Combining multiple Web Accessibility Evaluation Reports using Semantic Web Technologies

José R. Hilera *jose.hilera@uah.es*
University of Alcalá
Alcalá de Henares, Spain

Salvador Otón Tortosa *salvador.oton@uah.es*
University of Alcalá
Alcalá de Henares, Spain

Cristian Timbi-Sisalima *ctimbi@ups.edu.ec*
Universidad Politécnica Salesiana
Cuenca, Ecuador

Juan Aguado-Delgado *j.aguado@edu.uah.es*
University of Alcalá
Alcalá de Henares, Spain

Francisco J. Estrada-Martínez *francisco.estrada@edu.uah.es*
University of Alcalá
Alcalá de Henares, Spain

Héctor R. Amado-Salvatierra *hr_amado@galileo.edu*
Galileo University
Guatemala City, Guatemala

Abstract

This paper describes a process for automatic combination of testing reports for the accessibility of Web applications, obtained by different testing tools and applying different standards on Web accessibility. Interoperability is guaranteed using semantic Web technologies, which allow describing the reports by RDF (Resource Description Framework) triples. The reports refer to elements of a knowledge base consisting of vocabularies, ontologies and rules of inference, in which the conceptual relations between accessibility standards, as WCAG (Web Content Accessibility Guidelines) or Section 508 among others, are formalized previously. A software prototype that uses the Apache Jena framework for implementing the process is presented.
Keywords: Interoperability, Semantic Web, Software Testing, Web Accessibility.

1. Introduction

The accessibility of a Web application is essential to make it understandable, usable and practical for all users, including disabled people. The evaluation of the accessibility of a website must be done checking the compliance of accessibility requirements established by well-known specifications, standards or laws. The most used standard is WCAG (Web Content Accessibility Guidelines) [20], created by the World Wide Web Consortium (W3C) and adopted as a standard by the International Organization for Standardization (ISO) [8]. Another important set of accessibility requirements are the ones included in the American Law known as Section 508 [21].

When the accessibility of a website is analyzed, the results are collected in an evaluation report, which describes Web pages reviewed, assessment rules applied, automatic evaluation tools used, etc.

It is usual that the results of the assessment depend on the evaluators involved and the testing tools used, obtaining different reports for the same Web page. For instance, an evaluator could use an automated tool that checks that all page images have alternative text, which is necessary so that screen readers can read the text to visually impaired users who cannot see the images, while another evaluator could use a more intelligent tool which also verifies the content of the alternative text makes sense. The first report would appear that everything is ok, while the second could indicate that an image has associated an incorrect alternative text. It might also happen that accessibility standard applied in a report is more demanding than the other one. For example, this occurs with flickering images on the screen, since the requirements of American legislation are more stringent than those of the WCAG standard.

In this context, it is necessary to implement mechanisms that allow interoperability between accessibility assessment tools. The Semantic Web technologies can be very suitable to solve the problem. The term "Semantic Web" refers to W3C's vision of the Web of linked data. Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data [13].

This article describes the process shown in Figure 1 to combine Web accessibility evaluation reports. The starting point are reports of accessibility expressed using Semantic Web vocabularies. It should also have a knowledge base containing information on accessibility standards and relations between them. From this, it uses the open source Semantic Web framework Apache Jena, which allows the application of reasoners to generate new knowledge. Finally, a query language created by the W3C for the Semantic Web is used, in order to obtain the final result of the evaluation.

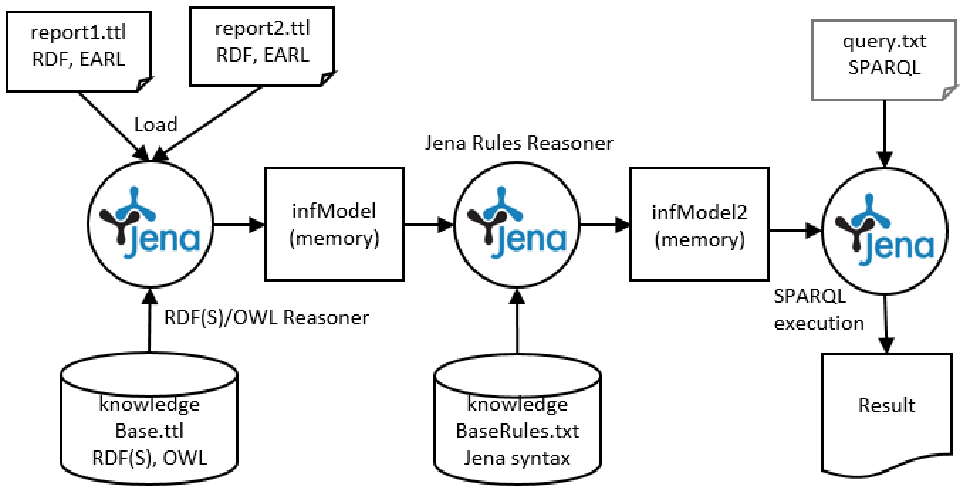

Fig. 1. Process of combining accessibility reports.

The following section describes the Semantic Web technologies used. In Section 3, the structure of accessibility reports according W3C recommendations is presented. Section 4 describes the steps for combining reports using the Semantic Web technologies. In the final section, some conclusions and other related works are presented.

2. Semantic Web Technologies

In this paper we have used the following technologies that are part of the Semantic Web [12]:

- **RDF(S)**: RDF is the original core technology of the Semantic Web, named Resource Description Framework, for describing information resources in the form of triples with the following structure, using Turtle notation [10]:

```
Subject Predicate Object .
```

An example of a RDF triple may be the following, extracted from DBPedia.org, which is a RDF representation of the Wikipedia content. It indicates that ACM (`Subject`) has a headquarter (`Predicate`) in New York (`Object`):

```
dbr:Association_for_Computing_Machinery
dbo:headquarter dbr:New_York_City .
```

For simplicity, we have used prefixes for each element of the triple. These prefixes identify the vocabulary in which defined the meaning of each element, which will be associated with an IRI (Internationalized Resource Identifier). Prefix `dbo` refers to `http://dbpedia.org/ontology/`, where it is defined what is meant by `headquarter`.

Coding 1 shows an excerpt of a RDF triples files used in this work. For simplicity, Turtle notation allows combine two or more triples with the same subject (s) in a unique sentence using ";". And triples with the same subject (s) and predicate (p) using ",". Another simplification is the predicate `rdf:type`, that is expressed only as "`a`".

Coding 1. Extract of the RDF(S)/OWL knowledge base (*knowledgeBase.ttl*).

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix s508: <http://example.org/section508.owl#> .
@prefix wcag2: <http://www.AccessibleOntology.com/WCAG2.owl#> .

#####################################################
# Knowledge about the US Law Section 508

s508:req_1194_22_a a s508:Requirement;
s508:hasDescription "1194.22(a) A text equivalent for every non-
text element shall be provided"@en .

s508:req_1194_22_j  a s508:Requirement;
s508:hasDescription "1194.22(j) Pages shall be designed to avoid
causing the screen to flicker with a frequency greater than 2 Hz
and lower than 55 Hz."@en .

#####################################################
# Knowledge about the standard WCAG 2.0

wcag2: SuccessCriterion_111 a wcag2:SuccessCriterion;
wcag2:hasDescription "Non-text Content: All non-text content that
is presented to the user has a text alternative that serves the
equivalent purpose, except for the situations listed
below."^^xsd:string .

wcag2:SuccessCriterion_231
    a wcag2:SuccessCriterion;
    wcag2:hasDescription "Three Flashes or Below Threshold: Web
pages do not contain anything that flashes more than three times in
any one second period, or the flash is below the general flash and
red flash thresholds."^^xsd:string .

#####################################################
# Knowledge about relations between WCAG 2.0 and Section 508

s508:req_1194_22_a owl:sameAs wcag2:SuccessCriterion_111 .
wcag2:SuccessCriterion_111 owl:sameAs s508:req_1194_22_a .
```

The ability to reference vocabularies allows a semantic extension of RDF. For this purpose, other associated technology is RDF Schema (RDFS), a vocabulary for data modeling [11]:

- **OWL (Ontology Web Language)**: An RDF triple store is a knowledge base. In this context, OWL can be understood as an RDF vocabulary based on descriptions logics that can be used in triples. The goal is that a reasoner can subsequently make inferences about them, generating new triples automatically and obtaining therefore new knowledge [9]. Coding 1 shows a knowledge base which will be explained in the next section, where there is a predicate from the OWL vocabulary, it is `owl:sameAs`.
- **Rules**: When in a knowledge base in the form of RDF triples, we want to add rules such as "if / then", it is not sufficient to use OWL, we need a language to express rules. For this, some languages have been created, as RuleML or SWRL (Semantic Web Rule Language), still under discussion by the W3C [15]. But there are other languages defined for this purpose, as the one created by the Apache Jena project [11], which has been used in the example of Coding 2.
- **Reasoners**: These are programs that can generate new RDF triples in a knowledge base from existing triples described with OWL, and from rules. Some of the best known are Pellet, Racer or FaCT [12].
- **SPARQL (SPARQL Protocol and RDF Query Language)**: To consult on an RDF triple store, the SPARQL language [14] is used. Coding 10 shows an example of query.

Coding 2. Extract of the rules base (*knowledgebaserules.txt*).

```
@prefix earl: <http://www.w3.org/nss/earl#>.
@prefix s508: <http://example.org/section508.owl#>.
@prefix wcag2: <http://www.AccessibleOntology.com/WCAG2.owl#>.

[rule_epilepsy:
  (?aser earl:test wcag2:SuccessCriterion_231),
  (?aser earl:result ?res),
  (?res earl:outcome earl:failed) -> (?aser earl:test
s508:req_1194_22_j)]
```

2.1. Knowledge base with OWL and rules

The process of combining reports of the same website shown in Figure 1 requires the existence of a knowledge base with the description of the accessibility standards or laws that apply when evaluating the website. Coding 1 shows an extract of the created knowledge base. It contains RDF triples, where it can be distinguished a first part in which the American Law on accessibility (known as Section 508) is described. For simplicity only appears the description of two articles of the law. The first is the one that requires that all images in a page must have alternative text, so that screen readers used by visually impaired people can read the description aloud. The second article included is the one that sets the permissible range of flashing for the contents of a page, to avoid problems of epilepsy.

The second part of the code of Coding 1 shows the description of the WCAG 2.0 accessibility standard created by the W3C, which is the one legally in force in many countries. Only the description of two success criteria established by this standard are shown, the first one refers to the need for alternate text and the second one is about flashing.

The last part of Coding 1 describes the relationship between the two accessibility standards. It only includes the relationship that indicates that the requirement of both standards for alternative text is the same. For this the predicate `sameAs` defined by OWL [8] is used.

In the case of epilepsy requirement, it is different in both standards, as in the US law the minimum flashing allowed is 2 Hz, while in the WCAG standard is 3 Hz. So we can say that if a Web page violates the WCAG standard, also violates the American law. Otherwise not be true. This cannot be expressed with OWL, but we can use a rule language to indicate it. In this case we have used the language provided by the Apache Jena [11] framework, and the code of the rule can be seen in Coding 2.

3. Web accessibility evaluation reports using EARL

When evaluating the accessibility of a page or website, the results are presented in a report. This report may contain information obtained through automatic evaluation tools as described in [19], combined with results achieved by manual testing or with the help of end users, including disabled people. The W3C has created the EARL vocabulary to express this kind of reports using RDF triples [5]. Coding 3 shows an excerpt of an evaluation report of a hypothetical Web page as `http://www.example.org/page.html`, applying the WCAG 2.0 standard and using as support the automatic evaluation tool named `OAW`, whose URL is in the report itself. Some authors of this paper have participated in the development of OAW [16], and for this reason it is used in this example.

Coding 3. Extract of a first EARL accessibility report.

```
@prefix earl: <http://www.w3.org/nss/earl#> .
@prefix ptr: <http://www.w3.org/2009/pointers#>.
@prefix doap: <http://usefulinc.com/ns/doap#> .
@prefix a11y: <http://example.org/a11yResources.owl#> .
@prefix wcag2: <http://www.AccessibleOntology.com/WCAG2.owl#>.
@prefix ex: <http://www.example.org#> .

##########################################################
# Definition of the accessibility evaluation tool used: OAW

a11y:OAW a earl:Software;
   doap:name "Analizador Web";
   doap:homepage <http://observatorioweb.ups.edu.ec/oaw/> .

##########################################################
# Failure 1 about Success Criterion 1.1.1 of ISO WCAG 2.0 in line
37

ex:assertion_OAW_1 a earl:Assertion ;
  earl:assertedBy a11y:OAW ;
  earl:subject <http://www.example.org/page.html> ;
  earl:test wcag2:SuccessCriterion_111;
  earl:result ex:OAWResult_1 .

wcag2:SuccessCriterion_111 a earl:TestRequirement .

ex:OAWResult_1 a earl:TestResult;
  earl:outcome earl:failed;  earl:pointer ex:ptr_OAWResult_1 .

ex:ptr_OAWResult_1 a earl:Pointer, ptr:LineCharPointer;
  ptr:lineNumber  "37"; ptr:charNumber  "8" .

##########################################################
# Failure 2 about Success Criterion 2.3.1 of ISO WCAG 2.0 in line
56

ex:assertion_OAW a earl:Assertion ;
  earl:assertedBy a11y:OAW ;
  earl:subject <http://www.example.org/page.html> ;
  earl:test wcag2:SuccessCriterion_231;
  earl:result ex:OAWResult .

wcag2:SuccessCriterion_231 a earl:TestRequirement .
```

```
ex:OAWResult_2 a earl:TestResult;
  earl:outcome earl:failed;  earl:pointer ex:ptr_OAWResult_2 .

ex:ptr_OAWResult_2 a earl:Pointer, ptr:LineCharPointer;
  ptr:lineNumber  "56"; ptr:charNumber  "6" .
```

Coding 4 shows the contents of another evaluation report about the same Web page, but using another tool called `AChecker`, and applying as accessibility standard the American Law Section 508. This tools is used in this example because is one of the few tools that allow the user to get the report in EARL format [17].

Coding 4. Extract of a second EARL accessibility report.

```
@prefix earl: <http://www.w3.org/nss/earl#> .
@prefix ptr: <http://www.w3.org/2009/pointers#>.
@prefix doap: <http://usefulinc.com/ns/doap#> .
@prefix a11y: <http://example.org/a11yResources.owl#> .
@prefix s508: <http://example.org/section508.owl#> .
@prefix ex: <http://www.example.org#> .

##########################################################
# Definition of the accessibility evaluation tool used: AChecker

a11y:AChecker a earl:Software;
   doap:name "AChecker: Web Accessibiilty Checker";
   doap:homepage <http://achecker.ca> .

##########################################################
# Failure about article 1192.22(a) of US Law “Section 508” in line
25

ex:assertion_AChecker a earl:Assertion ;
  earl:assertedBy a11y:AChecker ;
  earl:subject <http://www.example.org/page.html> ;
  earl:test s508:req_1194_22_a;
  earl:result ex:ACheckerResult .

s508:req_1194_22_a a earl:TestRequirement .

ex:ACheckerResult a earl:TestResult;
  earl:outcome earl:failed;  earl:pointer ex:ptr1_ACheckerResult .

ex:ptr1_ACheckerResult a earl:Pointer, ptr:LineCharPointer;
  ptr:lineNumber  "25"; ptr:charNumber  "10"
```

4. Combining multiple reports

We have developed a software prototype for combining multiple accessibility evaluation reports of the same website. Considering that in the reports different assessment tools have been used, and different standards or accessibility laws have been applied. For this, we have used the library Apache Jena for Java, an open source Semantic Web framework that provides an API to extract data from and write to RDF stores (known as graphs); and provides support to execute reasoners on OWL triples and generic rules. In the following sections the operation of the prototype is described, which is based on the scheme in Figure 1.

4.1. Loading reports in memory

As shown in Figure 1, the first step is to load in memory, in the Jena Model class object called `reports`, both reports to be combined, available in files with Turtle RDF triples (TTL) format. In Coding 5 the Java code used is shown.

Coding 5. Java code to load the two EARL reports in memory.

```
import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
...
String report1_file = "report1.ttl";
String report2_file = “report2.ttl";
Model reports = ModelFactory.createDefaultModel();
reports.read(report1_file,"TTL");
reports.read(report2_file,"TTL");
```

4.2. Loading knowledge/rules base in memory

The next step is to load in memory the knowledge base described in Coding 1, available as RDF triples in a “TTL” file. Coding 6 shows the Java code for this, creating the object `knowledgeBase`. It must be also loaded into memory the rules that complete the knowledge base, in this case available in a separate file, with extension txt. The content of this file is in Coding 2. Coding 7 shows the Java code to store the rules in a variable named rules, containing a list of Jena `Rule` class objects.

Coding 6. Java code to load the knowledge base in memory.

```
import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;

String knowledgeBase_file = "knowledgeBase.ttl";
Model knowledgeBase = ModelFactory.createDefaultModel();
knowledgeBase.read(knowledgeBase_file, "TTL");
```

Coding 7. Java code to load rules in memory.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.List;
import org.apache.jena.reasoner.rulesys.Rule;

String knowledgeBaseRules_file = "knowledgeBaseRules.txt";

BufferedReader knowledgeBase_rules = new BufferedReader(new
FileReader(knowledgeBaseRules_file));

List rules = Rule.parseRules (Rule.rulesParserFromReader
(knowledgeBase_rules) );
```

4.3. Using reasoners to infer new knowledge

Once all the knowledge is stored in different objects in memory, it's time to infer new knowledge from the restrictions included in the knowledge base, expressed with OWL and with the rules that complement the base.

First, the internal reasoner that Jena offers to work with OWL is executed. Coding 8 shows the Java code to do this. The reasoner starts from the knowledge base available in memory in the object `knowledgeBase`, and applies them to the reports stored in the object reports. The final result is the object `infModel`, which contains all the original RDF triples plus the new ones inferred. In this case, it is evident that new triples have been inferred because of the restriction `owl:sameAs`, which establishes the equivalence between two accessibility requirements: the success criteria 1.1.1 of WCAG 2.0 and the article 1194.22(a) from Law Section 508. For every RDF triple in which one of these requirements appears, a new triple is created with the other, as it has been expressed that they are equivalent.

Coding 8. Java code to use reasoner to infer new knowledge.

```
import org.apache.jena.reasoner.Reasoner;
import org.apache.jena.reasoner.ReasonerRegistry;
import org.apache.jena.rdf.model.InfModel;

...
Reasoner reasoner = ReasonerRegistry.getOWLReasoner();
reasoner = reasoner.bindSchema(knowledgeBase);
InfModel infModel = ModelFactory.createInfModel(reasoner, reports);
```

Then, on the knowledge available in the object `infModel` (original plus generated), a new reasoner is executed, able to interpret the rules expressed in the Jena rule language, which were loaded into memory in the list rules. Coding 9 shows the Java code to do it, obtaining a final knowledge model in the object `infModel2`.

Coding 9. Java code to run the rules reasoner.

```
Reasoner reasoner2 = new GenericRuleReasoner(rules);
InfModel infModel2 = ModelFactory.createInfModel (reasoner2,
infModel);
```

In this case, it is also clear that it should have generated new knowledge in the form of new RDF triples, since applying the rule of Coding 2 to the report of Coding 3, it must be inferred that if the website assessed did not meet the requirement on epilepsy according to WCAG 2.0, does not meet the American law on accessibility, which is more restrictive in this regard, as indicates the rule.

4.4. Generating the final evaluation results using SPARQL

After the steps above, we have in memory, in the object `infModel2`, a RDF model with all the knowledge associated with the accessibility evaluation of the website, both the included in the two original evaluation reports as new knowledge generated by the two reasoners. It is possible to make queries on the model, which is stored as RDF triples. To do this, we must use the SPARQL language, as stated in section 2. In this case, we will launch a query about the accessibility failures registered in the model, according to American Law Section 508. Coding 10 shows the way to express that query with SPARQL.

Coding 10. SPARQL sentence to obtain web accessibility evaluation results according US law section 508.

```
prefix earl: <http://www.w3.org/nss/earl#>
prefix ptr: <http://www.w3.org/2009/pointers#>
prefix doap: <http://usefulinc.com/ns/doap#>
prefix s508: <http://example.org/section508.owl#>
SELECT ?tool ?desc ?line
WHERE { ?a a earl:Assertion .
        ?a earl:assertedBy ?tool .
        ?a earl:test ?req .
        ?req a s508:Requirement .
        ?req s508:hasDescription ?desc .
        ?a earl:result ?res .
        ?res earl:outcome earl:failed .
        ?res earl:pointer ?pt .
        ?pt ptr:lineNumber ?line .
        ?pt ptr:charNumber ?char . }
```

The query with the source code of Coding 10 is stored in a file named query.txt, and it must be loaded into memory and then run it using resources from the Jena framework. In Coding 11 the corresponding Java code is shown.

Coding 11. Java code to load and run the SPARQL sentence.

```
import org.apache.jena.query.Query;
import org.apache.jena.query.QueryExecution;
import org.apache.jena.query.QueryExecutionFactory;
import org.apache.jena.query.QueryFactory;
import org.apache.jena.query.QuerySolution;
import org.apache.jena.query.ResultSet;

String sparql_file = "query.txt";
BufferedReader bf = new BufferedReader(new
FileReader(sparql_file));
String line;
String sparql_sentence = "";
while((line = bf.readLine())!=null) {
  sparql_sentence += line;
  }
  bf.close();

Query sparqlQuery = QueryFactory.create(sparql_sentence);
QueryExecution qe =

QueryExecutionFactory.create(sparqlQuery,infModel2);
ResultSet rs = qe.execSelect();
while (rs.hasNext()) {
  QuerySolution sol = rs.next();
  System.out.println(sol.toString());
  }
```

The query results can be seen in Table 1. The query has been designed to obtain the accessibility requirements unfulfilled by the website evaluated, according to the American legislation. Three columns appear: the first one contains the assessment tool that, according the processed reports, detected the fail. The second column includes the textual description of the unfulfilled requirement. The third column shows the line in the HTML page code where the failure has been found. The importance of the example shown is that it has combined results of both reports, and that some of the results did not explicitly appear in any of the reports, and have appeared thanks to the information inferred by reasoners. Thus, the failure in line 37 appears because of the restriction `owl:sameAs` on the requirements about alternative text. And the failure in line 56 has appeared thanks to the inference rule about the limits of screen flickering.

Table 1. Results of the SPARQL query.

Tool	Description	Line
AChecker	"1194.22(a) A text equivalent for every non-text . ."	25
OAW	"1194.22(j) Pages shall be designed to avoid . ."	56
OAW	"1194.22(a) A text equivalent for every non-text . ."	37

5. Conclusions

The relationship between Semantic Web technologies and Web accessibility can be treated from different points of view. This relationship is evident in the case of the W3C, because this organization is involved in the development of both fields. Such is the case of EARL language for expressing accessibility test results in the form of semantic RDF triples. It is beginning to appear on the market accessibility evaluation tools that obtain their reports in this format, as is the case of the known AChecker, one of the most used by the evaluators [3]. The advantage of using EARL in reports is that we can apply technologies of the Semantic Web to process these reports, and take advantage of the great possibilities offered, such as those have been shown in this work. EARL 1.0 is a Working Draft since 2011, but it has been reactivated in 2017 mainly due to the creation of the Accessibility Conformance Testing (ACT) Task Force in the W3C, to develop a framework and repository of test rules, to promote a unified interpretation of WCAG 2.0 among different web accessibility test tools, including examples using JSON-LD and EARL []. Taking into account that the main developers of evaluation tools are part of this working group, the potential of EARL as a language that they use in the future for evaluation reports is clear.

Semantic Web technologies will facilitate the interoperability between accessibility evaluation tools and will allow the creation of federated evaluation systems to ensure obtaining the best results, as the authors of this paper have proposed in [7].

Other studies have also combined semantic technologies in the context of the evaluation of Web accessibility. In [6] and [18] a semantic assessment environment called WaaT (Web Accessibility Assessment Tool) is presented. It uses semantic models, representing the most of main accessibility constrains and terms which are required for the design and development of Web applications, through the use of generic and domain ontologies. This tool evaluates the accessibility status of a Web page according to the WCAG 2.0 and WAI-ARIA [2] guidelines, and generate EARL reports. To do this, it accesses different ontologies to complete the information in reports. One of the ontologies created by the authors of this tool is about the WCAG standard, and has been used in our work as reference vocabulary in the evaluation reports based on this standard (Coding 3). The advantage of ontologies is that it is easy to reuse knowledge, as in this case. Unlike our work, the authors based their work on a single standard as WCAG and a single assessment tool. In our case, we combine several reports obtained from different tools and applying different accessibility standards.

Another relevant work is [4], where a conceptual framework for automatic accessibility evaluation of Rich Internet Applications (RIA) is presented. RIA applications included widgets. In this work, the core semantics of the widgets are represented as OWL ontology. This ontology is used to test the conformance of the application with the WAI-ARIA specification, making use of the taxonomy about WAI-ARIA that has been published by the W3C in RDF format, as an appendix included in the specification itself [5]. Like in the first work, this one also used Semantic Web technologies for the evaluation of accessibility according to a single standard and using a single tool.

In conclusion, we can say that there are no published studies using Semantic Web techniques to combine accessibility reports of different tools and applying different accessibility standards. In general, the published works are aimed at creating assessment tools that use these technologies to generate EARL reports semantically enriched with terms obtained from predefined vocabularies or ontologies.

As future work, we intend to integrate the prototype in a complete service-based system for Web Accessibility Federated Evaluation, whose architecture was presented in [7]. The system will use the federation of RESTful services exposed by different accessibility evaluation tools returning EARL results. And will combining the results of evaluations of the same Website by different tools using Semantic Web technologies, applying different criteria or preferences established by the evaluator.

Acknowledgements

Thanks to the ESVI-AL Cooperation Network, and to the “Master on Software Engineering for the Web” of the University of Alcalá.

References

1. Accessibility Conformance Testing (ACT) Rules Format 1.0. World Wide Web Consortium, 2017. https://www.w3.org/TR/act-rules-format/. [retrieved: 06, 2017]
2. Accessible Rich Internet Applications (WAI-ARIA) 1.0. World Wide Web Consortium, 2014.
3. AChecker. Inclusive Design Research Centre, 2011. http://achecker.ca. [retrieved: 06, 2017]
4. Doush, I. A., Alkhateeb, F., Al Maghayreh, E., and Al-Betar, M. A. The design of RIA accessibility evaluation tool. Advances in Engineering Software, 57, 2013, 1-7.
5. Evaluation and Report Language (EARL) 1.0 Schema. World Wide Web Consortium, 2017. https://www.w3.org/TR/EARL10-Schema/. [retrieved: 06, 2017]
6. Fernandes, N., Kaklanis, N., Votis, K., Tzovaras, D., and Carriço, L. An analysis of personalized web accessibility. In Proceedings of the 11th Web for All Conference (W4A'14) (Seoul, Republic of Korea, April 07-09, 2014). ACM Press, New York, NY, 2014, Article 19, 10 pages.
7. Hilera, J.R., Otón, S., Martin-Amor, C., and Timbi-Sisalima, C. Towards a Service-based Architecture for Web Accessibility Federated Evaluation. In Proceedings of the 9th International Conference on Advances in Computer-Human Interactions (ACHI'16) (Venice, Italy, April 24-28, 2016). IARIA, 2016, 6-10.
8. ISO/IEC 40500:2012, Information technology -- W3C Web content accessibility guidelines (WCAG) 2.0. International Organization for Standardization, 2012.
9. OWL 2 Web Ontology Language Primer (Second Edition). World Wide Web Consortium, 2012.
10. RDF 1.1 Turtle. World Wide Web Consortium, 2014.
11. RDF Schema 1.1. World Wide Web Consortium, 2014.
12. Reasoners and rule engines: Jena inference support. Apache Software Foundation, 2017. https://jena.apache.org/documentation/inference/. [retrieved: 06, 2017]
13. Semantic Web. World Wide Web Consortium, 2016. https://www.w3.org/standards/semanticweb/. [retrieved: 06, 2017]
14. SPARQL 1.1 Query Language. World Wide Web Consortium, 2013.
15. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. World Wide Web Consortium, 2004.
16. Timbi-Sisalima C., Hilera J., Otón S. and Ingavelez P. Developing a RESTful API for a Web Accessibility Evaluation Tool. In Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS'16) (Rome, Italy, April 25-28, 2016). SCITEPRESS, 2016, 443-450.
17. Timbi-Sisalima C., Martín-Amor, C., Otón, S., Hilera, J.R., and Aguado-Delgado, J. Comparative Analysis of Online Web Accessibility Evaluation Tools. In Proceedings of the 25th International Conference on Information Systems Development (ISD'16) (Katowice, Poland, August 24-26, 2016). University of Economics in Katowice, 2016, 562-573.
18. Votis, K., Lopes, R., Tzovaras, D., Carrico, L., and Likothanassis, S. A Semantic Accessibility Assessment Environment for Design and Development for the Web. In Proceedings of Int. Conf. on Universal Access in Human-Computer Interaction (UAHCI'09) (San Diego, CA, USA, July 19-24, 2009). Springer, Berlin, 803-813.
19. Web Accessibility Evaluation Tools List. Wide Web Consortium, 2016. http://www.w3.org/WAI/ER/tools/. [retrieved: 06, 2017]
20. Web content accessibility guidelines (WCAG) 2.0. World Wide Web Consortium, 2008.
21. Web-based Intranet and Internet Information and Applications (1194.22). In Guide to the Section 508 Standards. United States Access Board, 2001. https://www.access-board.gov/guidelines-and-standards. [retrieved: 06, 2017]