Complex Systems Informatics and Modeling Quarterly (CSIMQ)
eISSN: 2255-9922
Published online by RTU Press, https://csimq-journals.rtu.lv
Article 90, Issue 15, June/July 2018, Pages 90–109
https://doi.org/10.7250/csimq.2018-15.05

Design-Time Web Usability Evaluation with Guideliner

Jevgeni Marenkov*, Tarmo Robal, and Ahto Kalja

Tallinn University of Technology, Tallinn, Estonia

jevgeni.marenkov@gmail.com, tarmo.robal@ati.ttu.ee, ahto.kalja@ttu.ee

Abstract. The diversity of smartphones and tablet computers has become an integral part of modern life. An essential requirement for web application development is following web usability guidelines, while designing web user interface (UI). Even a minor change in UI could lead to usability problems. Empirical evaluation methods like interviews and questionnaires with user-tests and card sorting are effective in finding such problems. Nevertheless, there are multiple obstacles preventing the application of these methods especially for evaluating minor UI changes, for instance, due to the time and human-resources they require, and the amount of data to be processed. The purpose of this current publication is to present Guideliner – a tool for implementation-time automatic evaluation of web UI conformance to predefined usability guidelines. The main contribution of the presented solution is enabling immediate cost-efficient and automated web UI evaluation that conforms to available and set standards.
Keywords: Web usability, Usability guidelines, Web user interface.

1 Introduction

The diversity of computing platforms – computers, laptops, smartphones, tablet computers and smartwatches – has become an intrinsic part of modern life and culture. Therefore, web user interface (UI) compatibility with different platforms, e.g. mobile devices, is an essential requirement for each web application. Furthermore, UIs should also be compatible with the diversity of software platforms (including Android, iOS, Windows, Linux, etc.) and different browsers (Safari, Chrome, Firefox, etc.) regardless of their version. Device and platform compatibility covers only a minor part of requirements set for UIs. In fact, UIs of web applications should be consistent between pages, attractive, user-friendly, easy to use and navigate. All such characteristics are included in the definition of usability. Usability is 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 [1]. Usability covers many

* Corresponding author

Reference: J. Marenkov, T. Robal, and A. Kalja, "Design-Time Web Usability Evaluation with Guideliner," Complex Systems Informatics and Modeling Quarterly, CSIMQ, no. 15, pp. 90–109, 2018. Available: https://doi.org/10.7250/csimq.2018-15.05

Additional information. Authors ORCID iD: J. Marenkov – orcid.org/0000-0001-8764-1465, T. Robal – orcid.org/0000-0002-7396-8843, and A. Kalja – orcid.org/0000-0001-6416-0503. PII S225599221800090X. Received: 11 February 2018. Accepted: 29 June 2018. Available online: 31 July 2018.

areas such as accessibility: referring to UI requirements for people experiencing disabilities, and learnability: assuring that web application functionality is complete and correctly displayed.

Demand for usable web applications has led to a variety of approaches and methods which help to achieve a high level of usability. Many studies [2], [3], [4], [5] provide usability guidelines, best design practices, recommendations and patterns to follow when designing web UI. Usability guidelines and usability criteria advise on how certain UI element should be designed [6]. In the context of our research, we are focusing on those usability guidelines that could be evaluated automatically – without the involvement of potential users with the UI evaluation process.

A crucial responsibility of UI designers and quality assurance specialists is to verify that the designed solution satisfies all business requirements and predefined usability guidelines; and that its overall information architecture is clear for potential users. There are two major groups of methods for evaluating usability [7]: empirical and inspection methods. Usability Inspection methods require the expertise of usability inspectors to detect usability problems in user interface design. They include such methods as: Heuristic Evaluation, Formal Usability Inspection, Pluralistic and Cognitive Walkthrough. For their part, Empirical Testing methods require the participation of real users and include: Card Sorting, Eye Tracking and Questionnaires conducted with usability test participants. Empirical Testing is efficient in discovering key issues in information architecture and identifying the flaws and misplacements of elements in web application design. Despite the fact that Empirical Testing methods are commonly more efficient than inspection methods, there are many obstacles preventing the wide application of these methods:

- Organizing and conducting user tests is relatively expensive because it requires a high demand for human and time resources [8], [9];
- Small software companies do not have the funds to pay for complete consultancy or for involving usability specialists, as they are expensive to hire [10];
- Difficulty of getting potential users to participate in usability evaluations [9], [11];
- It is not always possible to increase the coverage of evaluated features as evaluating every single aspect of UI [8] may itself also not be possible.

Herein we focus only on automated usability evaluation. Usability inspection methods are more flexible towards automation, and multiple methods for the latter already exist. In fact, tools on automated inspection are applicable for the majority of WUIs (Web User Interface) without any extra configuration, whereas tools for empirical evaluation require additional application specific configuration as they evaluate interaction-based problems that, in most cases, are application specific.

There are multiple sources of guidelines for inspection methods such as Web Content Accessibility Guidelines (WCAG) [2] and Section 508 [3] standards, multiple design recommendations and best practices to create a good web experience [4], [5]. The evaluation of UI conformance to guidelines using inspection methods can be done without involving potential users of web applications. Moreover, improving the manual usability inspection via automation, is feasible. There are multiple tools for UI automatic evaluation, e.g. Ocawa [*], Magenta[†], and Evaluera[‡]. Yet, these tools are limited to finding deviations in the HTML source code only; and evaluating visual aspects of web application UIs, such as the contrast rate of UI elements, the position of elements on the screen, and many others, is not possible by these tools. Their integration into the WUI development process is extremely complicated and very often not possible at all because solutions are distributed as standalone applications without the possibility of being extended and integrated into the process of continuous delivery.

[*] http://www.ocawa.com

[†] http://giove.isti.cnr.it/accessibility/magenta

[‡] http://www.evaluera.co.uk

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 existing usability guidelines could change or new guidelines appear (e.g. with the emergence of new devices, e.g. tablet computers, or with the evolution of new technologies such as HTML 5). It is highly unlikely that any tool contains usability guidelines suitable for each web application. For instance, e-commerce usability guidelines presented in e-commerce UX report [12] require to embrace large product images showing more details and multiple views of the product. The guidelines for e-commerce however may not be suitable to governmental portals and e-health web applications, e.g. for retrieving patient's prescription information. In general, usability guidelines need to be kept up-to-date, otherwise there is a high risk that a tool soon becomes obsolete.

Different methods exist for describing usability guidelines, including custom usability guidelines. These methods are embedding usability guidelines into the code of evaluation tool, using a definition table based approach [13], and XML-based approaches [14], which are capable of defining various usability guidelines by 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. Nevertheless, defining guidelines to evaluate visual aspects of a WUI such as the presence of scrolling, the layout of elements, the positions of elements on the screen, the distance between elements and many other assessments is not possible with the latter approaches [15]. The main disadvantage of XML Schema is that it defines the structure of a document providing the prescriptions how the document is styled. Thereby, it introduces an additional layer of complexity when it is used to describe the domain of knowledge. A drawback of XML based languages is that they are HTML centric focusing on HTML tags, attributes and their relations. Such approach does not suit for defining visual usability guidelines as the HTML standard does not contain any tags to describe WUI visual characteristics (e.g. there is no HTML tag responsible for scrolling). Adding additional language constructs would make a XML-based language opaque and unclear as it then must support both HTML tag specific definition of guidelines and visual object specific usability guidelines.

To address the aforementioned deficiencies, in this article we explore a possibility to solve the shortcomings of XML-based languages by constructing a special domain ontology which enables to capture usability guidelines for evaluating HTML-code as well as visual aspects of WUI.

Constantly changing business requirements drive a need to update and modify existing UI design and structure. Changing tested and validated UI should be done with extreme caution as even minor change of UI, or the content of a page, could lead to severe usability problems [15]. For instance, changing the color of link text, making it lighter or darker, could potentially lead to severe usability problems, e.g. due to low contrast. According to usability guidelines, the contrast ratio between the letters and the background that is immediately behind the letter should be kept above 4.5:1. Violating this guideline leads to lower usability for the target users, including people with disabilities. Thus, usability is extremely dependent on every modification of UI and, as a result, the immediate evaluation of UI conformance to usability guidelines and subsequent feedback to UI developers becomes another critical demand. That is important because finding usability problems early in the development stage makes the fix less costly than for problems that are found later.

To address the issues of automated WUI evaluation, we propose *Guideliner* – a fully-functional tool for implementation-time automatic evaluation of UI conformance to category-specific usability guidelines during the WUI design and implementation stage. The main contribution of this solution is to enable immediate cost-efficient and automatic web UI evaluation and feedback for developers to ensure the UI under development conforms to set guidelines. Hence, this approach will assist developers and UI testers in finding out usability problems in an automated way in early stages of UI development; taking advantage of a usability ontology that we established to store usability domain knowledge together with custom usability

guidelines. This domain ontology addresses both HTML-centric accessibility guidelines as well as usability guidelines covering visual characteristics of WUI.

The rest of the article is organized as follows. In Section 2 we discuss related works of the research area. Section 3 provides an architecture of the *Guideliner* tool, while Section 4 provides information about evaluation of *Guideliner*. In Section 5 we present the work in progress, and finally in Section 6 we draw conclusions.

2 Related Works

An essential part of every automated usability evaluation tool (including our solution) is a set of guidelines against which the UI is to be evaluated. Web Content Accessibility Guidelines (WCAG) [2] and Section 508 Standards for Electronic and Information Technology [3] are technical standards providing guidelines that explain how to make web content more accessible to target users including people with disabilities. In fact, WCAG and Section 508 standards contain quite similar and partly overlapping accessibility guidelines. Web accessibility is an attribute through which people with disabilities can perceive, understand, navigate, and interact with the web, and, moreover, they can contribute to it [16]. Nevertheless, accessibility is only a certain subset of usability. Many other categories like home page, navigation, content organization guidelines are not covered by standards. That is the reason why many researchers aim to establish usability guidelines covering certain elements of UIs [4], [5] not addressed in standards.

Several researchers have contributed to the development of automated usability evaluation tools [17], [14], [18], [19]. Schiavone and Paterno [14] proposed Mauve – a tool for automated usability evaluation capable of evaluating WUI conformance to WCAG accessibility guidelines of all three levels from A through AAA, containing around 80 different guidelines. Mauve also provides functionality for designing custom usability guidelines in addition to existing set of predefined guidelines. Dingli [17] developed a framework called USEFul for automating usability evaluation enabling a non-expert in the field of usability to conduct usability evaluation. USEFul separates the definition of guidelines from the usability evaluation logic. Such approach allows adding, modifying and deleting of guidelines without altering the source code of the tool. The disadvantage of USEFul is the sophisticated way of adding guidelines. Gay and Li in [20] proposed an open source tool for automated usability evaluation called AChecker containing around 100 guidelines. This tool allows checking the compliance of WUI against WCAG, Section 508 and BITV (a German variant of the internationally recognized web accessibility standard WCAG 2.0) accessibility guidelines. Although AChecker provides an easy to use WUI for triggering the evaluation process, it does not allow defining custom usability guidelines.

In common, all aforementioned tools are limited to finding deviations in the HTML code, lacking the abilities to evaluate visual aspects of a web application like the presence of scrolling, layout of elements on the web page, the position of elements on the screen and the distance between them [15]. This shortcoming has mainly resulted from the selected evaluation approach – these solutions are based on parsing the HTML code and subsequent validation of HTML syntax against guidelines. However, today web applications are built not only using HTML but including stylesheets and scripting into it, and the final rendering is done in user browser. Thereby, in order to evaluate fully functional WUI, it is needed to consider CSS styles and JavaScript scripts that might alter HTML, and, after final rendering, perform visual evaluation of the WUI.

Commercial tools for automated usability evaluation, such as PowerMapper[*], contain, in addition to WCAG accessibility guidelines, also some HTML-specific usability guidelines and search optimization guidelines that ensure correct indexing by search engines. Also, there exist

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

tools that can evaluate certain visual aspects of WUI with some limitations. For instance, Google Mobile Friendly Test[*] service performs a sanity check of WUI checking if it is compatible with mobile devices. It only 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 whether WUI is compatible with mobile devices or not. Another tool called Wave[†] also proposes additional value to WCAG guidelines by checking the contrast rate of elements. In fact, contrast guidelines are a part of WCAG, but very few accessibility tools are capable of evaluating the contrast rate of elements. Nevertheless, Wave lacks the possibility to define custom usability guidelines in this tool.

Analyzing users' behavior and reusing the knowledge with the purpose of providing more usable UI is also a promising research direction. There is a category of tools predicting the usage of the UI based on the knowledge discovery approach [21], [22]. Boza et al. presented a heuristic approach based on data mining techniques with a purpose of determining relationships between UI components and discovering possible problems [22]. Using data mining in combination with mathematical algorithms, they generated rules based on the analysis of test reports. For instance, when "the site prevents users from making mistakes" then "error messages are written in the user language". Preliminary results indicate that the approach is viable for discovering patterns and relationships between different UI components. Commonly though, such approaches cannot guarantee high accuracy of evaluation results, because they tend to have misleading results resulting from the features of algorithms used [23].

An essential output of each evaluation tool is a report providing feedback about UI compliance to predefined usability guidelines. There are multiple types of research analyzing the structure of reports containing usability defects, with the purpose of improving the existing format [24], [25], [26]. Yusop et al. surveyed practitioners in industrial software organizations and in open source communities about their usability defect reporting practices [24]. Their research showed that usability reports should contain at least the following information: title/summary, steps to reproduce, observed result and expected result. In the context of our research, it is important to provide a clear description of usability problems that were found, and to include expected and actual results.

In terms of our previous research in the field of UI usability, we have formulated the problem of assessing application UI conformance against usability guidelines and have designed a framework that could be used to solve the addressed problem [15]. This framework enables the tackling of a large set of usability issues during UI development and saves costs and resources in later system development phases, especially in testing. In [27] we outlined the problem of design-time usability evaluation and presented proof of a solution concept that provide that it is feasible to evaluate usability automatically during the design phase of UI development. The value of the current study with respect to the previous achievements is that we propose a fully functional tool called *Guideliner* for implementation-time usability evaluation of web UI. Our purpose is to provide an overview of all developed components, emphasizing technical implementation details and also to perform the testing of *Guideliner* based on different web applications.

3 Guideliner – Tool for Automated Usability Evaluation

In order to tackle the problems of implementation-time usability evaluation, this section delivers a system (called *Guideliner*) that addresses the problem and enables automated evaluation of WUI conformance to usability guidelines (HTML-centric as well as visual usability guidelines)

[*] https://search.google.com/test/mobile-friendly
[†] http://wave.webaim.org

already to be accomplished during the implementation phase of WUI development. Also, *Guideliner* makes it possible 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 web UI as it performs the evaluation of HTML-specific usability guidelines as well as guidelines addressing the visual characteristics of web UI. The applicability of the *Guideliner* does not stick to any particular web UI development process (e.g. agile or waterfall); but rather it is a universal tool challenging the problem of immediate usability evaluation, especially during web UI development and implementation phases.

3.1 Guideliner Overview

Guideliner is based on Selenium Web Driver[*] – a tool that provides API for automated functional WUI testing, and that verifies that WUI behaves in the way expected. *Guideliner* takes advantage of the Selenium Web Driver, which provides a full-stack of instruments and operations needed for automated testing of user interfaces, and uses it as a mechanism to automate the evaluation process of visual usability aspects against usability guidelines. In principle, this provides *Guideliner* with the capability to evaluate WUIs against usability guidelines on various platforms, including desktop browsers, e.g. Mozilla Firefox, Google Chrome, Internet Explorer and others; and different mobile platforms such as Android and iOS with their browser versions.

Architecturally, *Guideliner* is divided into four self-sufficient software modules based on the aspect of functionality they cover (see Figure 1 for more details). Such an approach can be called a separation of concerns [28] and is widely used in the software development industry when one component has a very limited and narrow scope of functionality that it is responsible for. That approach allows development and testing of each component in isolation from the rest of the system, eliminating failures caused by unintentional side effects.

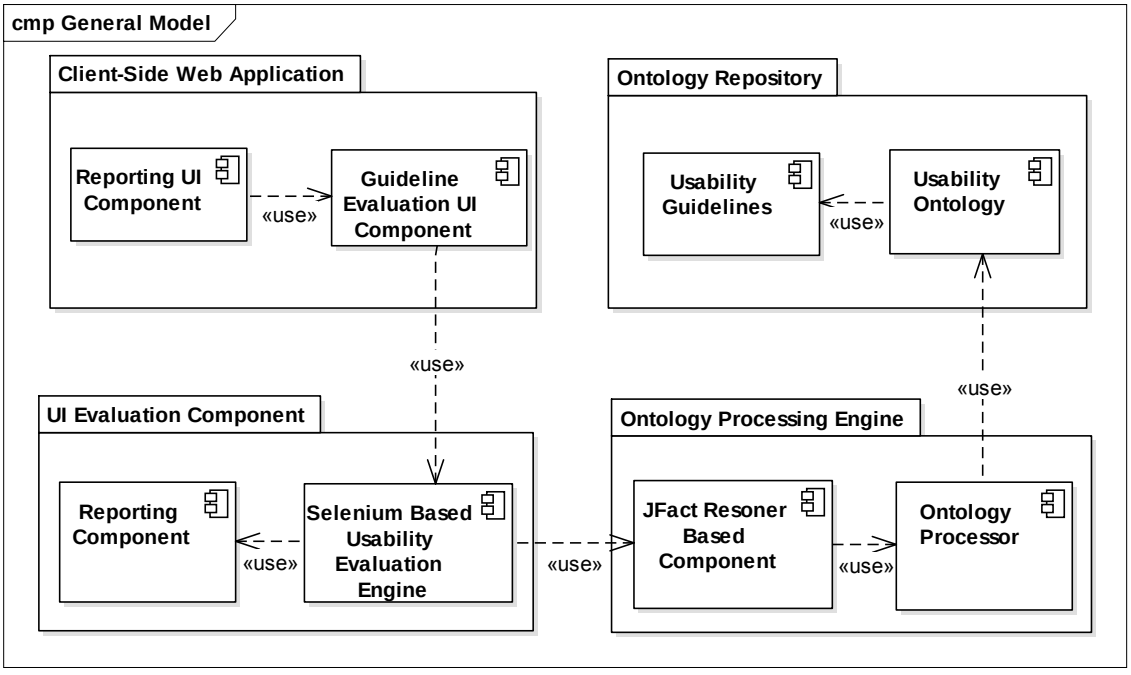

Figure 1. High-level architecture of *Guideliner*

A core element of *Guideliner* is an Ontology Repository containing descriptions of usability guidelines that can be machine-processed on a particular WUI. Such an approach simplifies sharing of metrics and concepts between various guidelines and between different groups. The

[*] https://www.seleniumhq.org/projects/webdriver/

Ontology Processing Engine is a mediator component responsible for errorless communication between the ontology repository and the user interface evaluation component. The UI evaluation component assesses UI conformance to usability guidelines containing a reporting component responsible for generating a usability evaluation report. A client side web application contains easy to use UI for managing guidelines and triggering the evaluation process.

Initially, a usability guideline is described by means of ontology. Then, the *Ontology Processing Engine* transforms the guideline to the format understandable to the *UI Evaluation Component*. Afterwards, the process of automatic evaluation is triggered, checking whether the UI is accessible from the browser and whether there is a predefined set of guidelines to be processed. Then, the evaluation of the UI is performed according to the set of guidelines. Finally, a report is provided to the developer, containing the conformance of UI to the usability guidelines.

The proposed solution supports HTML 4 and 5, CSS 3 (providing backward compatibility with previous versions) and JavaScript based user interfaces. It does not require additional adaptation for web user interfaces, allowing evaluation of any web UI without any extra configurations.

Below a brief overview of each component is presented:

- *Ontology Repository* is usability ontology consisting of descriptions of usability guidelines that are machine-processable. *Ontology Repository* is responsible for saving and modifying specified guidelines into categories such as desktop or mobile platform usability guidelines. Guidelines include various aspects and parts of UIs such as information organization guidelines, link visual consistence guidelines, text appearance guidelines and form guidelines.
- *Ontology Processing Engine* is an intermediary component responsible for reliable data transfer between *Ontology Repository* and *UI Evaluation Component*. *Ontology Processing Engine* uses the ontology as an input and returns transformed data into the format specific to *UI Evaluation Component*.
- *UI Evaluation Component* assesses web UI conformance to predefined usability guidelines. Also, it contains a reporting component used for generating usability evaluation reports based on the web UI usability evaluation results. The component contains the main evaluation logic including opening the browser with the web UI being evaluated and comparing the conditions set in usability guidelines with actual results extracted from web UI.
- *The Client-Side Web Application* is an easy-to-use UI for managing usability guidelines and triggering the evaluation process. Also, it presents comprehensive evaluation results containing detailed descriptions of passed and failed guidelines.

3.2 Ontology Repository

Ontology is a formal, explicit specification of a conceptualization [29]. Ontology is a prominent component of an intelligent system. It enables storing and capturing domain knowledge in a human-understandable, and, at the same time, machine-processable way. In general, ontology contains entities, attributes, relations and axioms, allowing formal presentation of knowledge as concepts within a domain, and the relations between these 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 paper 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. OWL was selected as it is the most widely used language for creating ontologies [30]. OWL enables capturing knowledge by representing the concepts and relations between them. Primary components of OWL ontologies are classes (sets consisting of individuals), properties

(relationships that link two individuals together) and individuals (also called instances). The most frequently applied relations between ontology concepts are the is-a relation, that defines the hierarchy between class and sub-class, and the part-of relation, defining the relationships of an entity and its components. The formal semantics of OWL allow inferring of classification taxonomies and thus help to identify inconsistencies in the established ontology at any time. Thus, OWL is progressive language that is used in developing intelligent systems, and also coincides with the aim of current work. We used Web Ontology Language (OWL) as a knowledge representation language; and open source feature rich Protégé ontology editor 5.1[*] for creating the ontology.

The idea of integrating WUI usability guidelines into ontology is not new. Xiong et al in [31] presented an ontology-based approach for organizing and generalizing usability guidelines. They point out that the main drawback of their approach is the necessity for additional mappings. Also, the ontology they proposed contained only HTML-specific domain knowledge; e.g., concepts for defining HTML tags, attributes and relations between them. The authors point out that, with their approach, it is impossible to define usability guidelines for describing background color or contrast between elements on the page.

Our ontology [32] defines only those usability guidelines that can be automatically evaluated. The ontology contains WCAG and Section 508 guidelines including guidelines involved with people with disabilities as well as common usability guidelines. Presently the ontology is used only for storing usability domain knowledge. It does not perform any kind of evaluations of UI conformance to the guidelines; however, based on available descriptions this could be achieved.

Established usability ontology contains main concepts defined as primitive classes (they have only necessary conditions defined) including *Guideline*, *GuidelineElement*, *ElementAttribute*, *PageAttribute* and *ValuePartition* (a special class used to refine guideline descriptions through a pre-defined set of value concepts). Figure 2 outlines class hierarchy of defined usability ontology and some descendant classes.

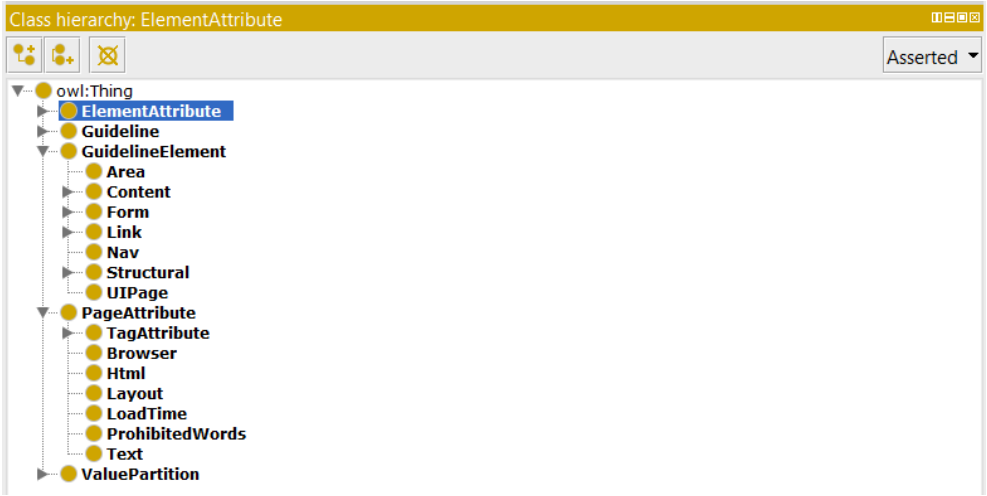

Figure 2. Class Hierarchy of defined usability ontology and some descendant classes: excerpt from the Protégé ontology editor

The purpose of ontology main concepts is described below in more details:

- *Guideline* – all guidelines are described as subclasses of the general *Guideline* class. *Guideline* class can hold both primitive and defined classes. Defined classes are used to infer new subclasses (e.g. guideline classes concerned with links) via reasoning based on the knowledge already available in the ontology.

[*] Protégé, http://protege.stanford.edu/

- *GuidelineElement* – descriptions of elements a guideline may be applied to. *GuidelineElement* class holds various WUI elements (e.g. link, button, text input) that could be defined as an element being evaluated.
- *ElementAttribute* – defines attributes of WUI elements. *ElementAttribute* class can hold both visual characteristics (e.g. distance, alignment) of WUI elements as well as HTML-centric attributes (e.g. alternative text, title).
- *PageAttribute* – defines attributes applicable to the page as a whole such as page layout and load time.
- *ValuePartition* – used to refine the guideline descriptions. For instance, describing the importance and strength of evidence of a particular guideline.

Before we continue with usability ontology, let us have a look on how the guidelines are presented. Generally, usability guidelines are presented in a text format. Table 1 demonstrates usability guideline presented by Google Mobile research group[*]: *All buttons should be at least 48 CSS pixels wide*. The example guideline presentation contains the following items: brief name of the guideline, detailed description of the guideline (including the detailed explanation of the logic behind the guideline), type of guideline (visual usability guideline or the HTML-centric guideline), evaluation conditions (how the guideline should be evaluated), platform (mobile or desktop) and source (reference to the standards). The presented guideline is suitable for automated usability evaluation as it contains concrete evaluation condition that the width of the link for mobile device should not exceed 48 pixels.

Table 1. Example guideline on link width targeting mobile platforms presented in a text format with evaluation conditions suitable for automated usability evaluation

Guideline:	Every link on WUI should be at least 48 CSS pixel wide.
Description:	The average size of finger pad is approximately 10 millimeters for adults. The minimal recommended size of tap target is about seven millimeters that are roughly equal to 48 CSS pixels.
Type of guideline:	Visual guideline. The width of links can be calculated only on the finally rendered WUI; it is not possible to calculate the width of links accurately parsing the source code of the page (as the width can be affected by JavaScript scripts and various CSS styles).
Evaluation conditions:	The width of every link in pixels on the page should be calculated and checked that it is more or equal to 48 pixels.
Platform:	Mobile
Source:	Google HCI, Android HCI

Let us define guidelines presented in the text format using introduced usability ontology. Usability guidelines are defined in the ontology as subclasses of class *Guideline*. Figure 3 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, a new subclass is added as a subclass *UsabilityGuideline*, and a naming convention shown by Figure 4 must be applied. The name of the class starts with a unique identifier (this is needed to distinguish between usability guidelines), followed by the code of the category (determines what is the category of usability guideline), finally, the short name of the guideline is added. The class is made to be disjointed from other subclasses of the class *UsabilityGuideline*, so the instance of one subclass of *UsabilityGuideline* class cannot be the instance of another subclass (as each usability guideline is unique). As shown in Figure 3, class *28-ButtonShouldBeWideEnough* contains the object property *hasGuidelineElement* which defines the element being evaluated. The statement *hasGuidelineElement only Button* shows that the defined guideline evaluates only labels; the

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

statement *hasDeviceType only Mobile* outlines that the guideline is applicable only to mobile devices.

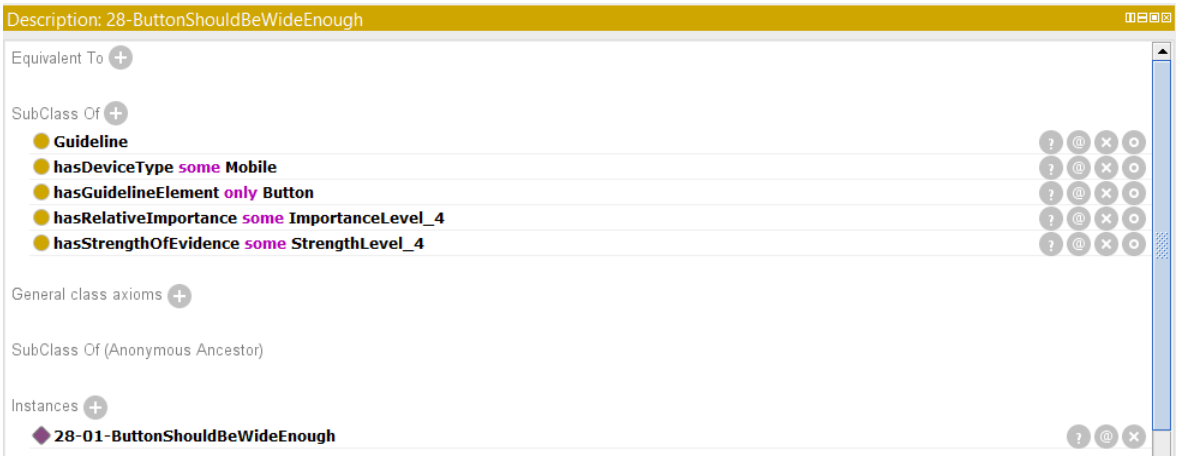

Figure 3. Example of a usability guideline concept definition for the guideline "Buttons should be wide enough" (screenshot from the Protégé ontology editor)

Figure 4. Naming convention applied while describing classes of usability guidelines in the ontology

Additional information about the guidelines is provided as *Class Annotations* where the 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 5).

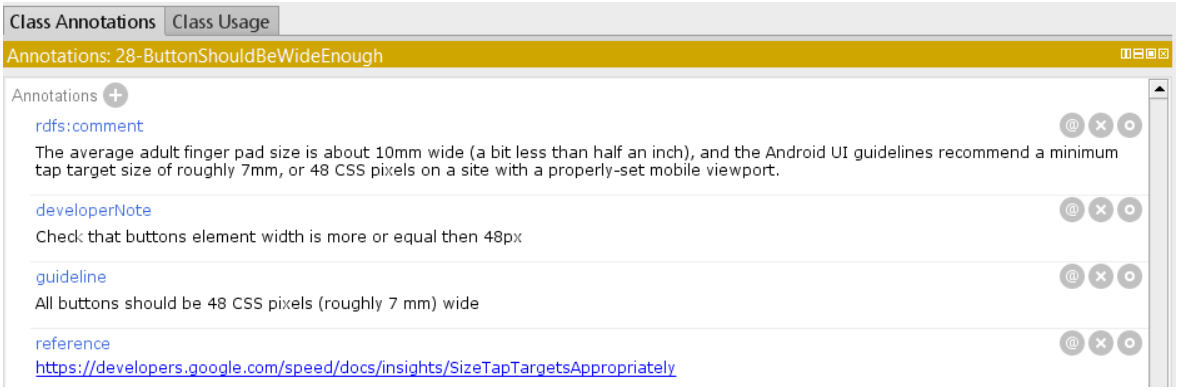

Figure 5. Example of guideline annotation with human-readable comments (screenshot from the Protégé ontology editor)

3.3 UI Evaluation Component

The ontology described in Section 3.2 is used as an input for *Ontology Processing Component*. Before evaluating UI, we should transform usability guidelines (individuals) defined in ontology by means of OWL Web ontology language (in XML format) to a format that is understandable by the UI evaluation engine. The transformation is required because processing guidelines in native OWL format is not trivial due to the complicated API of OWL language. OWL API [33]

library has been used for serializing usability ontology into appropriate Java classes. The library is aligned with OWL 2 structural specification providing interfaces for parsing, rendering the ontology and manipulation of ontological structures. We used JFact for reasoning over the domain – a Java port of FaCT++ reasoner [34] having full compatibility with the OWL API library.

UI Evaluation Component uses *Ontology Processing Engine* for retrieving usability guidelines from *Ontology Repository*. Afterwards, it identifies the element of WUI (and its corresponding characteristics) that should be evaluated (see Figure 6). Then, it calls evaluation adapter (adapters are responsible for performing the evaluation of certain WUI components; one such, *LinkAdapter*, is responsible for evaluation of link specific characteristics) providing the characteristics of the element as a parameter of the adapter (e.g. contrast $\geq$ 4.5). The corresponding adapter retrieves the actual values of the WUI element characteristic being evaluated using Selenium WebDriver and asserts whether the retrieved values are corresponding to the value defined in the usability guideline. An illustrative example could be that if the length of the *Link* text is evaluated then the *LinkAdapter* asks Selenium WebDriver to find all *Links* and calculate the length of *Link* text. Afterwards, *LinkAdapter* checks if the values returned by Selenium WebDriver 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 the success response is generated; otherwise a failure response is generated.

Thereby, the core mechanism of processing the Web User Interface is Selenium WebDriver API[*]. Selenium is used for automation of UI tests providing a simple and concise programming interface. Selenium has full support for most programming languages (Java, C#, Python, JavaScript, etc.), being compatible with most popular browsers (Chrome, Firefox, Internet Explorer, etc.). Selenium WebDriver provides rich API commands and operations containing interfaces for fetching a page, locating UI elements on the screen, filling in forms and many other operations.

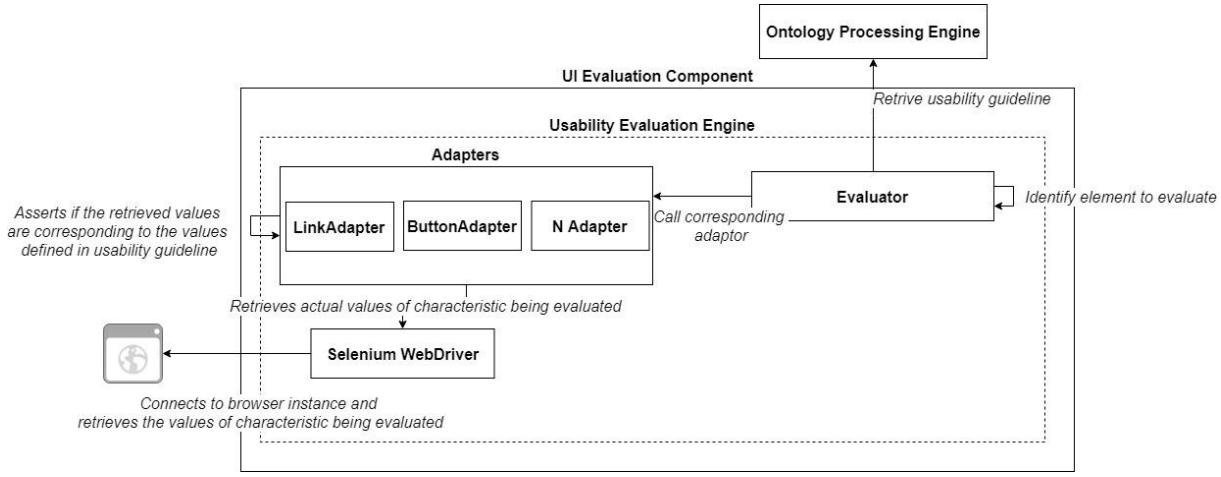

Figure 6. Process of Evaluation WUI conformance to usability guideline

3.4 Client Side Web Applications

Client side web application contains easy to use UI for managing guidelines and triggering the evaluation process. The potential users of the proposed solution are technical personnel (including developers and quality assurance specialists) and business users (including analysts and product owners). The primary deliverable for UI developers is a library providing the API for evaluating UI conformance to the guidelines on local or remote machine. Business users do

[*] http://www.seleniumhq.org/projects/webdriver

not commonly have a required technical background to run UI tests from the code. That is why a web application has been designed containing visual functionality for managing guidelines and triggering the evaluation process.

At first, a set of guidelines to be evaluated should be selected. After selecting the category of usability guidelines to be evaluated, the URL of the web application to be evaluated must be specified. After this, the evaluation process can be initiated.

Once the evaluation process has been finished, the report containing usability evaluation results is shown (Figure 7). Violated guidelines are highlighted in red color; passed guidelines – in green color. The progress bar on the top of the screen shows the ratio of failed and passed tests. Evaluation results provide full information of evaluated guidelines including name, code and description.

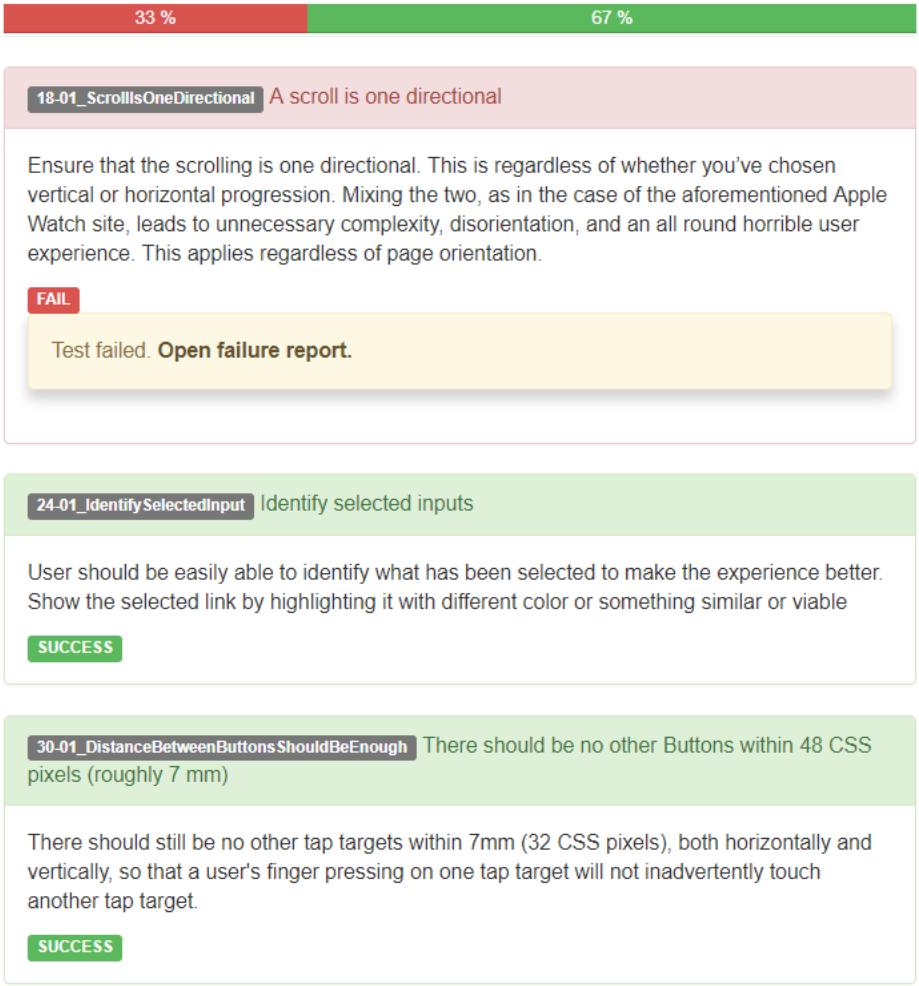

Figure 7. Screenshot of the view containing usability evaluation results

By clicking the link *Open Failure Report* a dialogue opens containing a screenshot, text and the type (e.g. link, button) of the element violating the guideline (see Figure 8). Also, it contains a human readable explanation of the reasons for the failure's value. Such a screenshot is provided whenever possible – for guidelines checking the consistency and validity of HTML code, in common, no screenshot images are presented; whereas for guidelines evaluating the visual characteristics of WUI a screenshot of the failed element is always presented making it easier to understand the reasons for evaluation failure.

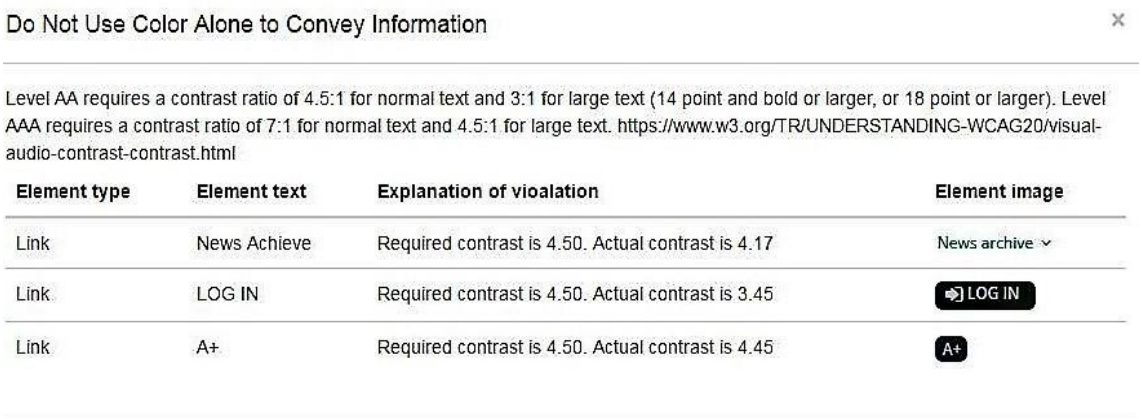

Figure 8. Screenshot of dialog presenting additional information about a failed guideline

The *Client-Side Web Application* is a single-page web application that has been built based on Angular 4 JavaScript framework[*] using Bootstrap[†] stylesheets containing an extensive list of components for designing client web applications. Alternatively, any other front end JavaScript framework might be selected, such as ReactJS[‡]. In fact, ReactJS and Angular are the most widely used JavaScript frameworks[§], both technologies being powerful and flexible. As there are are no obvious advantages in either technology, we decided in favor of Angular because of the extensive experience and better expertise in Angular web application development.

Instead of using pure JavaScript, we used TypeScript[**] – a 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 any limitations to using Typescript instead of pure JavaScript.

4 Evaluation of Guideliner

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 upon [35]. From the perspective of *Guideliner*, the software verification process should prove that *Guideliner* is capable of evaluating different types of applications (e.g. entertainment portals, public sector portals, etc.) effectively, regardless of the technology used on UI. Despite the fact that *Guideliner* supports HTML/CSS/JavaScript based WUI, the frameworks used for WUI development and the methods used for composing and formalizing the structure of WUI HTML/JavaScript/CSS code can, potentially, affect the accuracy of the evaluation process. There are two types of software verification in common use: 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 the software runtime, verifying that the software behaves in an expected way.

The current article concentrates on dynamic verification with the purpose of finding deviations in *Guideliner's* behavior when conformance to usability guidelines for different WUIs is evaluated. Different types of deviations can be detected such as slow response time, errors occurring during evaluation, or missing response fields (e.g. missing status of evaluation, missing error description or missing name of an element that violates the guideline).

[*] Angular Framework, https://angular.io/
[†] Bootstrap CSS, https://getbootstrap.com/
[‡] React, https://reactjs.org/
[§] https://insights.stackoverflow.com/survey/2016
[**] Typescript, https://www.typescriptlang.org/docs/home.html

To perform dynamic verification of *Guideliner*, 14 different web applications of Estonian public service organizations and USA portals from different areas; such as medicine, real estate, environment and news were used. Estonian public sector portals were selected, as target users of such portals are all inhabitants of Estonia, having different experiences in using various web user interfaces and various devices to access the applications. Thus, all selected web applications had to satisfy general usability guidelines applicable for most web applications. In addition to public sector web applications, Estonia and USA news portals were selected, providing information services for wider audiences. The following web applications and portals were selected for evaluation:

- Republic of Estonia Road Admission Portal[*] having the same design template as all other ministry web pages of the Republic of Estonia,
- E-government Portal[†] providing e-services for all inhabitants in Estonia,
- Government Real Estate Portal[‡] providing real estate services,
- Government info system management portal [§] containing repositories for public e-government services,
- Estonian Research Information System [**] providing information about Estonian research activities,
- Republic of Estonia Information System Authority[††],
- Public healthcare institutions each having absolutely individual design themes: East-Tallinn Central Hospital[‡‡] web site and Rakvere hospital web site[§§],
- Estonian News Portal Delfi[***],
- Estonian News Portal Postimees[†††],
- Estonian Business News Portal Aripaev[‡‡‡],
- National Aeronautics and Space Administration (NASA) web page[§§§],
- CNN news portal[****],
- BBC news portal[††††].

All web applications under study have different WUI design, target users and categories. All of the aforementioned applications are based on HTML, JavaScript and CSS technologies and, thus, can be evaluated by *Guideliner*.

To prove the viability of the *Guideliner* approach, a set of 98 predefined usability guidelines was used that covers most elements of WUI including Links, Tags and Text Appearance (see Table 2 for more details). The set of guidelines contains 55 accessibility guidelines, 23 common usability guidelines suitable for desktop and mobile devices and 20 usability guidelines suitable only for mobile devices. The conformance of WUIs to usability has been automatically performed on Chrome version 65.0.3325.181. All 98 selected usability guidelines were defined using usability ontology presented in Section 3.2.

In order to conduct an experiment, a special Java program has been created that uses *Guideliner* REST API for initiating the evaluation process (URL of web application under study was used as a request parameter) and for storing *Guideliner* evaluation responses to database for further processing. The stored data contains the evaluated guidelines, evaluation result status,

[*] https://www.mnt.ee/eng
[†] https://www.eesti.ee/en/
[‡] http://www.rkas.ee/en
[§] https://riha.eesti.ee/riha/main
[**] https://www.etis.ee/?lang=ENG
[††] https://www.ria.ee/en/
[‡‡] http://www.itk.ee/en
[§§] http://www.rh.ee/
[***] http://www.delfi.ee/
[†††] https://www.postimees.ee/
[‡‡‡] http://www.aripaev.ee/
[§§§] https://www.nasa.gov/
[****] http://edition.cnn.com/
[††††] https://www.bbc.co.uk/

WUI URL and data of WUI elements that violated the guidelines, including the violation reason and the text of the element.

Table 2. Categories of usability guidelines for the established usability ontology

Category	Mobile	Common	Accessibility	Total
Organization 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

Table 3 presents the results of the evaluation, pointing out the number of guidelines violated by each web application. The experiment's results show that the accessibility guidelines were most commonly violated by WUIs under study.

The most frequently occurred violations were documented during the testing for each web application. The guideline stating that *The width of the link should be at least 48 pixels for mobile devices presented* was violated by every WUI under study. Another usability guideline that was violated by all WUIs under study was *The distance between links should be at least 30 pixels for mobile devices*. The most violated common usability guidelines were the guidelines checking the contrast ratio of the elements of WUI.

The common problem of most WUIs under study is small size and the wrong contrast ratio of clickable elements (links, buttons). Such violations complicate text reading and navigation between web pages for people, including those with color blindness or other visual impairments. Really, that should be avoided, especially on governmental web applications. In particular, the small size of clickable elements and the insufficient distance between elements was detected on a mobile platform. In fact, these guidelines are even more critical on the mobile platform as small or closely located clickable elements are more complex for users to tap accurately on a touchscreen than with a common mouse. The common rule for the size of links and buttons is based on the research done by Google[*] – the most critical links and buttons should be at least 48 48 CSS pixels wide and there should not be any other tap targets within 7 mm either horizontally or vertically. These guidelines were violated by every WUI under study on a mobile platform.

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

Table 3. Results of usability evaluation executed on various WUIs

Web Application	Mobile Usability	Common Usability	Accessibility	Total
https://www.ria.ee/en/	7	4	8	19
https://www.postimees.ee/	6	6	10	22
https://www.nasa.gov/	7	3	9	19
https://www.mnt.ee/eng	7	5	8	20
https://www.etis.ee/?lang=ENG	6	7	12	25
https://www.eesti.ee/en/	6	3	7	16
https://www.aripaev.ee/	6	6	13	25
https://riha.eesti.ee/riha/main	2	2	6	10
http://www.rkas.ee/en	7	3	10	20
http://www.rh.ee/	6	5	10	21
http://www.itk.ee/en	4	4	8	16
http://www.delfi.ee/	8	6	13	27
http://www.bbc.com/	7	6	8	21
http://edition.cnn.com/	8	6	12	26

The evaluation of *Guideliner* proved that the taken approach with usability ontology is feasible and provides enough functionality for describing different types of usability guidelines including mobile, common and accessibility guidelines for WUI evaluation. The *UI Evaluation Component* that used guidelines defined in ontology to perform the evaluation was able to handle and process guidelines described in the ontology, and no run-time errors occurred. The component successfully carried out the evaluation of different aspects of WUI such as layout and element positioning, contrast rate, HTML-validity and so forth. Overall, there were no exceptions thrown up during the evaluation of *Guideliner*. All evaluation responses were properly structured containing relevant information about violations detected in the code, such as the description and the text of the element.

5 Work in Progress

The Estonian Information Systems Authority [*] (RIA), responsible for development of governmental portals in Estonia, showed a great interest into Guideliner and thus it was introduced to their development process with the purpose to integrate *Guideliner* to evaluate the usability of Estonian eGovernment components. RIA coordinates the development and administration of Estonian State Portal[†].

For the first phase of the integration, we studied the Estonian State Portal (ESP) – eesti.ee, which is the primary gateway to public information and services in Estonia. It is a secure Internet environment that the residents of Estonia use to access the state's information, services and portals. The main reason why Estonian State Portal was selected for our research is because it is the most popular public governmental portal with more than 40 Million visits during the year 2016 as stated in [36]. The UI of ESP is based on HTML, CSS and JavaScript technologies.

The main motivation for integrating automated usability and accessibility evaluation into the RIA development process is to check automatically the compliance of the portal UI to WCAG standards. The conformance to WCAG became an even more critical requirement after the directive, on making websites and mobile apps of public sector bodies more accessible, was

[*] RIA, https://www.ria.ee/en/

[†] Estonian State Portal, https://www.eesti.ee/eng

introduced on 26 October 2016 (Directive (EU) 2016/2102 of the European Parliament and of the Council of 26 October 2016 on the accessibility of the websites and mobile applications of public sector [37]). The main purpose of this directive is to encourage all governmental applications to be more accessible by people with disabilities, by increasing the importance of following WCAG standard guidelines.

Despite the fact that there are multiple existing solutions for checking conformance to the WCAG standard (e.g. AChecker [15] and Mauve [14]), there are no solutions that could be integrated into the RIA development process that entirely suit their development model. For instance, both AChecker and Mauve are distributed as standalone web applications requiring a separate server with special configurations (AChecker requires Apache server with preinstalled PHP environment), where they could be deployed. Such an approach does not comply with the RIA development process requiring that tools for automated usability evaluation could be built and run as a part of development process and executed every time developers commit changes to the source code of applications. *Guideliner* addresses the disadvantages of existing solutions by supporting integration into web UI development process, enabling implementation-time usability evaluation.

Figure 9 demonstrates the view of the ESP home page containing the most popular sections of the web application. The UI is designed using a responsive web design approach and adapts to the size of the screen it is viewed on. This means that the positions of elements, their size and design, vary across devices ensuring high usability for every device.

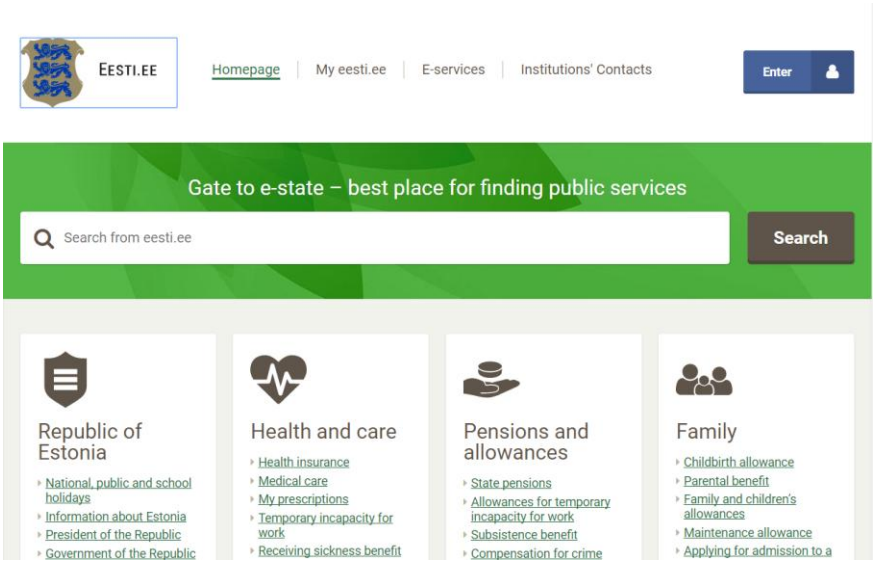

Figure 9. Screenshot of Estonian State Portal containing the most popular sections of the web application

In cooperation with RIA quality assurance team and designers, we concluded that integration of the framework for the evaluation of UI compliance to usability guidelines (especially WCAG guidelines) is beneficial for the RIA. Based on the feedback provided, we extended designed ontology by dividing all guidelines into subgroups like WCAG and Section 508 guidelines. Such an approach allows the evaluation of UI conformance only to the selected group of guidelines, not to all of them.

6 Conclusions

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

In order to tackle the problem of design-time usability evaluation, we deliver a system called *Guideliner* that enables automated evaluation of conformance to usability guidelines (both

HTML-centric and visual usability guidelines) during design phase of WUI development. Also, *Guideliner* provides possibility to perform usability pre-release testing verifying that all developed features are compliant with usability guidelines. In general, *Guideliner* increases the overall quality of developed WUIs as it performs the evaluation of both HTML-specific and visual usability guidelines. Even further, the applicability of *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.

A major contribution of this work is the domain ontology for storing usability knowledge. The ontology design allows to define custom usability guidelines as necessary, and thus extend *Guideliner* also for specific use cases. The usability ontology is proposed as a solution to overcome the limitations of existing approaches suffering from inability to define visual usability guidelines; and presents an alternative method to define usability guidelines in a machine-processable form, yet in a human-understandable way.

In summary, our tool *Guideliner* is capable of evaluating HTML and JavaScript based web user interfaces without additional configurations and check their compliance to the WCAG and Section 508 standards, as well as to best practices and recommendations for UI design introduced in scientific publications and various usability researches. Presently it supports 98 essential usability guidelines, and we continue adding new ones.

References

[1] International Organization for Standardization: ISO 9241-210:2010 Ergonomics of human-system interaction Part 210: human-centred design process for interactive systems, 2010.

[2] Web Content Accessibility Guidelines. [Online]. Available: http://www.w3.org/WAI/intro/wcag

[3] Section 508. [Online]. Available: https://www.section508.gov/

[4] User Experience for Mobile Applications and Websites. [Online]. Available: https://www.nngroup.com/reports/mobile-website-and-application-usability/

[5] Navigation and Page Layout. [Online]. Available: https://www.nngroup.com/reports/intranet-navigation-layout-and-text/

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

[7] E. Kock, J. Biljon, and M. Pretorius, "Usability evaluation methods: mind the gaps," *Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, ACM, pp.122–131, 2009. Available: https://doi.org/10.1145/1632149.1632166

[8] M. Y. Ivory and M. Hearst, "The state of the art in automating usability evaluation of user interfaces," *ACM Comput. Surv.*, vol. 33, pp. 470–516, 2001. Available: https://doi.org/10.1145/503112.503114

[9] J. O. Bak, K. Nguyen, P. Risgaard, and J. Stage, "Obstacles to usability evaluation in practice: a survey of software development organizations," *Proceedings of the 5th Nordic conference on Human-computer interaction*, ACM, pp. 23–32, 2008. Available: https://doi.org/10.1145/1463160.1463164

[10] A. Häkli, "Introducing user-centered design in a small-size software development organization." M.S. thesis, Department of Computer Science and Engineering, Helsinki University of Technology, Helsinki, 2005.

[11] F. Lizano, M. M. Sandoval, A. Bruun, and J. Stage, "Is Usability Evaluation Important: The Perspective of Novice Software Developers," *Proceedings of the 27th International BCS Human Computer Interaction Conference*, Article 31, British Computer Society, Swinton, 2013.

[12] E-Commerce User Experience. [Online]. Available: https://www.nngroup.com/reports/ecommerce-user-experience/

[13] 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.

[14] A. G. Schiavone and F. Paterno, "An extensible environment for guideline-based accessibility evaluation of dynamic Web applications," *Journal Universal Access in the Information Society*, pp. 111–132, 2015.

[15] J. Marenkov, T. Robal, and A. Kalja, “A framework for improving web application user interfaces through immediate evaluation,” *Databases and Information Systems*, vol. 291, pp. 283 – 296. IOS Press, 2016. Available: https://doi.org/10.3233/978-1-61499-714-6-283

[16] World Wide Web Consortium W3C, 2010b. Web Accessibility Initiative (WAI). [Online]. Available: http://www.w3.org/WAI/intro/accessibility.php

[17] A. Dingli, “USEFul: A Framework to Mainstream Web Site Usability,” *International Journal of Human Computer Interaction*, pp. 10–30, 2011.

[18] A. Dingli and S. Cassar, “An intelligent framework for website usability,” *Advances in Human-Computer Interaction*, Article 5, 2014. Available: https://doi.org/10.1155/2014/479286

[19] B. Leporini, F. Paterno, and A. Scorcia, “Flexible tool support for accessibility evaluation,” *Interacting with Computers*, vol. 18(5), pp. 869–890, 2006. Available: https://doi.org/10.1016/j.intcom.2006.03.001

[20] G. Gay and C. Q. Li, “AChecker: open, interactive, customizable, web accessibility checking,” *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility* (W4A), ACM, p. 23, 2010. Available: https://doi.org/10.1145/1805986.1806019

[21] P. A. Davis and F. M. Shipman, “Learning usability assessment models for web sites,” *Proceedings of the 16th international conference on Intelligent user interfaces*, ACM, pp. 195–204, 2011. Available: https://doi.org/10.1145/1943403.1943433

[22] B. C. Boza, S. Schiaffino, A. Teyseyre, and D. Godoy, “An approach for knowledge discovery in a web usability context,” *Proceedings of the 13th Brazilian Symposium on Human Factors in Computing Systems*, pp 393–396, 2014.

[23] M. A. A. Winckler, C. M. D. S. Freitas, and J. V. De Lima, “Usability remote evaluation for WWW,” *CHI '00 Extended Abstracts on Human Factors in Computing Systems* (CHI EA '00), ACM, pp. 131–132, 2000. Available: https://doi.org/10.1145/633292.633367

[24] N. S. M. Yusop, J. Grundy, and R. Vasa, “Reporting usability defects: do reporters report what software developers need?” *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, ACM, pp. 1–10, 2016. Available: https://doi.org/10.1145/2915970.2915995

[25] S. Davies and M. Roper, “What’s in a Bug Report?” *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Article 26, ACM, 2014. Available: https://doi.org/10.1145/2652524.2652541

[26] N. S. M. Yusop, J. Grundy, and R. Vasa, “Reporting Usability Defects: Limitations of Open Source Defect Repositories and Suggestions for Improvement,” *Proceedings of ASWEC Australasian Software Engineering Conference*, ACM, pp. 38–43, 2015. Available: https://doi.org/10.1145/2811681.2811689

[27] J. Marenkov, T. Robal, and A. Kalja, “A Tool for Design – Time Usability Evaluation of Web User Interfaces,” in Kirikova M., Nørvåg K., Papadopoulos G. (eds.) *Advances in Databases and Information Systems*. *Lecture Notes in Computer Science*, Springer, vol. 10509, 2017. Available: https://doi.org/10.1007/978-3-319-66917-5_26

[28] E. Ernst, “Separation of concerns,” *Proceedings of the AOSD 2003 Workshop on Software-Engineering Properties of Languages for Aspect Technologies* (SPLAT), Boston, MA, USA. 2003.

[29] T. R. Gruber, “Toward principles for the design of ontologies used for knowledge sharing?” *International journal of human-computer studies*, vol. 43(5–6), pp. 907–928, 1995. Available: https://doi.org/10.1006/ijhc.1995.1081

[30] L. Yu, “A developer’s guide to the semantic Web,” Springer Science & Business Media, 2011. Available: https://doi.org/10.1007/978-3-642-15970-1

[31] J. Xiong, C. Farenc, and M. Winckler, “Towards an ontology-based approach for dealing with web guidelines,” *Proceedings of the International Conference on Web Information Systems Engineering*, Springer, pp. 132–141, 2008. Available: https://doi.org/10.1007/978-3-540-85200-1_15

[32] T. Robal, J. Marenkov, and A. Kalja, “Ontology design for automatic evaluation of web user interface usability,” in *Portland International Conference on Management of Engineering and Technology (PICMET)*, IEEE , pp. 1-8, 2017. Available: https://doi.org/10.23919/picmet.2017.8125425

[33] M. Horridge and S. Bechhofer, “The OWL API: A Java API for OWL ontologies,” *Semantic Web*, vol. 2, pp. 11–21, 2011.

[34] D. Tsarkov and I. Horrocks, “FaCT++ Description Logic Reasoner: System Description,” *Proceedings of the Third international joint conference on Automated Reasoning*, Springer, pp. 292–297, 2006. Available: https://doi.org/10.1007/11814771_26

[35] D. Beyer, “Status report on software verification,” *Proceeding of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science*, Springer, vol 8413, pp. 373–388, 2014. Available: https://doi.org/10.1007/978-3-642-54862-8_25

[36] The State Portal eesti.ee in numbers. (2017, November 24). [Online]. Available: https://www.eesti.ee/eng/topics/business/riigiportaali_abi/partnerile_1/eesti_ee_2016_aasta_statistika

[37] 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. (2017, November 24). [Online]. Available: https://eur-lex.europa.eu/homepage.html