Ontology Design for Automatic Evaluation of Web User Interface Usability

Tarmo Robal[1], Jevgeni Marenkov[1], Ahto Kalja[2]
[1] Dept. of Computer Systems, Tallinn University of Technology, Tallinn, Estonia
[2] Dept. of Software Science, Tallinn University of Technology, Tallinn, Estonia

Abstract—**The rapid development of the Internet and associated technologies for content delivery has led to a situation where the Web can be accessed on a multitude of different platforms – from desktop computers, laptops and tablets to smart-phones, which have become a crucial part of our lives, raising the question of usability on this plethora of devices. Testing and validating user experience (UX) throughout the whole development process is costly. However, some of this work done by humans could be executed automatically starting in early development. In this paper we address web user interface (UI) automatic evaluation and in particular discuss the ontology design for capturing knowledge of web usability domain for UI evaluation.**

I. Introduction

The proliferation of the Internet, development of hard- and software, and the advances in technologies for content delivery has led to a situation where the Web can be accessed not only from personal computers but on a multitude of different platforms including desktop computers, laptops and tablets to smart-phones, all of which have become a crucial part of our lives. In addition, each of these devices can run different operating systems, various browsers and apps, each of which may have their peculiarities for delivering the content – regardless of several standards existing. This plethora of devices, platforms and apps has introduced a problem how to guarantee usability and end-user satisfaction.

Usability as a quality attribute of a system is becoming increasingly important in developing and delivering new systems, and conquering market share. Usability, as first defined by Nielsen [1], applies to all aspects of a system with which a human might interact, and this diverse property of user interface is traditionally related to the following five attributes: learnability as the easiness to get something done rapidly, efficiency as the ability to productively use the system after having learned it, memorability as the capability for a user to return to system usage after a longer pause without having to re-learn it, low error rate for user-performed actions, and satisfaction addressing pleasant use of the system. ISO 9241-11 Guidance on Usability [2] 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 of these definitions are quite similar having the same characteristics for usability. Thereby, usability is not just only about pretty user interface (UI), and has become increasingly important with a crucial role on how end-users percept, experience, and accept the systems provided for them. If a user finds a website or app difficult to use, its structure confusing, is faced with problems finding necessary information at once or content hard to read, they usually leave the website or discontinue the use of app. The demand to produce usable web applications that are compatible platform-wise is increasingly important to gain and maintain end-user satisfaction. Thus, usability has become a key success factor for any software product.

Thereby, web and app user interfaces should be developed compatible with different platforms end-users might use, and keeping in mind that these must have intuitive navigation and structure, be smooth and easy to learn, and provide information in a clear and understandable way. To ensure this, aspects of usability should be addressed as early as possible in the development process of any UI, and compliance must be assured and supported throughout the whole process. Parts of this testing and validation work, could be automated, saving valuable development time and cost.

Reproducing, testing and validating the experience end-users get while viewing a website or using an app on different platforms, majorly dependent on the viewing options, screen size, and platform, but not only (comfort of device usage is also an important aspect) is laborious and costly, also in terms of human resources and time. If some of this work could be done automatically already in early development phase, using knowledge and best practices from web usability domain, then problems could be identified as they occur during development process and solutions provided immediately allowing to save time and cost on necessity to re-develop the UI. Also, this would economize time and cost of using human experts and testers wherever possible, and contribute to and stimulate production of high-quality user interfaces.

In this paper we address the problem how to effectively evaluate user interfaces' cross-platform compatibility and usability during production-time with minimal cost and time. This is achieved through establishment of a domain ontology that is a part of automatic usability evaluation tool addressing the gap between system usability final evaluation and its UI development stage by enabling immediate feedback during production. This paper is based on the framework we proposed in [3] with the focus set particularly on the ontology design for the task and tool developed. In terms of UIs, of particular interest are web user interfaces and apps.

The rest of the paper is organized as follows. In Section 2, we provide overview of related works, whereas in Section 3 we outline the usability evaluation framework. Section 4 discusses the ontology design to capture usability knowledge, and

978-1-890843-36-6 ©2017 PICMET

presents some examples of the preliminary tool. Section 5 draws conclusions and presents future works.

II. Related Works

Usability and user experience as a research branch has been contributed by many researchers by exploring solutions of automated testing [4], usability guidelines verification [5], enhancing usability assessing approaches [6], [7], or improving usability metrics and guidelines [8], [9], with the main outcome as improved collections of guidelines, UI quality evaluation methods, and solutions for verifying UI conformance to a set of guidelines.

The responsibility to deliver usable UI is on designers, whereas they can apply different guidelines and non-functional requirements that UIs should satisfy [10], [11] (e.g. the Web Content Accessibility Guidelines WCAG [1] 1.0 and 2.0), evaluation approaches such as heuristic evaluation [12], [13], [14], formal usability inspections [15], [16], pluralistic usability walkthrough [15], or empirical methods where direct feedback from users is available, such as card sorting [17], questionnaires [18], and interviews [19]. It is their job to verify that the designed UI satisfies all business requirements, predefined usability guidelines and its overall information architecture is clear for potential users.

The development of automated usability evaluation tools has been an attractive research area, as these tools provide an easy, fast, and cheap way to detect usability issues by reducing the costs of usability evaluation and increasing the coverage of UI presentation and delivery areas [5], [20]. Different solutions have been proposed with different sets of guidelines as input, including the WCAG [5], [17], [21], [22], mainly based only on parsing the HTML and validating its syntax, excluding any visual analysis of developed UI. A category of tools predicting the usage of UI based on knowledge discovery approach has been explored in [23] and [24]. Yusop and colleagues [25], [26] surveyed usability defect reporting practices in industrial software organizations and in open source communities, finding that the most needed information for developers was summary and cause of UI errors. Evaluation reports and their content have also been explored by Davies and Roper in [27].

Using ontologies to capture knowledge in software engineering (SE) domain for its reuse is not a new idea. In [28] authors explored development of SE ontology for multi-site software development, whereas their ontology consisted of 362 concepts and 303 relations implemented in Web Ontology Language OWL. In [29] Antunes and colleagues applied ontologies for software development knowledge reuse, whereas Seremeti et al. [30] for multimedia and Falbo et. al [31] for service modeling, with the aim to specify the terms in the domain and relations among them.

Ontologies have been successfully applied also in the research of UI and usability, as they allow efficiently to capture and store domain knowledge. For example, Fatima and colleagues [32] used ontology for merging positive user experience with an effective semantic search engine interface. In [33] Cysneiros, Werneck and Kushniruk claimed that there is a necessity for ontology (or classification) of measurable aspects of usability that could be used in the process of specifying usability requirements, and established a catalogue of reusable knowledge on usability requirements. Bakaev and Gaedke [34] used ontologies for semi-automated engineering of human-computer interfaces using evolutionary algorithms. Bacikova and Porubän [35] studied domain usability of user interfaces, emphasizing that the language used in the UI presented to end-users has to be domain specific in order those domain end-users could efficiently use it.

In terms of our previous research connected to the work presented herein, in [3] we established a framework for automatic usability evaluation, whereas in [36] we explored user behavior in UIs. Also, we have developed and applied ontologies for modelling web information systems and users and their behavior in them [37], [38].

III. Usability Evaluation Framework

A. *Motivation*

Throughout web or app UI development constant changes are made. Changes are made even after release. These changes to UI may lead to violation of preset usability rules for that particular UI defined by designers, or common usability guidelines. For instance, an error in some stylesheet for color code could lead the contrast between text and background color be almost nonexistent for a part of UI or only one of its elements, and it could even be missed in manual inspection. This, nevertheless would decrease accessibility for users (especially users with cognitive impairments) and lead to decrease of end-user satisfaction as usability is compromised. Many of these problems are addressed in usability guidelines and can be avoided by their effective use, either automated or manual.

Although empirical UI evaluation methods, including card sorting, questionnaires and interviews are effective in finding usability problems in system UIs, they suffer from weaknesses that prevent their extensive use: they are the most human resource and time consuming [39], it may not be possible to evaluate every single aspect of system UI using empirical methods [40], and also it might be difficult to get together necessary number of users for target test group [41]. This makes assessment of any mid-size or large system UI and its conformance to usability guidelines on different platforms a complex and resource-demanding task, as preparing platform-specific questionnaires, scenarios, finding target users, and feedback analysis takes time and resources. Thereby, providing solutions to UI problems found during such empirical analysis cannot be immediate, and instantly applied in production. This is the situation where we can successfully benefit from automated tools that can follow the best practices captured in usability guidelines, such as WCAG and Section 508[2], which requires conformance to WCAG 2.0 Level AA in addition to conformance to other guidelines defined in the standard. The aim of these sets of guidelines is to assure usable UIs for all end-users. However, WCAG and Section 508 do not cover all aspects of usability, and additional sources from research and

[1] http://www.w3.org/WAI/intro/wcag

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

active practice are also used, e.g. recommendations from the Nielsen Norman Group[3]. In addition, many companies adopt their own usability guidelines to secure coherent UI design between different apps and platforms they have, and afford users a pleasant user experience.

To verify UI against most of guidelines (e.g., W3C Web Content Accessibility Guidelines WCAG 1.0 and WCAG 2.0, Section 508) today does not require human evaluators any more, although each and every of them can be assessed manually. There are many tools (for a list of different available tools for web accessibility evaluation one may refer to W3C tools list[4] of more than 80 tools) to conduct the task (semi-)automatically and check whether the developed web UI meets accessibility guidelines. Still, tools based on WCAG and other common guidelines alike, do not consider corporate rules or specific rules set for a particular system UI. Not every violence of a guideline is to be dealt with (dependent on severity level), leading to a necessity to be able to customize what is being assessed. Moreover, these tools are to be used on ready-made or pre-release solutions and do not consider a possibility to be exploited in early phases of development for automatic UI evaluation and immediate feedback with inconsistency checking. The majority of tools for automatic evaluation are web-based using application URL (Uniform Resource Locator) as an input and providing a report highlighting deviations from guidelines as an output, and conduct the assessment only based on the accessibility guidelines by parsing the HTML code, and thereby ignoring executable scripts (e.g. JavaScript) and page events. Also, existing automated usability evaluation tools cannot be integrated into UI production development flow. Therefore, these tools are only suitable for evaluating pre-release or existing system UI, not a UI in production development, providing a motivation for our framework establishment and development of an evaluation tool. Our approach with the automatic usability evaluation framework differs from the latter as it is based on domain ontology, allows selection and addition of different guidelines and is especially meant for the development phase of system UI, enabling immediate feedback and possibility to fix problems before they could get into final (pre-release) version and manual testing.

B. Evaluation Framework and Preliminary Tool

The automatic usability evaluation framework targets to exploit usability knowledge captured in repositories in the form of ontology to evaluate a set of UI pages, in particular web pages. The concept is that for the development environment a special monitor hook can be set to invoke the evaluator tool ‘Guideliner’ on changes in UI to check the modified UI against a set of guidelines. A configuration file (default or user modified) is read with the start of the evaluator tool before initiating UI inspection. According to settings in the configuration file, guidelines described in the ontology repository are accessed and the usability knowledge in a machine-processable form in an ontology is read by the interpreter module. Special processing adaptors are used to access UI elements and assess each and every of them against selected UI guidelines and the metrics associated with them, as described in the ontology. Finally, the results are returned in the form of a report to developer. The concept of evaluation in the framework is on fetching the page and analyzing it with the help of processing adaptors. This approach reduces the effort needed for UI processing. Fig. 1 outlines the components of the framework, together with some third-party core components used to establish the evaluator tool ‘Guideliner’.

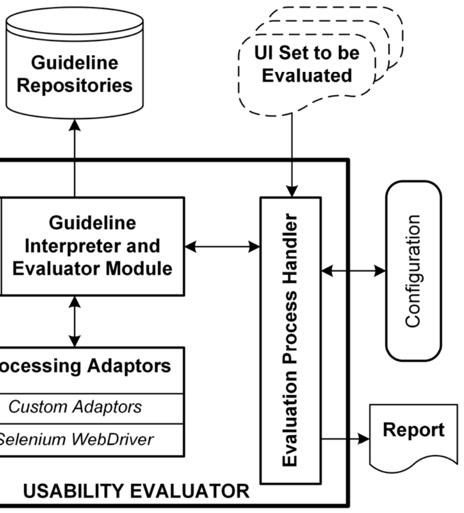


Fig. 1. Usability evaluation framework and setup of the ‘Guideliner’ tool

In the first preliminary prototype [3] the ontology repository was not present in the framework, instead we used a metalanguage to define guidelines, calling it usability guideline definition language (UGDL). The experiments with established preliminary tool were successful and encouraging. At time the test suite contained 117 guidelines combined from different usability areas with various evaluation metrics. However, as UGDL was limited in expressing the guidelines, we are now exchanging it with ontology descriptions in our framework. The ontology for the framework is discussed in more detail in Section 4.

The evaluator tool is developed in Java programming language and it takes advantage of two third-party components – the OWL API[5], which is a Java API and reference implementation for creating, manipulating and serializing ontologies described in OWL, and Selenium WebDriver API[6], which is a tool that provides API for automated user interface testing and allows automatic verification that UI works as expected. The Selenium WebDriver has support for most programming languages (e.g. Java, C#, Python, PHP, etc.) and development environments. It is compatible with various browsers such as Firefox, Internet Explorer, Chrome, Opera, and platforms like MS Windows, iOS, Android etc. The use of Selenium WebDriver is essential as it provides operations and commands for fetching the web page, locating UI elements on it, performing automatically actions with fetched UI elements (e.g. clicking), filling inputs, moving between windows and so

[3] https://www.nngroup.com
[4] https://www.w3.org/WAI/ER/tools/

[5] http://owlapi.sourceforge.net/
[6] http://www.seleniumhq.org/projects/webdriver

forth. Thus, it enables a full-stack of instruments and methods to automatically and programmatically manipulate and evaluate UI conformance to guidelines. In the framework, the Selenium is a part of the processing adaptors, together with custom adaptors which address specific aspects of UI being evaluated, for instance links, general, page layout, colors, etc.

There are however limitations concerned with the tool: it presently only has a support for HTML5 and CSS3 (with backwards compatibility) and Javascript based user interfaces. This means that the tool is unable to provide assessment for interfaces developed in Flash or as Java applets, or in any other technology, e.g. Microsoft Silverlight.

IV. Ontology Design for Usability Knowledge

A. *Why Ontology?*

Ontology is an explicit specification of conceptualization, it represents domain knowledge and makes it possible to reuse the latter [42], [43]. Ontologies are an important part of any intelligent system. They allow capturing domain knowledge in a human-understandable, yet in a machine-processable way, consisting of entities, attributes, relationships and axioms, and thereby enable formal presentation of knowledge as a set of concepts within a domain, and the relationships between those concepts, and enable reasoning about the entities within that domain.

A number of languages exist to describe knowledge in the form of ontology, e.g. Loom and Ontolingua, and Semantic Web languages, such as OIL, DAML+OIL, RDF Schema and W3C Web Ontology Language (OWL). Out of these, OWL is of interest for us, as a standard language for ontology description recommended by the W3C. OWL allows to capture knowledge by representing the concepts and relationships among concepts, and axioms that formalize the definitions and relations for a given domain. The components of OWL ontologies are classes (sets containing individuals), properties (binary relations linking two individuals together) and individuals (instances). The most important relations between concepts are the *is-a* relation, which defines the class – sub-class hierarchy, and the *part-of* relation, relating 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. Consequently, OWL is an advanced language that can be used in creating intelligent systems, which is also the aim of our work.

As already previously noted, we were dissatisfied with the expressive possibilities of the UGDL, and thereby were looking for something more expressive than just a simple metalanguage. For the reasons outlined above, the use of an ontology would provide powerful means to capture usability domain knowledge, in particular different sets of guidelines, and would allow automatic classification and reasoning over the captured concepts. Thereby, the use of ontology instead of a UGDL provides a good way to uniformly describe and store usability domain knowledge through various aspects, being both human- and machine-readable, and deliver a shared vocabulary describing the type of concepts that exist in the domain, and their properties (metrics) and relations. Altogether, the exploitation of ontology allows us to turn the framework and our software tool more flexible, adaptive and intelligent, and re-use domain knowledge whenever needed.

B. *Building the Ontology*

The knowledge captured in the ontology is mainly based on the guidelines described in WCAG 1.0, 2.0 and Section 508, Research-Based Web Design and Usability Guidelines from U.S. Dept. of Health and Human Services [44], and supplemented with recommendations from the Nielsen Norman Group. As each of these addresses specific aspects of usability and none of the sources is comprehensive – e.g., WCAG and Section 508 mainly address accessibility not covering aspects like navigation and screen controls – designing and setting up a custom repository of usability guidelines together with metrics that are machine-processable in the form of an ontology is a perfect solution within our framework of guidelines evaluation.

To capture the knowledge of the usability domain we used OWL (in particular OWL-DL – Description Logic variance of OWL) version 2 as a language, mainly to gain support for automatic reasoning and classification.

Ontologies are usually composed using special software – ontology editors. We constructed our ontology manually using the Protégé ontology editor[7] version 4.3 for the task. Protégé is a free open-source platform providing a suite of tools to construct domain models and knowledge-based applications with ontologies. In principle, there are no limitations on editors to be used.

To check consistency of the ontology and run classification tasks, we used reasoners. Reasoners, also known as classifiers, infer logical consequences from a set of asserted facts or axioms, and also check the consistency (whether a class can have any instances or not based on its description) of an ontology. They are also used to compute inferred ontology class hierarchy based on the concept definitions. Thereby, the use of reasoners guarantees consistent and coherent hierarchy in an ontology. The Protégé ontology editor allows easy integration with many reasoner tools, for example the Pellet, Fact++, and HermIT reasoners, out of which we used the latter for the research herein.

In terms of the ontology design, our ontology is not focused to capture everything from the usability domain, but to describe the usability guidelines of interest for automated evaluation of web UIs. There are five main concepts defined as primitive classes (they have only necessary conditions defined) in our ontology (Fig. 2): *Guideline*, *GuidelineElement*, *ContentType*, *ElementAttribute* and *ValuePartition*, which is a special class used to refine guideline descriptions through a pre-defined set of value concepts. Presently three types of value partitions have been described: the importance and strength of evidence of a particular guideline on the scale of 1–5 (added to the guideline description over the *hasRelativeImportance* and *hasStrengthOfEvidence* object properties accordingly in the ontology), and different units used for UI elements such as point, pixel, em, etc. Fig. 2 outlines a small selection of the

[7] http://protege.stanford.edu/

ontology showing the five main concepts and their descendant classes over the *is-a* relationship. The arrows at the end of the concept (ellipse) denote that the concept is a superclass for at least one another concept in the ontology.

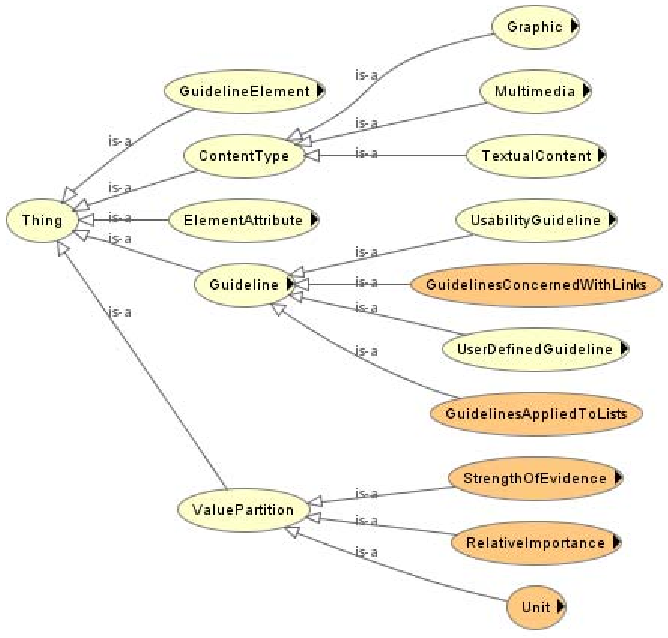


Fig. 2. Outline of the established usability ontology and its main concepts together with some selected classes: excerpt from the Protégé ontology editor

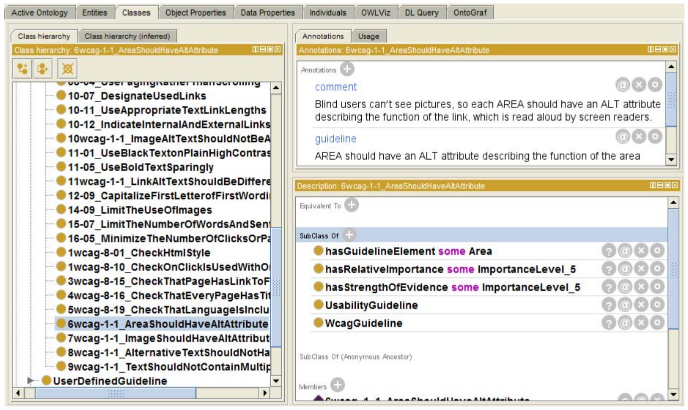
The main concept of the ontology is the class named as *Guideline*, under which two subclasses *UsabilityGuideline* and *UserDefinedGuideline* are located. The *Guideline* class also holds defined classes that are used to infer new subclasses (e.g. *GuidelinesConcernedWithLink* – a defined class that 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 should be defined in the primitive class *UsabilityGuideline* or *UserDefinedGuideline* as subclasses. Each guideline is linked to the *GuidelineElement* class via the *hasGuidelineElement* object property. Also, each guideline is annotated using the following annotation properties:

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

The *GuidelineElement* class holds subclasses describing elements a guideline may be applied to (e.g. form, frame, link, etc.). This taxonomy of subclasses reflects HTML tags or their equivalents and their related structure. The *ElementAttribute* class describes the attributes evaluated elements may contain. Not every attribute is present at each element, and an element may have several attributes. Attributes may be enumerated classes (e.g., target: {_blank, _parent, _self, _top}). The *GuidelineElement* class is connected to *ElementAttribute* via the *hasAttribute* object property. The *ContentType* class is used as a classifier for the *GuidelineElement* over *hasContentType* object property.

The ontology also contains several data properties to hold particular values elements and attributes connected to some guideline, e.g. data property *hasUnit*, *hasMinContrastValue*, *hasMaxContrastValue*, etc.

Fig. 3 shows an example of a guideline concept description in the Protégé ontology editor for the WCAG 1.1 guideline stating that areas should have an ALT attribute. The guideline's general description is added as the 'guideline' annotation, whereas the annotation 'comment' provides further details (Fig. 3 upper right corner), and 'reference' (in the annotation box but not visible on the figure) provides URL for the particular guideline.



Class hierarchy: 6wcag-1-1_AreaShouldHaveAltAttribute

10-07_DesignateUsedLinks
10-11_UseAppropriateTextLinkLengths
10-12_IndicateInternalAndExternalLinks
10wcag-1-1_ImageAltTextShouldNotBeA
11-01_UseBlackTextonPlainHighContras
11-05_UseBoldTextSparingly
11wcag-1-1_LinkAltTextShouldBeDiffere
12-09_CapitalizeFirstLetterofFirstWordi
14-09_LimitTheUseOfImages
15-07_LimitTheNumberOfWordsAndSen
16-05_MinimizeTheNumberOfClicksOrPa
1wcag-8-01_CheckHtmlStyle
1wcag-8-10_CheckOnClickIsUsedWithO
3wcag-8-15_CheckThatPageHasLinkToF
4wcag-8-16_CheckThatEveryPageHasTit
5wcag-8-19_CheckThatLanguageIsInclu
6wcag-1-1_AreaShouldHaveAltAttribute
7wcag-1-1_ImageShouldHaveAltAttribut
8wcag-1-1_AlternativeTextShouldNotHa
9wcag-1-1_TextShouldNotContainMultip
UserDefinedGuideline

Annotations: 6wcag-1-1_AreaShouldHaveAltAttribute

comment
Blind users can't see pictures, so each AREA should have an ALT attribute describing the function of the link, which is read aloud by screen readers.

guideline
AREA should have an ALT attribute describing the function of the area

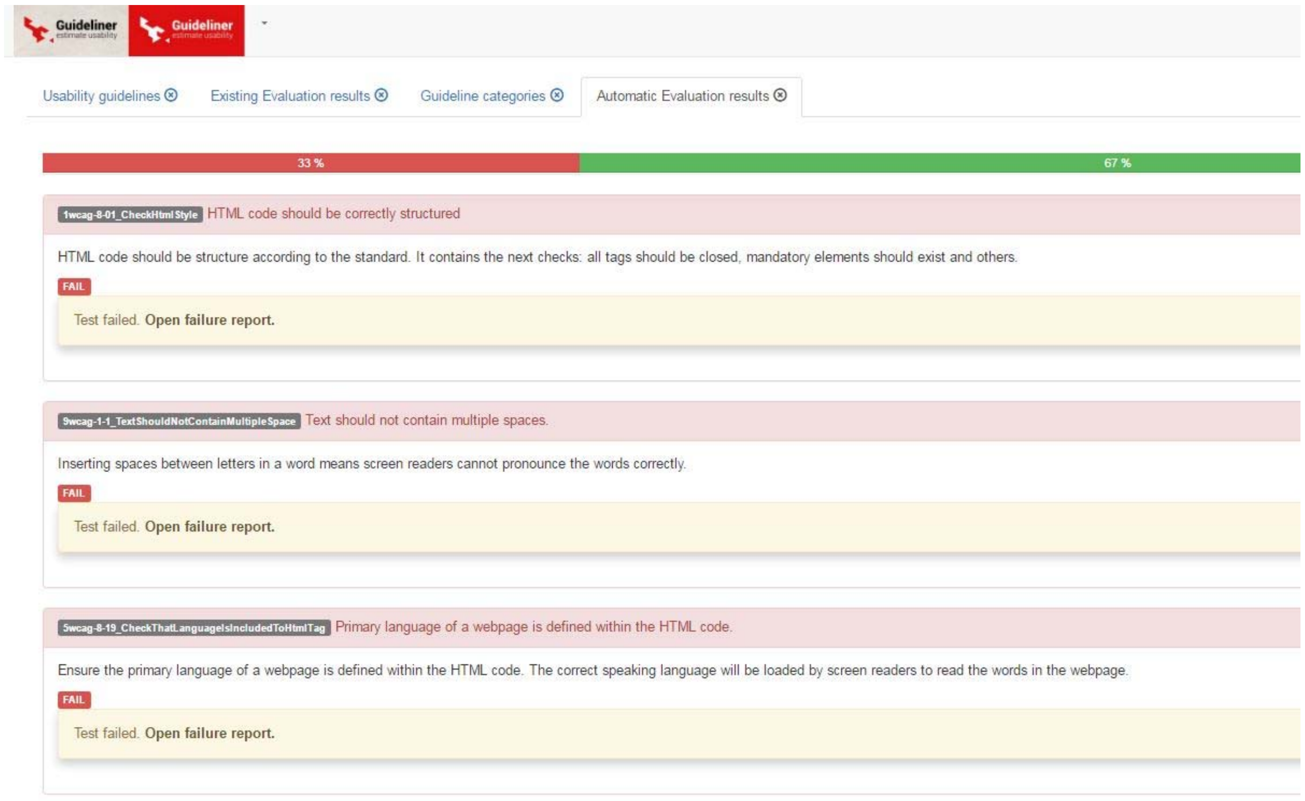
Description: 6wcag-1-1_AreaShouldHaveAltAttribute

SubClass Of
hasGuidelineElement some Area
hasRelativeImportance some ImportanceLevel_5
hasStrengthOfEvidence some ImportanceLevel_5
UsabilityGuideline
WcagGuideline

Fig. 3. Description of a guideline in the ontology (screenshot from the Protégé ontology editor)

In order to validate the ontology design and prove its applicability within the framework, we chose for the first development iteration of the ontology 115 guidelines from different categories as outlined in Table 1. This set of guidelines is more than enough to test and validate the ontology as well as the framework and the evaluator tool 'Guideliner' established according to it. New guidelines can always be added to the ontology, however, this is a laborious task and each new guideline adds complexity to the ontology. Therefore, the use of reasoner becomes essential in ontology management.

The ontology in action within the 'Guideliner' tool can be seen on Fig. 4, showing an example where the tool was run for the e-Estonia.com state portal. The tool has found some problems according to described guidelines regarding HTML style, multiple spaces in text that might disturb the effective use of screen readers, and the primary language definition is missing.



1wcag-8-01_CheckHtmlStyle HTML code should be correctly structured

HTML code should be structure according to the standard. It contains the next checks: all tags should be closed, mandatory elements should exist and others.

FAIL

Test failed. **Open failure report.**

9wcag-1-1_TextShouldNotContainMultipleSpace Text should not contain multiple spaces.

Inserting spaces between letters in a word means screen readers cannot pronounce the words correctly.

FAIL

Test failed. **Open failure report.**

5wcag-8-19_CheckThatLanguageIsIncludedToHtmlTag Primary language of a webpage is defined within the HTML code.

Ensure the primary language of a webpage is defined within the HTML code. The correct speaking language will be loaded by screen readers to read the words in the webpage.

FAIL

Test failed. **Open failure report.**

Fig. 4. The ‘Guideliner’ tool has identified some shortcomings in the e-Estonia.com state portal

TABLE I. GUIDELINES IN THE ESTABLISHED ONTOLOGY

Category	Guidelines covered
Compatibility and accessibility guidelines	15
Content organization, layout, scrolling and paging	35
Navigation	11
Graphichs, images and multimedia	6
Screen-based controls	12
Page elements (e.g. headings, links, text appearance)	28
Information search	8
TOTAL	115

V. CONCLUSIONS

Usability, which is not only about pretty user interface, is becoming increasingly important and plays a crucial role how end-users percept, experience, and accept the systems provided for them. Websites and apps that are difficult to use produce unsatisfied end-users, who soon stop visiting website or discontinue app usage. Therefore, it is essential to pay attention to usability, following the accessibility guidelines, best practices and ensuring that users are able to effectively and efficiently use system UI that is clear, easily memorable and understandable to them, and does not allow them to make mistakes.

Usability of a system user interface can be secured in several ways, traditionally using empirical UI evaluation methods, including card sorting, questionnaires and interviews – all which are effective in finding usability problems in system UIs. Yet, this is prone to human factor, takes time and needs plenty of human resources. Several accessibility and usability guidelines exist, that are used in the assessment process. Although all of them can be applied manually, there is a vast majority of things that can be computerized and applied automatically. Several standalone tools exist that are able to check UI conformance to guidelines such as WCAG and Section 508, however they are not applicable to be used in-production starting from early development phases. This is the gap we are trying to minimize with our framework – allow in-production immediate feedback on user interface changes according to a set of guidelines defined for the UI project.

In this paper we have outlined the motivation and basic structure of the framework, and focused our effort on the need and development of the ontology to capture usability domain knowledge for automated UI assessment. The first results with 115 guidelines captured in the ontology are encouraging, and the Information Systems Authority of the Republic of Estonia has already shown an interest to take advantage of the project in the process of testing and validating the new release of state portal for citizens and entrepreneurs.

ACKNOWLEDGMENT

This research was partially supported by the Estonian Ministry of Research and Education institutional research grant no. IUT33-13 “Methods, tools and processes for software with strong guarantees”. Authors would like to acknowledge the cooperation of Estonian Information Systems Authority state portal team.

REFERENCES

[1] J. Nielsen, Usability engineering, Academic press, Boston, 1994.

[2] ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) -- Part 11: Guidance on usability, WWW: http://www.iso.org/iso/catalogue_detail.htm?csnumber=16883

[3] J. Marenkov, T. Robal, and A. Kalja, "A study on immediate automatic usability evaluation of web application user interfaces," Databases and Information Systems: 12th International Baltic Conference, DB&IS 2016, Ed. Arnicans, G., et al. Cham: Springer, Communications in Computer and Information Science; 615, 2016, pp. 257−271.

[4] J. Kaasila, D. Ferreira, V. Kostakos, and T. Ojala, "Testdroid: automated remote UI testing on Android," Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia, art. 28. ACM, New York, 2012, pp. 28:1--28:4.

[5] A. Dingli, "USEFul: A framework to mainstream web site usability," International Journal of Human Computer Interaction, 2011, pp. 10-30.

[6] W. Wetzlinger, D. Nedbal, A. Auinger, C. Grossauer, C. Holzmann, and F. Lettner, "Mobile usability testing requirements and their implementation in the automation engineering industry," in Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia (MoMM '14), ACM, New York, 2014, pp. 62-71.

[7] T. Porat, A. Schclar, and B. Shapira, "MATE: a mobile analysis tool for usability experts," in CHI '13 Extended Abstracts on Human Factors in Computing Systems, ACM, New York, 2013, pp. 265-270.

[8] 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, ACM, New York, 2008, pp. 567-570.

[9] R. van Solingen, and E. Berghout, The Goal/Question/Metric method: a practical guide for quality improvement of software development. McGraw Hill, 1999.

[10] A. M. Moreno, A. Seffah, and R. Capilla, "HCI practices for building usable software," Computer, 46, 2013, pp. 100-102.

[11] J. A. Borges, I. Morales, and N. J. Rodríguez, "Guidelines for designing usable World Wide Web pages," in Conference Companion on Human Factors in Computing Systems, ACM, New York, 1996, pp. 277-278.

[12] J. Nielsen, and R. Molich, R. "Heuristic evaluation of user interfaces," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, New York, 1990, pp. 249-256.

[13] H. Yehuda, and J. McGinn, "Coming to terms: comparing and combining the results of multiple evaluators performing heuristic evaluation," in CHI '07 Extended Abstracts on Human Factors in Computing Systems, ACM, New York, 2007, pp. 1899-1904.

[14] M. Eksioglu, E. Kiris, B. Capar, M. N. Selcuk, and S. Ouzeir, "Heuristic evaluation and usability testing: case study," in Proceedings of the 4th international conference on Internationalization, design and global development, Springer-Verlag, Berlin, 2011, pp. 143-151.

[15] T. Hollingsed, and D. G. Novick, "Usability inspection methods after 15 years of research and practice," in Proceedings of the 25th annual ACM international conference on Design of communication, ACM, New York, 2007, pp. 249-255.

[16] A. Fernandez, S. Abrahao, and E. Insfran, "Empirical validation of a usability inspection method for model-driven Web development.," Journal of Systems and Software, 86, 2013, pp. 161-186.

[17] B. Leporini, F. Paterno, and A. Scorcia, "Flexible tool support for accessibility evaluation," Interacting with Computers, 18, 2006, pp. 869-890.

[18] 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, ACM, New York, 2008, pp. 53-62.

[19] L. Kantner, D. H. Sova, and S. Rosenbaum, "Alternative methods for field usability research," in Proceedings of the 21st annual international conference on Documentation, ACM, New York, 2003, pp. 68-72.

[20] A. Dingli, and S. Cassar, "An intelligent framework for website usability," Advances in Human-Computer Interaction, Article 5, 2014.

[21] A. Aizpurua, M. Arrue, M. Vigo, and J. Abascal, "Transition of accessibility evaluation tools to new standards," in Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibililty, ACM, New York, 2009, pp. 36-44.

[22] B. Leporini, F. Paterno, and A. Scorcia, "An environment for defining and handling guidelines for the web," in Proceedings of 10th International Conference on Computers Helping People with Special Needs, Springer, Berlin, 2006, pp. 176-183.

[23] P. A. Davis, and F. M. Shipman, "Learning usability assessment models for web sites," in Proceedings of the 16th international conference on Intelligent user interfaces, ACM, New York, 2011, pp. 195-204.

[24] B. C. Boza, S. Schiaffino, A. Teyseyre, and D. Godoy, "An approach for knowledge discovery in a web usability context," in Proc. of the 13th Brazilian Symposium on Human Factors in Computing Systems, Sociedade Brasileira de Computação, Porto Alegre, 2014, pp 393 – 396.

[25] N. S. M. Y. 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, ACM, New York, 2016, pp. 1-10.

[26] N. S. M. Yusop, J. Grundy, and R. Vasa, "Reporting usability defects: limitations of open source defect repositories and suggestions for improvement," in Proceedings of ASWEC Australasian Software Engineering Conference, ACM, New York, 2015, pp. 38 – 43.

[27] S. Davies, and M. Roper, "What's in a bug report?," in Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ACM, Torino, 2014, p. 26.

[28] P. Wongthongtham, E. Chang, T. Dillon, and I. Sommerville, "Development of a software engineering ontology for multisite software development," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 8, August, 2009, pp. 1205-1217.

[29] B. Antunes, N. Seco, and P. Gomes, "Using ontologies for software development knowledge reuse," 13th Portuguese Conference on Aritficial Intelligence, EPIA 2007, Workshops: GAIW, AIASTS, ALEA, AMITA, BAOSW, BI, CMBSB, IROBOT, MASTA, STCS, and TEMA, Guimarães, Portugal, December 3-7, 2007, LNCS 4874, Springer Berlin Heidelberg, 2007, pp. 357-368.

[30] L. Seremeti, and A. Kameas, "Multimedia ontologies," in Proceedings of the 3rd international conference on Mobile multimedia communications, ICST, Brussels (2007), p. 69.

[31] R. A. Falbo, G. K. Quirino, J. C. Nardi, M. P. Barcellos, G. Guizzardi, N. Guarino, A. Longo, and B. Livieri B, "An ontology pattern language for service modeling," in Proceedings of the 31st Annual ACM Symposium on Applied Computing, ACM, New York, 2016, pp. 321 – 326.

[32] A. Fatima, C. Luca, and G. Wilson, "User experience and efficiency for semantic search engine," 2014 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM), Bran, 2014, pp. 924-929.

[33] L. M. Cysneiros, V. M. Werneck, and A. Kushniruk, "Reusable knowledge for satisficing usability requirements," in Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE '05). IEEE Computer Society, Washington, DC, USA, 2005, pp. 463-464.

[34] M. Bakaev, and M. Gaedke, "Application of evolutionary algorithms in interaction design: From requirements and ontology to optimized web interface," 2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW), 2016, pp. 129-134.

[35] M. Bačíková, and J. Porubän, "Ergonomic vs. domain usability of user interfaces," 6th International Conference on Human System Interactions (HSI), Sopot, 2013, pp. 159-166.

[36] J. Marenkov, T.Robal, and A. Kalja, "A study on user click behaviour for WIS user interface improvements," Databases and Information Systems VIII: Selected Papers from the 11th International Baltic Conference, Baltic DB&IS 2014, Amsterdam: IOS Press (Frontiers in Artificial Intelligence and Applications; 270), 2014, pp. 173−186.

[37] T. Robal, and A. Kalja, "Managing knowledge in web portals for improved customer loyalty and satisfaction," PICMET '13: Proceedings, Technology Management in the IT-Driven Services, Ed. D. Kocaoglu, et al., Portland, Oregon, USA: PICMET, 2013, pp.1207−1216.

[38] T. Robal, and A. Kalja, “Applying user domain model to improve Web recommendations,” Databases and Information Systems VII: Selected Papers from the Tenth International Baltic Conference, DB&IS 2012. Ed. A. Caplinskas, G. Dzemyda, A. Lupeikiene, O. Vasilecas, Amsterdam: IOS Press (Frontiers in Artificial Intelligence and Applications; 249), 2013, pp. 118−131.

[39] 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, ACM, New York, 2008, pp 23-32.

[40] M. Y. Ivory, and M. Hearst, “The state of the art in automating usability evaluation of user interfaces,” ACM Computing Surveys (CSUR): Vol. 33(4), 2001, pp. 470-516.

[41] A. Lecerof, and F. Paterno, “Automatic support for usability evaluation,” IEEE Transactions on Software Engineering, vol. 24, 1998, pp. 863-888.

[42] T. Gruber, “Toward Principles for the Design of ontologies used for knowledge sharing,” Int. Journal on Human and Computer Studies, vol. 43(5/6), pp. 907–929, November 1995.

[43] N. Guarino, “Formal Ontology in Information Systems,” Proc. First International Conference on Formal Ontology in Information Systems (FOIS´98), IOS Press, 1998, pp. 3-15.

[44] Research-Based Web Design and Usability Guidelines, U.S. Dept. of Health and Human Services, 2006.