ee/en/oppimine/kalender/

4. Estonian University of Life Sciences

S: https://www.emu.ee/en/
P: http://pk.emu.ee/en/
F: https://www.emu.ee/about-the-university/events/kalender/2017-11

5. Tartu University

S: https://www.ut.ee/et
P: https://www.ut.ee/en/research
F: https://elurikkus.ut.ee/search_er2.php?lang=eng

6. Estonian Academy of Music and Theatre

S: http://www.ema.edu.ee/en/
P: http://www.ema.edu.ee/en/continuing-education/organisation-of-courses/
F: http://www.ema.edu.ee/en/studies/curricula/courses-subject-catalogue/

Appendix 7 Comparison of the Benchmark Tools used for the Guideliner Verification

Characteristic	Wave	Powermapper	AChecker	Tenon	DynoMapper	TotalValidator	Guideliner
HTML-specific guidelines	WCAG 2.0 Section 508 HTML ARIA CSS	WCAG 2.0 Section 508 HTML/HTML5 Usability.gov HTML syntax	WCAG 1.0 WCAG 2.0 Section 508 Stanca Act BITV 1.0	WCAG 1.0 WCAG 2.0 Section 508 Stanca Act BITV 1.0	WCAG 2.0 Section 508	WCAG 2.0 Section 508 HTML CSS XHTML	WCAG 2.0
Visual usability guidelines	Contrast	Compatibility SEO	Not supported	Not supported	Not supported	Contrast	Scrolling Position Location Contrast Distance Size
Number of guidelines	80	150	152	74	89	115	98
Environment (Web, desktop, other)	Web Browser Extension	Web Desktop	Web	Web	Web	Web Browser Extension	Web Command line
Ways of defining custom guidelines (if available)	None	None	Table-based approach	None	None	None	Ontology
Reporting interface	HTML REST	HTML REST (HTML, PNG, CSV, XML)	HTML REST EARL	HTML REST	HTML REST	HTML REST	REST EARL
Commercial/Free ware	Web is free API is commercial	Commercial (with 30 days trial)	Freeware	Commercial (with 15 days trial)	Commercial	Freeware (Basic) Commercial (Pro)	Freeware

Appendix 8 Comparison of Features of Automated Usability Evaluation Tools

The matrix of associations of the tools is shown where “1” means that the tool supports characteristic and “0” means that the tool does not support the characteristic. The metrics presents the important characteristics that should be satisfied by the tools for automated usability evaluation to be competitive.

Characteristic	Wave	Power Mapper	AChecker	Tenon	Dyno Mapper	Total Validator	Guideliner
API (e.g. REST, SOAP)	1	1	1	1	1	1	1
User Interface for Evaluation	1	1	1	1	1	1	1
Online Availability	1	1	1	1	1	1	0
Accessibility Guidelines	1	1	1	1	1	1	1
Usability Guidelines	0	1	0	0	0	0	1
Visual usability guidelines	0	0	0	0	0	0	1
Extendibility with new guidelines	0	0	1	0	0	0	1
License type commercial(0)/free(1)	1	0	1	0	0	1	1

Appendix 9 Results of Usability Evaluation

Table 1. Total sum of detected violations by tools for automated usability evaluation
The table shows the result of the *Guideliner* verification: total number of detected violations for each tool grouped by the type of usability guidelines. The results demonstrate that the *Guideliner* in additional to accessibility guidelines detected similar number of violations of mobile usability guidelines and smaller number of violations of common usability guidelines.

	Accessibility	Compatibility	SEO	Usability	Mobile usability	Failed tests
Wave	3306	–	–	–	–	1
Powermapper	1365	48	166	893	–	4
Achecker	2355	–	–	–	–	2
Tenon	2309	–	–	–	–	6
DynoMapper	2386	–	–	–	–	1
TotalValidator	3742	–	–	–	–	3
Guideliner	3457	–	–	1145	3860	0

Table 2. Number of violations grouped by university WUI and the tool
The table shows total number of violations detected on each university WUI by the tools. The table shows raw data that was captured during the verification of the *Guideliner.*

Nr.	University	Tools						
		Wave	PowerMapper	Achecker	DynoMapper	Tenon	TotalValidator	Guideliner
1	London's Global University	53	84	25	65	22	20	185
2	University of Edinburgh	49	36	50	60	14	114	239
3	University of California	140	126	79	75	35	69	299
4	Stanford University	66	29	49	58	35	62	390
5	Imperial College London	117	92	62	59	87	53	371
6	King's College London	241	62	83	80	173	90	551
7	University of Chicago	204	91	81	112	43	28	415
8	University of Oxford	83	193	62	63	45	70	517
9	Tartu University	157	176	59	0	105	25	515
10	Tallinn University	170	155	142	124	135	89	335
11	Estonian Academy of Arts	32	56	159	146	105	167	219
12	The London School of Economics and Political Science	270	220	47	159	108	64	404
13	Estonian Academy of Music and Theatre	248	112	187	264	53	40	216
14	Massachusetts Institute of Technology	141	30	170	151	83	234	341
15	University of Cambridge	397	135	98	96	288	281	429

Nr.	University	Tools						
		Wave	PowerMapper	Achecker	DynoMapper	Tenon	TotalValidator	Guideliner
16	Harvard University	137	146	107	120	93	141	638
17	Paris University École normale supérieure (ENS)	108	47	40	143	47	1501	602
18	Technical University of Lausanne	213	184	73	116	73	85	402
19	ETH Zurich – Swiss Federal Institute of Technology Zurich	146	169	101	116	160	169	304
20	Estonian University of Life Sciences	119	96	437	142	258	161	449
21	Tallinn University of Technology	215	233	244	237	347	279	641
	Total	3306	2472	2355	2386	2309	3742	8462

Table 3. Minimum, Maximum and Average Violation Coverages

The table demonstrates the violation coverage for the 6 benchmark tools grouped by the university WUI. Also, the table compares the results of the *Guideliner* with the benchmark tools. The result clearly outline that the *Guideliner* minimum coverage is higher for all WUI, while the maximum coverage is defined as 100% in most cases meaning that the *Guideliner* showed the highest maximum coverage between all tools. The average *Guideliner* coverage results are also not homogeneous with average results of all tools exceeding the average coverage results between all tools multiple times.

Nr.	University	Violation coverage for the 6 benchmark tools			Guideliner			
		Min	Max	Avg	Violation coverage	Min %	Max %	Average %
1	London's Global University	0.01	0.11	0.04±0.03	0.11	1151.7%	100 %	158.74
2	The university of Edinburgh	0.01	0.17	0.05±0.05	0.17	1133.33%	100%	238.12
3	California Institute of Technology (Caltech)	0.02	0.15	0.06±0.05	0.15	582.45%	100%	158.68
4	Stanford University	0.02	0.2	0.05±0.07	0.20	1207.29%	100%	274.19
5	Imperial College London	0.02	0.16	0.06±0.05	0.16	791.84%	100%	161.92

Nr.	University	Violation coverage for the 6 benchmark tools			Guideliner			
		Min	Max	Avg	Violation coverage	Min %	Max %	Average %
6	Kings College London	0.02	0.21	0.09±0.07	0.21	760.11%	100%	129.87
7	University of Chicago	0.01	0.16	0.06±0.05	0.16	1308.16%	100%	153.99
8	University of Oxford	0.02	0.17	0.07±0.06	0.17	888.78%	100%	162.14
9	University of Tartu	0.10	0.19	0.07±0.07	0.19	93.88%	100%	190.20
10	Tallinn University	0.03	0.17	0.09±0.05	0.17	422.73%	100%	88.10
11	Estonian Academy of Arts	0.02	0.15	0.07±0.05	0.15	618.37%	100%	119.81
12	London School of Economics and Political Science	0.02	0.17	0.09±0.06	0.17	1030.03%	100%	90.81
13	Estonian Academy of Music and Theatre	0.02	0.20	0.08±0.08	0.17	897.45%	14.23%	117.77
14	Massachusetts Institute of Technology	0.02	0.18	0.08±0.05	0.18	726.53%	100%	127.76
15	University of Cambridge	0.03	0.28	0.11±0.09	0.19	623.81%	32.77%	67.17
16	Harvard University	0.03	0.20	0.09±0.06	0.2	591.99%	100%	125.90
17	Paris University École normale supérieure (ENS)	0.01	0.29	0.11±0.11	0.21	1502.72%	27.24%	96.04
18	Technical University of Lausanne	0.02	0.14	0.08±0.05	0.14	606.58%	100%	79.86
19	ETH Zurich – Swiss Federal Institute of Technology Zurich	0.03	0.16	0.08±0.04	0.16	448.82%	100%	92.59
20	Estonian University of Life Sciences	0.06	0.19	0.1±0.05	0.19	235.56%	100%	90.31

Nr.	University	Violation coverage for the 6 benchmark tools			Guideliner			
		Min	Max	Avg	Violation coverage	Min %	Max %	Average %
21	Tallinn University of Technology	0.06	0.25	0.15±0.07	0.25	309.91%	100%	71.00

Curriculum vitae

Personal data

Name:	Jevgeni Marenkov
Date of birth:	09.07.1988
Place of birth:	Estonia
Citizenship:	Estonian

Contact data

E-mail:	jevgeni.marenkov@ttu.ee

Education

2013 – …	PhD in Information and Communication Technology, Tallinn University of Technology
2010 – 2012	MSC in Computer Engineering, Tallinn University of Technology (*cum laude*)
2007 – 2010	BSC in Computer Engineering, Tallinn University of Technology (*cum laude*)
2004 – 2007	Tallinna Kesklinna Vene Gümnaasium

Language competence

English	Fluent
Estonian	Fluent
Russian	Native

Professional employment

2014 – …	Kühne + Nagel IT Service Centre AS, Software Engineer
2013 – 2014	Helmes AS, Software Engineer

Scientific work

Reviewer	East-European Conference on Advances in Databases and Information Systems (ADBIS 2017, ADBIS 2016)
Reviewer	12th International Baltic Conference on Databases and Information Systems (Baltic DB&IS'2016)

Defended theses

2012	Master of Science in Computer Engineering, Implementation of Business Rules using Aspect Oriented Programming. TUT, supervisor: lector Raul Liivrand
2010	Bachelor of Business Information Systems, UML and SysML comparison based on the information system of security company. TUT, assoc. professor Erki Eessaar

Main Areas of Scientific Work

Web user interfaces, web usability evaluations, automated usability evaluation, web usability guidelines, and software development

Elulookirjeldus

Isikuandmed

Nimi:	Jevgeni Marenkov
Sünniaeg:	09.07.1988
Sünnikoht:	Eesti
Kodakondsus:	Eesti

Kontaktandmed

E-post:	jevgeni.marenkov@ttu.ee

Hariduskäik

2013 – ...	Doktorantuur, Info- ja kommunikatsioonitehnoloogia, Tallinna Tehnikaülikool
2010 – 2012	Tehnikateaduste magister (M.Sc.), Tallinna Tehnikaülikool (*cum laude*)
2007 – 2010	Tehnikateaduste bakalaureus (B.Sc.), Tallinna Tehnikaülikool (*cum laude*)
2004 – 2007	Keskharidus, Tallinna Kesklinna Vene Gümnaasium

Keelteoskus

Inglise keel	Kõrgtase
Eesti keel	Kõrgtase
Vene keel	Emakeel

Teenistuskäik

2014 – ...	Kühne + Nagel AS, Tarkvara arendaja
2013 – 2014	Helmes AS, Tarkvara arendaja

Teadustegevus

Retsenseerimine	East-European Conference on Advances in Databases and Information Systems (ADBIS 2017, ADBIS 2016)
Retsenseerimine	12th International Baltic Conference on Databases and Information Systems (Baltic DB&IS'2016)

Kaitstud lõputööd

2012	Magistritöö, Ärireeglite Realiseerimine Aspekt Orienteeritud Programmeerimise Abil: lektor Raul Liivrand
2010	Bachelor Business Information Systems, UML ja SysML võrdlemine turvafirma infosüsteemi baasil, dotsent Erki Eessaar

Teadustöö põhisuunad

Veebi kasutajaliides, veebi kasutatavuse hindamine, automatiseeritud kasutatavuse hindamine, veebi kasutatavuse suunised ja tarkvara arendus