[contents]

# Developer Guide for Evaluation and Report Language (EARL) 1.0

## W3C Working Draft 10 May 2011

This version:
http://www.w3.org/TR/2011/WD-EARL10-Guide-20110510/
Latest version:
http://www.w3.org/TR/EARL10-Guide/
Previous version:
http://www.w3.org/TR/2009/WD-EARL10-Guide-20091029/

Editors:
Carlos A Velasco, Fraunhofer Institute for Applied Information Technology FIT
Shadi Abou-Zahra, W3C/WAI
Previous Editors:
Johannes Koch (until November 2010 while at Fraunhofer Institute for Applied Information Technology FIT)

## Abstract

This document provides guidance for developers on implementing Evaluation and Report Language (EARL) 1.0 in software tools and process. EARL is a vocabulary, the terms of which are defined across a set of specifications and technical notes, that is used to describe test results. The primary motivation for developing this vocabulary is to facilitate the exchange of test results between web accessibility evaluation tools in a vendor-neutral and platform-independent format. It also provides reusable terms for generic quality assurance and validation purposes.

While this document provides developer guidance for using and implmenting EARL, Evaluation and Report Language (EARL) 1.0 Schema defines the core terms of the vocabulary, and other specifications provide additional terms for representing HTTP exchanges between clients and servers, HTTP Vocabulary in RDF 1.0, for representing web content itself, Representing Content in RDF 1.0, or for specifying particular locations within or sections of content, Pointer methods in RDF 1.0. An Evaluation and Report Language (EARL) Overview is also available.

## Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This 10 May 2011 Working Draft of the Developer Guide for Evaluation and Report Language (EARL) 1.0 provides an introduction to EARL and defines conformance requirements for software tools supporting EARL. It is a complete resource with different working examples, and it implements the decisions of the Evaluation and Repair Tools Working Group (ERT WG) to date. This document is intended to be published and maintained as a W3C Recommendation after review and refinement.

The Evaluation and Repair Tools Working Group (ERT WG) encourages feedback about this document, Developer Guide for Evaluation and Report Language (EARL) 1.0, by developers and researchers who have interest in software-supported evaluation and validation of websites, and by developers and researchers who have interest in Semantic Web technologies for content description, annotation, and adaptation. In particular, the Working Group is looking for feedback on the section on conformance and suggestions for the section on serialization that is currently under consideration.

Please send comments on this Developer Guide for Evaluation and Report Language (EARL) 1.0 document by 10 June 2011 to public-earl10-comments@w3.org (publicly visible mailing list archive).

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document has been produced by the Evaluation and Repair Tools Working Group (ERT WG) as part of the Web Accessibility Initiative (WAI) Technical Activity.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. The group does not expect this document to become a W3C Recommendation. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

## Table of Contents

## 1. Introduction

This document is a guide to the Evaluation and Report Language (EARL) 1.0 for developers of software tools and proccesses. It provides an introduction to EARL and its uses, defines conformance requirements for tools supporting EARL, and describes approaches for serializing EARL data in different formats.

EARL is a vocabulary, the terms of which are defined across a set of specifications and technical notes, that is used to describe test results in a machine-readable format. This set of specifications includes:

- Evaluation and Report Language (EARL) 1.0 Schema
- HTTP Vocabulary in RDF 1.0
- Representing Content in RDF 1.0
- Pointer Methods in RDF 1.0

An Evaluation and Report Language (EARL) Overview is also available.

### 1.1 Audience of this Document

The assumed audience of this document is developers of software tools and processes, who want to implement EARL. This includes developers of quality assurance tools, in particular developers of web accessibility evaluation tools, web authoring tools, and web quality assurance tools.

This document assumes that the reader is familiar with the Resource Description Framework (RDF) and can read its XML serialization. Readers who wish to understand more about RDF should read a general introduction or the RDF Primer [RDF-PRIMER]. This document is also written with consideration for developers who are more accustomed to XML than RDF, but reader is cautioned about notable differences between the syntax-based nature of XML and the semantic-based nature of RDF.

### 1.2. Document conventions

Keywords

The keywords must, required, recommended, should, may, and optional in this document are used in accordance with RFC 2119 [RFC 2119].

Namespaces

The RDF representation of the vocabulary defined by this document uses the namespace `http://www.w3.org/ns/earl#`. The prefix `earl` is used throughout this document to denote this namespace. Other prefixes used throughout this document include:

- `cnt` - Representing Content in RDF namespace http://www.w3.org/2011/content# (defined by [Content])
- `dct` - Dublin Core (DC) namespace http://purl.org/dc/terms/ (defined by [DC])
- `doap` - Description of a Project (DOAP) namespace http://usefulinc.com/ns/doap# (defined by [DOAP])
- `foaf` - Friend of a Friend (FOAF) namespace http://xmlns.com/foaf/0.1/# (defined by [FOAF])
- `http` - HTTP Vocabulary in RDF namespace http://www.w3.org/2011/http# (defined by [HTTP])
- `ptr` - Pointer Methods in RDF namespace http://www.w3.org/2009/pointers# (defined by [Pointers])
- `rdf` - RDF namespace http://www.w3.org/1999/02/22-rdf-syntax-ns# (defined by [RDF])
- `rdfs` - RDF Schema namespace http://www.w3.org/2000/01/rdf-schema# (defined by [RDFS])
- `xsd` - XMLS namespace http://www.w3.org/2001/XMLSchema# (defined by [XMLS])

## 2. About Evaluation and Report Language (EARL)

The Evaluation and Report Language (EARL) is a vocabulary to describe test results in a machine-readable format. EARL supports the test reporting stage in the testing process, as described by different standards on testing, such as IEEE 829 [IEEE-829]. Stages in the testing process include:

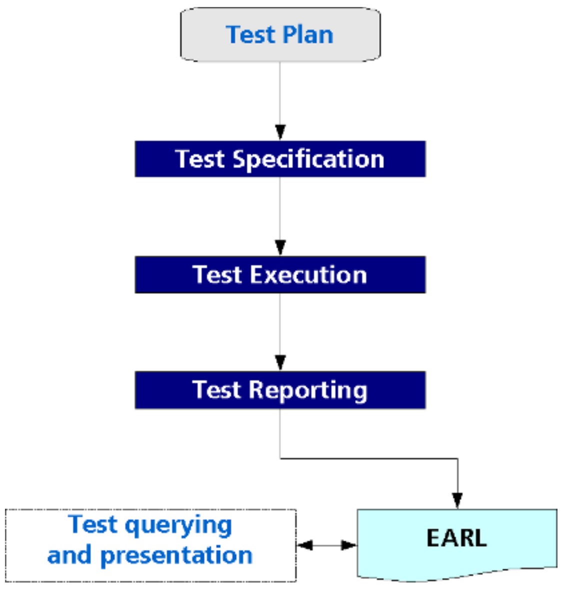

Figure 1. Stages in the testing processes.

1. Test Plan, which prescribes the scope, approach, resources, and schedule of the testing activities.
2. Test Specification, which describes the test cases and the test procedures for carrying out individual tests.
3. Test Execution, which covers the actual execution of the tests according to the test plan and specification.
4. Test Reporting, which deals not only with the creation of test results (i.e. reports), but may include their post-processing (e.g. filtering, aggregation, summarization, etc.).

EARL vocabulary can be used to describe resources to be tested as defined by the test plan, test cases and criteria as defined by the test specification, and outcomes of the test execution stages. More importantly, EARL provides a format to uniformly record these and other elements in semantically rich testing reports.

## 2.1 Structure of EARL

[Editor note: this section overlaps with section "2. Classes" of EARL 1.0 Schema and will be reviewed after further review and refinement.]

The terms of EARL are defined using the Resource Description Framework [RDF], which is technology to express semantic data in a machine-readable format. Like any RDF vocabulary, EARL is a collection of statements about resources, each with a subject, a predicate (or verb), and an object. RDF statements describe resources and relationships, such as in the following example:

```
<#someone> <#checks> <#resource> .
<#resource> <#fails> <#test> .
```

EARL provides a standardized vocabulary to describe specific resources and relationships that are relevant to test reporting. The core construct of EARL is an Assertion, which describes the context and outcome of an individual test execution. It contains the following information:

Assertor
: This can include information about who or what ran the test. For example human evaluators, automated accessibility checkers, or combinations of these.

Test Subject
: This can include web content (such as web pages, videos, applets, etc.), software (such as authoring tools, user agents, etc.), or other things being tested.

Test Criterion
: What are we evaluating the test subject against? This could be a specification, a set of guidelines, a test from a test suite, or some other testable statement.

Test Result
: What was the outcome of the test? A test result could also include contextual information such as error messages or relevant locations within the test subject.

Examples

Example 1: A person carries out a manual evaluation of a web page to an accessibility requirement.

Assertor
: Bob B. Bobbington

Test Subject
: A web page located at `http://www.example.org/page.html`

Test Criterion
: Success Criterion 1.1.1 of the Web Content Accessibility Guidelines (WCAG) 2.0

Test Result
: Passed

Example 2: A software application carries out automated validation of a web page to a technical specification.

Assertor

The W3C Markup Validator located at http://validator.w3.org/

Test Subject

The XHTML returned from a GET request to the URI http://www.example.org/page.html at 2004-04-14T14:00:04+1000

Test Criterion

The validity of the XHTML code

Test Result

Failed, the <li> element on line 53, char 7 was not closed.

## 2.2 Uses of EARL

As a standardized machine-readable format, EARL facilitates processing of test results, such as those generated by automated or semi-automated web accessibility evaluation tools. Web authoring tools and quality assurance software can aggregate such test results to support website developers in developing high quality web content. EARL has been specifically designed to support a broad variety of uses cases, including the following:

Combining results from software tools
: Quality assurance testing, such as web accessibility evaluation, is often carried out by combinations of software tools and human evaluators. For instance, different evaluators may be testing different parts of a website, and single a evaluator may be using one or more software tools for testing or recording test results. Some tests might be fully automatable, and may be executed without any human intervention. Partial reports from different software tools can be combined, by using EARL as the standardized format for expressing test results.

Exchanging data between software tools
: A standardized format for expressing test results also allows software tools to exchange data. In particular, testing tools such as checkers and validators can be more easily integrated into authoring tools, such as content management systems. Testing tools can also be integrated into quality assurance tools that help process and analyze the test results, or that provide customized reports of the test results for different audiences.

Querying and analyzing test reports
: EARL provides fine-grained data about the context and outcome of test results, including information about the tested resources and the testing modalities, to allow many different types of queries and analysis. For instance, queries can be used to generate customized reports for managers who want a higher-level view, project managers who want information specific to the resources they are managing, and developers who want detailed bug reports about errors they need to fix. The nature of RDF also allows semantic inference and other approaches for advanced data mining.

Benchmarking software testing tools
: EARL can also be used to compare the results provided by different testing tools, such as web accessibility evaluation tools. In particular, it can be used to compare the results provided from executing test suites that have normalized outcome, and so benchmark deviations such as false positives and false negative generated by different testing tools.

Evaluating dynamic and multilingual websites
: EARL includes vocabulary to describe comprehensively web resources, including any parts of the entire HTTP exchange between a client and a server. This is particularly useful to record HTTP headers relevant for language and content negotiation, as well as the actual content received from the server and that has been tested. Moreover, user interaction with a website can be recorded, to help describe the particular context of the test execution.

Annotating web resources with metadata
: Test results can also be used to describe the availability or lack of particular features of the resources tested. As RDF data, EARL test results are particularly useful as metadata for describing features of web content in a machine-readable format. For instance, EARL reports could be associated with web resources using RDFa [provide reference], and can be processed by RDF-aware browsers and search-engines to serve particular user preferences.

## 2.3 Limitations of EARL

[Editor note: this section will be extended and refined in later iterations

It is important to consider potential security and privacy issues when using EARL. For instance, test results expressed in EARL could contain sensitive information such as the internal directory structure of a web server, username and password information, parts of restricted Web pages, or testing modalities. The scope of this document is limited to the use of the EARL vocabulary: security and privacy considerations need to be made at the application level. For example, certain parts of the data may be restricted to appropriate user permissions, encrypted or obfuscated.

# 3. Using the Evaluation and Report Language (EARL)

[Editor note: this entire section will be revised and refined in later iterations.

EARL is not a standalone vocabulary and builds on top of many existing vocabularies that cover some of its needs for metadata definition. This approach avoids the re-creation of applications already established and tested like the Dublin Core elements. The referenced specifications are:

- Dublin Core Metadata Initiative (DCMI) Terms. Dublin Core is a metadata standard for describing digital resources, often expressed in XML. The aforementioned document is an up-to-date specification of all metadata terms maintained by the Dublin Core Metadata Initiative. Included are the fifteen terms of the Dublin Core Metadata Element Set, which have also been published as IETF RFC 5013 [RFC5013], ANSI/NISO Standard Z39.85-2007 [NISOZ3985] and ISO Standard 15836 [ISO15836]. RDF Schema versions of the DCMI term declarations are available at [DCMISCHEMAS].
- Friend of a Friend (FOAF) project. The FOAF project is about creating a Web of machine-readable resources describing people, the links between them, and the things they create and do [FOAF].
- Representing Content in RDF [Content-RDF]. This is an RDF vocabulary to semantically represent any type of content, either on the Web or in any storage media.
- HTTP vocabulary in RDF [HTTP-RDF]. This is an RDF vocabulary used to represent HTTP requests and responses. It is useful to identify online resources accessed via HTTP(S), which cannot be uniquely resolved via a URI [URI]. Typical examples are Web servers accessed via content negotiation, Web applications using the POST method, etc.
- Pointer Methods in RDF [Pointers-RDF]. This is an RDF vocabulary to enable pointing in an accurate way to certain parts within a document, particularly HTML and XML documents.

These vocabularies are referenced via namespaces in the corresponding RDF serialization. The list of the normative namespaces can be found in the EARL 1.0 Schema. RDF can be serialized in many equivalent ways, but its XML presentation [RDF/XML] is the preferred method and will be used throughout this document.

### 3.1 Basic report components

In the following sections, we will construct an EARL report with several examples of each component of the report. The root element of any EARL report is an RDF node, in which we declare the namespaces used to define additional classes and/or properties.

Example 3.1. The root element of an EARL report [download file for example 3.1].

```
  <rdf:RDF
         xmlns:earl="http://www.w3.org/ns/earl#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

    <!-- ... -->

</rdf:RDF>
```

Next, let us assume that we want to express the results of an XHTML validation in a given document with the W3C HTML Validator. The tested document can be found in the fictitious URL `http://example.org/resource/index.html` and has the following HTML code:

Example 3.2. An XHTML document to be validated [download file for example 3.2].

```
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html lang="en" xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
  <title>Example of project pages</title>
  </head>
  <body>
  <h1>Project description</h1>
  <h2>My project name</h2>

    <!-- ... -->
  </body>
</html>
```

This document has three errors that will constitute the basis of our EARL report:

1. Error: Line 14, column 7: document type does not allow element "`li`" here; missing one of "`ul`", "`ol`" start-tag.
2. Error: Line 15, column 6: end tag for "`li`" omitted, but OMITTAG NO was specified.
3. Error: Line 16, column 9: there is no attribute "`alt`".

The first step is to define who performed the test, either a human being or a software tool. This is noted in the EARL framework as an `Assertor`. Let us consider different use cases. First, let us assume that only the W3C HTML Validator performed the test. This could be expressed as an `Assertor`:

Example 3.3. A generic tool as an `Assertor` [download file for example 3.3].

```
    <earl:Assertor rdf:about="http://validator.w3.org/about.html#">
    <dct:title xml:lang="en">W3C HTML Validator</dct:title>
    <dct:description xml:lang="en">
    W3C Markup Validation Service, a free service that checks Web documents in formats like HTML and XHTML for conformance to W3C Reco
    </dct:description>
  </earl:Assertor>
```

Notice that the `Assertor` class provides a mechanism by which to specify more information by leveraging standard Dublin Core properties like `dct:title` and `dct:description`. This is not the only possible serialization of this report. An alternative, expressed in N3, could be:

Example 3.4. An `Assertor` expressed in N3 notation [download file for example 3.4].

```
  @prefix earl:    <http://www.w3.org/ns/earl#> .
@prefix dct:     <http://purl.org/dc/terms/> .

<http://validator.w3.org/about.html#>
      a       earl:Assertor ;
      dct:description """W3C Markup Validation Service, a free service that checks Web documents in formats like HTML and XHTML for co
      dct:title "W3C HTML Validator"@en .
```

An `Assertor` is a generic type. EARL allows the use of certain FOAF classes like `Agent`, `Organisation`, or `Person` to provide more semantic information on the type of assertor. Additionally, EARL defines the `Software` class to declare tool assertors. Thus, our W3C Validator could be described more adequately in the following way:

Example 3.5. A `Software` assertor [download file for example 3.5].

```
      <earl:Software rdf:about="http://validator.w3.org/about.html#">
    <dct:title xml:lang="en">W3C HTML Validator</dct:title>
    <dct:hasVersion>0.7.1</dct:hasVersion>
    <dct:description xml:lang="en">
    W3C Markup Validation Service, a free service that checks web documents in formats like HTML and XHTML for conformance to W3C Reco
    </dct:description>
  </earl:Software>
```

Notice the aditional property, `dct:hasVersion`, indicating the version of the software. Let us consider now the case where the assertor is a person. This can be expressed as in the following example:

Example 3.6. A `Person` as an EARL assertor [download file for example 3.6].

```
<foaf:Person rdf:ID="john">
  <foaf:mbox rdf:resource="mailto:john@example.org"/>
  <foaf:name>John Doe</foaf:name>
</foaf:Person>
```

We could combine assertors as well. The typical example could be an expert evaluator and a software tool, which perform the analysis. This set of assertors can be expressed under the umbrella of a `foaf:Group`. We should define who is the main assertor within a `foaf:Group` through the `mainAssertor` property (notice in the example how the person is defined as a blank node):

Example 3.7. A `foaf:Group` (software tool and person) as an assertor [download file for example 3.7].

```
<foaf:Group rdf:ID="assertor01">
  <dct:title>John Doe and the W3C HTML Validator</dct:title>
  <earl:mainAssertor rdf:resource="http://validator.w3.org/about.html#"/>
  <foaf:member>
    <foaf:Person>
      <foaf:mbox rdf:resource="mailto:john@example.org"/>
      <foaf:name>John Doe</foaf:name>
    </foaf:Person>
  </foaf:member>
</foaf:Group>
```

The second step is to define what was analyzed, the tested resource. For that, EARL defines the `TestSubject` class. This class is a generic wrapper for things to be tested like Web resources (`cnt:Content`) or software (`earl:Software`). In this case, the Example 3.2 could be represented as:

Example 3.8. A `TestSubject` with some Dublin Core properties (non-abbreviated RDF/XML serialization) [download file for example 3.8].

```
<rdf:Description rdf:about="http://example.org/resource/index.html">
  <dct:title xml:lang="en">Project Description</dct:title>
  <dct:date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2006-02-14</dct:date>
  <rdf:type rdf:resource="http://www.w3.org/ns/earl#TestSubject"/>
</rdf:Description>
```

Using the Representing Content in RDF vocabulary (via the `cnt:ContentAsText` class), we could insert the content of the test XHTML file into the report:

Example 3.9. A test subject expressed as `cnt:ContentAsText` (notice that the special XML characters have been escaped because the document is not well-formed to be expressed as an XML Literal) [download file for example 3.9].

```
  <cnt:ContentAsText rdf:about="http://example.org/resource/index.html">
    <dct:title xml:lang="en">Project Description</dct:title>
    <dct:date rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2006-02-14</dct:date>
    <cnt:characterEncoding>UTF-8</cnt:characterEncoding>
    <cnt:chars>&lt;?xml version="1.0" encoding="UTF-8"?&gt;

&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"&gt;

&lt;html lang="en" xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"&gt;
  &lt;head&gt;
  &lt;title&gt;Example of project pages&lt;/title&gt;
  &lt;/head&gt;
  &lt;body&gt;
  &lt;h1&gt;Project description&lt;/h1&gt;
  &lt;h2&gt;My project name&lt;/h2&gt;
  &lt;p&gt;The strategic goal of this project is to make you understand EARL.&lt;/p&gt;

  &lt;ul&gt;
    &lt;li&gt;Here comes objective 1.
    &lt;li&gt;Here comes objective 2.&lt;/li&gt;
  &lt;/ul&gt;
  &lt;p alt="what?"&gt;And goodbye ...&lt;/p&gt;
  &lt;/body&gt;
&lt;/html&gt;
</cnt:chars>
  </cnt:ContentAsText>
```

The third step is to define the criterion used for testing the resource. EARL defines test criteria under the umbrella of the `TestCriterion` class. This class has two subclasses, `TestRequirement` and `TestCase`, depending on whether the criterion is a high level requirement, composed of many tests, or an atomic test case. In our example, we are testing validity against the XHTML 1.0 Strict specification, which could be expressed in the following way via the `TestRequirement` class:

Example 3.10. A `TestRequirement` with some Dublin Core properties [download file for example 3.10].

```
<earl:TestRequirement rdf:about="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  <dct:title xml:lang="en">XHTML 1.0 Strict Document Type Definition</dct:title>
  <dct:description xml:lang="en">DTD for XHTML 1.0 Strict.</dct:description>
</earl:TestRequirement>
```

The fourth step is to specify the results of the test. There were three errors discovered by the W3C Validator that need to be presented as `TestResult`s. In this case, we present only the errors, but it is also possible to present positive results. In the example below, we present the message errors as text messages within XHTML snippets. We will see later how to improve the machine-readability of such results.

Example 3.11. Results of the tests with the validator [download file for example 3.11].

```
<earl:TestResult rdf:ID="error1">
  <dct:description rdf:parseType="Literal">
    <div xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">
      <p>Error - Line 14 column 7: document type does not allow element
         <code>li</code>here; missing one of
         <code>ul</code>, <code>ol</code> start-tag.</p>
    </div>
  </dct:description>
  <earl:outcome rdf:resource="http://www.w3.org/ns/earl#failed" />
</earl:TestResult>

<earl:TestResult rdf:ID="error2">
  <dct:description rdf:parseType="Literal">
    <div xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">
      <p>Error - Line 15 column 6: end tag for
         <code>li</code> omitted, but OMITTAG NO was specified.</p>
    </div>
  </dct:description>
  <earl:outcome rdf:resource="http://www.w3.org/ns/earl#failed" />
</earl:TestResult>

<earl:TestResult rdf:ID="error3">
  <dct:description rdf:parseType="Literal">
    <div xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">
      <p>Error - Line 16 column 9: there is no attribute
         <code>alt</code>.</p>
    </div>
  </dct:description>
  <earl:outcome rdf:resource="http://www.w3.org/ns/earl#failed" />
</earl:TestResult>
```

## 3.2 Putting the pieces together

The final step is to merge together the created components. The EARL statements for this purpose are called `Assertion`s, and have four key properties: `earl:assertedBy`, `earl:subject`, `earl:test` and `earl:result`. Each of them serves to point to the corresponding assertors, test subjects, test requirements, and results, respectively. From our previous examples, we could build our first complete report with our three assertions:

Example 3.12. Results of the tests with the W3C Validator [download file for example 3.12].

```
<earl:Assertion rdf:ID="ass1">
      <earl:result rdf:resource="#error1" />
      <earl:test rdf:resource="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" />
      <earl:subject rdf:resource="http://example.org/resource/index.html" />
      <earl:assertedBy rdf:resource="#assertor01" />
  </earl:Assertion>
  <earl:Assertion rdf:ID="ass2">
      <earl:result rdf:resource="#error2" />
      <earl:test rdf:resource="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" />
      <earl:subject rdf:resource="http://example.org/resource/index.html" />
      <earl:assertedBy rdf:resource="#assertor01" />
  </earl:Assertion>
  <earl:Assertion rdf:ID="ass3">
      <earl:result rdf:resource="#error3" />
      <earl:test rdf:resource="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" />
      <earl:subject rdf:resource="http://example.org/resource/index.html" />
      <earl:assertedBy rdf:resource="#assertor01" />
  </earl:Assertion>
```

## 3.3 An accessibility example

Our next example presents the results of an accessibility test in a given Web resource. Let us consider a simple XHTML page, which presents the image of a cat:

Example 3.13. An XHTML document to be verified [download file for example 3.13].

```
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html lang="en" xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
 <head>
 <title>A cat's photography</title>
 </head>
 <body>
 <h1>A cat's photography</h1>
 <p>Image of a cat who likes
   <acronym title="Evaluation and Report Language">EARL</acronym>, although it
   seems quite tired.
   <img src="../images/cat.jpg" alt="Image of a white cat with black spots."/>
 </p>
 </body>
</html>
```

We have in this case a software tool called "Cool Tool" that performs a test against the Common Failure F65 from the (X)HTML techniques for WCAG 2.0 [WCAG20]. This technique proofs the existence of the `alt` attribute for given (X)HTML elements like `img`. The software can be represented as:

Example 3.14. A `Software` assertor [download file for example 3.14].

```
<earl:Software rdf:about="http://example.org/cooltool/">
   <dct:title xml:lang="en">Cool Tool accessibility checker</dct:title>
   <dct:hasVersion>1.0.c</dct:hasVersion>
   <dct:description xml:lang="en">A reliable compliance checker for Web Accessibility</dct:description>
 </earl:Software>
```

The test requirement can be represented as:

Example 3.15. A `TestCase` for a WCAG 2.0 technique [download file for example 3.15].

```
 <earl:TestCase rdf:about="http://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/F65">
   <dct:title xml:lang="en">Failure of Success Criterion 1.1.1 from WCAG 2.0</dct:title>
   <dct:description xml:lang="en">Failure due to omitting the alt attribute on img elements, area elements, and input elements of typ
 </earl:TestCase>
```

We can make the test result more amenable to machine processing by making use of the Pointers [Pointers-RDF] vocabulary. In this case, we identify the line number where the test was compliant:

Example 3.16. A `TestResult` with a pointer [download file for example 3.16].

```
 <ptr:LineCharPointer rdf:ID="pointer">
   <ptr:lineNumber>15</ptr:lineNumber>
   <ptr:reference rdf:resource="http://example.org/resource/index.html" />
 </ptr:LineCharPointer>

 <earl:TestResult rdf:ID="result">
   <earl:pointer rdf:resource="#pointer" />
   <earl:outcome rdf:resource="http://www.w3.org/ns/earl#passed" />
 </earl:TestResult>
```

Which leads to the following assertion:

Example 3.17. Accessibility `Assertion` [download file for example 3.17].

```
 <earl:Assertion rdf:ID="assert">
   <earl:result rdf:resource="result" />
   <earl:test rdf:resource="http://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/F65" />
   <earl:subject rdf:resource="http://example.org/resource/index.html" />
   <earl:assertedBy rdf:resource="http://example.org/cooltool/" />
 </earl:Assertion>
```

## 3.4 Identifying unambiguously the resource

There are cases where the identification of a resource on the Web requires more than a URL. This occurs typically when the user agent and the server exchange HTTP messages via Content Negotiation to deliver the best possible alternative to the client. A common scenario appears when the user expresses a preference for given languages with a ranking via the Accept-Language header. Under those circumstances, it is necessary to use the HTTP vocabulary in RDF [HTTP-RDF] to identify correctly the `TestSubject`.

Let us assume that our exemplary Web server can deliver under the URL `http://example.org/resource/index.html` two versions (English and Spanish) of a given XHTML page. The English version can be seen in Example 3.13. The Spanish version can be seen in the listing below:

Example 3.18. An XHTML file resource in Spanish [download file for example 3.18].

```
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html lang="es" xmlns="http://www.w3.org/1999/xhtml" xml:lang="es">
    <head>
        <title>Fotografia de un gato</title>
    </head>

    <body>
        <h1>Fotografia de un gato</h1>
        <p>Imagen de un gato al que le gusta
        <acronym title="Evaluation and Report Language" xml:lang="en" lang="en">EARL</acronym>, aunque aparenta estar muy cansado.
        <img src="../images/cat.jpg" />
        </p>
    </body>

</html>
```

The English resource can be represented as:

Example 3.19. RDF representation of Example 3.13 [download file for example 3.19].

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:earl="http://www.w3.org/ns/earl#"
    xmlns:dct="http://purl.org/dc/terms/"
    xmlns:cnt="http://www.w3.org/2008/content#"
    xmlns:http="http://www.w3.org/2006/http#"
    xml:base="http://www.example.org/resource/content_001#">

    <cnt:ContentAsBase64 rdf:ID="content1">
        <cnt:bytes rdf:datatype="http://www.w3.org/2001/XMLSchema#base64Binary">PD94bWwgdmVyc2lv...</cnt:bytes>
    </cnt:ContentAsBase64>

    <http:Response rdf:ID="response1">
        <http:httpVersion>1.1</http:httpVersion>
        <http:statusCodeNumber>200</http:statusCodeNumber>
        <http:sc rdf:resource="http://www.w3.org/2008/http-statusCodes#200" />
        <http:reasonPhrase>OK</http:reasonPhrase>
        <http:headers rdf:parseType="Collection">
            <http:MessageHeader>
                <http:fieldName>Vary</http:fieldName>
                <http:hdrName rdf:resource="http://www.w3.org/2008/http-headers#vary" />
                <http:fieldValue>Accept-Language</http:fieldValue>
            </http:MessageHeader>
            <!-- ... -->
        </http:headers>
        <http:body rdf:resource="#content1" />
    </http:Response>

    <http:Connection rdf:ID="connection1">
        <http:connectionAuthority>www.example.org:80
        </http:connectionAuthority>
        <http:requests rdf:parseType="Collection">
            <http:Request rdf:resource="#request1" />
        </http:requests>
    </http:Connection>

    <http:Request rdf:ID="request1">
        <http:httpVersion>1.1</http:httpVersion>
        <http:methodName>GET</http:methodName>
        <http:mthd rdf:resource="http://www.w3.org/2008/http-methods#GET" />
        <http:abs_path>/resource/index.html</http:abs_path>
        <http:headers rdf:parseType="Collection">
            <http:MessageHeader>
                <http:fieldName>Accept-Language</http:fieldName>
                <http:hdrName rdf:resource="http://www.w3.org/2008/http-headers#accept-language" />
                <http:fieldValue>en</http:fieldValue>
            </http:MessageHeader>
            <!-- ... -->
        </http:headers>
        <http:resp rdf:resource="#response1" />
    </http:Request>

</rdf:RDF>
```

The Spanish one could be represented as:

Example 3.20. RDF representation of Example 3.18 [download file for example 3.20].

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:earl="http://www.w3.org/ns/earl#"
    xmlns:dct="http://purl.org/dc/terms/"
    xmlns:cnt="http://www.w3.org/2008/content#"
    xmlns:http="http://www.w3.org/2006/http#"
    xml:base="http://www.example.org/resource/content_002#">

    <cnt:ContentAsBase64 rdf:ID="content2">
        <cnt:bytes rdf:datatype="http://www.w3.org/2001/XMLSchema#base64Binary"
            >PD94bWwgdmVyc2lvbj...</cnt:bytes>
    </cnt:ContentAsBase64>

    <http:Response rdf:ID="response2">
```

```
        <http:httpVersion>1.1</http:httpVersion>
        <http:statusCodeNumber>200</http:statusCodeNumber>
        <http:sc rdf:resource="http://www.w3.org/2008/http-statusCodes#200" />
        <http:reasonPhrase>OK</http:reasonPhrase>
        <http:headers rdf:parseType="Collection">
            <http:MessageHeader>
                <http:fieldName>Vary</http:fieldName>
                <http:hdrName rdf:resource="http://www.w3.org/2008/http-headers#vary" />
                <http:fieldValue>Accept-Language</http:fieldValue>
            </http:MessageHeader>
            <!-- ... -->
        </http:headers>
        <http:body rdf:resource="#content2" />
    </http:Response>

    <http:Connection rdf:ID="connection2">
        <http:connectionAuthority>www.example.org:80</http:connectionAuthority>
        <http:requests rdf:parseType="Collection">
            <http:Request rdf:resource="#request2" />
        </http:requests>
    </http:Connection>

    <http:Request rdf:ID="request2">
        <http:httpVersion>1.1</http:httpVersion>
        <http:methodName>GET</http:methodName>
        <http:mthd rdf:resource="http://www.w3.org/2008/http-methods#GET" />
        <http:abs_path>/resource/index.html</http:abs_path>
        <http:headers rdf:parseType="Collection">
            <http:MessageHeader>
                <http:fieldName>Accept-Language</http:fieldName>
                <http:hdrName rdf:resource="http://www.w3.org/2008/http-headers#accept-language" />
                <http:fieldValue>es</http:fieldValue>
            </http:MessageHeader>
            <!-- ... -->
        </http:headers>
        <http:resp rdf:resource="#response2" />
    </http:Request>

</rdf:RDF>
```

Strictly speaking, for the representation of the `TestSubject`, only the `http:Response` object is needed. However, it is recommended to use the `http:Request` and `http:Connection` objects to facilitate the replicability of the results. The replicability of the results is also time-dependent as the resources may change over time. Therefore, timestamps or modification dates in the reports are also recommended.

We are now in the situation to allow our Cool Tool accessibility checker (see Example 3.14) to produce accurate reports on both versions of the page. The evaluation report for the English resource (assuming the same test requirement of Example 3.15) looks like the following snippet:

Example 3.21. Evaluation report for the English XHTML resource [download file for example 3.21].

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:earl="http://www.w3.org/ns/earl#"
  xmlns:dct="http://purl.org/dc/terms/"
  xmlns:cnt="http://www.w3.org/2008/content#"
  xmlns:ptr="http://www.w3.org/2009/pointers#"
  xml:base="http://www.example.org/earl/report1#">

  <earl:Assertion rdf:ID="assert">
    <earl:result rdf:resource="result" />
    <earl:test rdf:resource="http://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/F65" />
    <earl:subject rdf:resource="http://www.example.org/resource/content_001#response1" />
    <earl:assertedBy rdf:resource="http://example.org/cooltool/" />
  </earl:Assertion>

  <earl:Software rdf:about="http://example.org/cooltool/">
    <dct:title xml:lang="en">Cool Tool accessibility checker</dct:title>
    <dct:hasVersion>1.0.c</dct:hasVersion>
    <dct:description xml:lang="en">A reliable compliance checker for Web Accessibility</dct:description>
  </earl:Software>

  <earl:TestCase rdf:about="http://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/F65">
    <dct:title xml:lang="en">Failure of Success Criterion 1.1.1 from WCAG 2.0</dct:title>
    <dct:description xml:lang="en">Failure due to omitting the alt attribute on img elements, area elements, and input elements of typ
  </earl:TestCase>

  <ptr:LineCharPointer rdf:ID="pointer">
    <ptr:lineNumber>15</ptr:lineNumber>
    <ptr:charNumber>5</ptr:charNumber>
    <ptr:reference rdf:resource="http://www.example.org/resource/content_001#content1a" />
  </ptr:LineCharPointer>

  <earl:TestResult rdf:ID="result">
    <earl:pointer rdf:resource="#pointer" />
    <earl:outcome rdf:resource="http://www.w3.org/ns/earl#passed" />
  </earl:TestResult>

</rdf:RDF>
```

And the evaluation report for the Spanish resource looks like the following:

Example 3.22. Evaluation report for the Spanish XHTML resource [download file for example 3.22].

```
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:earl="http://www.w3.org/ns/earl#"
  xmlns:dct="http://purl.org/dc/terms/"
  xmlns:cnt="http://www.w3.org/2008/content#"
  xmlns:ptr="http://www.w3.org/2009/pointers#"
  xml:base="http://www.example.org/earl/report2#">

  <earl:Assertion rdf:ID="assert">
    <earl:result rdf:resource="result" />
    <earl:test rdf:resource="http://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/F65" />
    <earl:subject rdf:resource="http://www.example.org/resource/content_002#response2" />
    <earl:assertedBy rdf:resource="http://example.org/cooltool/" />
  </earl:Assertion>

  <earl:Software rdf:about="http://example.org/cooltool/">
    <dct:title xml:lang="en">Cool Tool accessibility checker</dct:title>
    <dct:hasVersion>1.0.c</dct:hasVersion>
    <dct:description xml:lang="en">A reliable compliance checker for Web Accessibility</dct:description>
  </earl:Software>

  <earl:TestCase rdf:about="http://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/F65">
    <dct:title xml:lang="en">Failure of Success Criterion 1.1.1 from WCAG 2.0</dct:title>
    <dct:description xml:lang="en">Failure due to omitting the alt attribute on img elements, area elements, and input elements of typ
  </earl:TestCase>

  <ptr:LineCharPointer rdf:ID="pointer">
    <ptr:lineNumber>16</ptr:lineNumber>
    <ptr:charNumber>9</ptr:charNumber>
    <ptr:reference rdf:resource="http://www.example.org/resource/content_002#content2a" />
  </ptr:LineCharPointer>

  <earl:TestResult rdf:ID="result">
    <earl:pointer rdf:resource="#pointer" />
    <earl:outcome rdf:resource="http://www.w3.org/ns/earl#failed" />
  </earl:TestResult>

</rdf:RDF>
```

Notice how both the result and the location of the element analyzed (in this case the `<img>` element in the page) is different in both reports.

## 3.5 Advance usage

This section presents some advanced use of the vocabularies. In particular, we will show an example demonstrating the extensibility of the vocabulary (without losing its interoperability) and another example showing how to merge reports from different sources.

### 3.5.1 Extending the vocabularies

Let us assume a software product (Cool Validator 2.0) that validates XML documents on the Web against given DTDs or XML Schemas. According to the XML specification [XML], there are two types of errors:

Fatal errors
: Errors after which the parser must not continue processing. Typically, these are well-formedness problems.

Errors
: Violations of the specification. These are normally violations of the validity constraints.

The product defines an additional category, warning, which are errors reported by the underlying SAX parser. These are basically violations not included in the XML specification, and allow the product to continue its normal processing work. With these elements in mind, the following RDF Schema was developed:

Example 4.1. RDF Schema in the namespace `http://example.org/ns/xmlval#` for the error extensions of Cool Validator, which contains new classes, extensions of `earl:Fail` [download file for example 4.1].

```
    <rdfs:Class rdf:ID="FatalError">
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> Fatal error when processing the XML file (well-formedness)
        <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> 1.0</owl:versionInfo>
        <rdfs:subClassOf rdf:resource="http://www.w3.org/ns/earl#Fail" />
    </rdfs:Class>
    <rdfs:Class rdf:ID="Error">
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> Error when processing the XML file (validation constraint)
        <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> 1.0</owl:versionInfo>
        <rdfs:subClassOf rdf:resource="http://www.w3.org/ns/earl#Fail" />
    </rdfs:Class>
    <rdfs:Class rdf:ID="Warning">
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> Warning when processing the XML file (parser issues)</rdfs
        <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> 1.0</owl:versionInfo>
        <rdfs:subClassOf rdf:resource="http://www.w3.org/ns/earl#Fail" />
    </rdfs:Class>
```

A user of the aforementioned validator defines her own XML Schema (see Example 4.2) for an e-commerce application. The schema defines some restrictions in an order element, against which running Web Services payloads must be verified. To facilitate this process and provide via the Web Service a more user-friendly error feedback to her customers, she uses this validator.

Example 4.2. XML Schema for the ordering Web Service [download file for example 4.2].

```
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema" elementFormDefault = "qualified">

    <xsd:element name = "order">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "item" maxOccurs = "unbounded"/>
            </xsd:sequence>
            <xsd:attribute name = "orderid" use = "required" type = "xsd:ID"/>
            <xsd:attribute name = "customer" use = "required" type = "xsd:integer"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "item">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "quantity"/>
                <xsd:element ref = "unitprice"/>
            </xsd:sequence>
            <xsd:attribute name = "itemid" type = "xsd:ID"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "quantity" type = "xsd:unsignedLong"/>
    <xsd:element name = "unitprice">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base = "xsd:float">
                    <xsd:attribute name = "currency" use = "required" type = "currencyType"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:simpleType name = "currencyType">
        <xsd:restriction base = "xsd:string">
            <xsd:enumeration value = "euros"/>
            <xsd:enumeration value = "dollars"/>
            <xsd:enumeration value = "pounds"/>
        </xsd:restriction>
    </xsd:simpleType>

</xsd:schema>
```

Customer X sends as a SOAP payload the following order:

Example 4.3. SOAP payload for Customer X [download file for example 4.3].

```
<order xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation = "order.xsd"
    orderid = "oj_384" customer = "12345">

    <item itemid = "cat_34894">
        <quantity>2</quantity>
        <unitprice currency = "dollars">40.88</unitprice>
    </item>
</order>
```

Which is evaluated through the Cool Validator, producing the following EARL report:

Example 4.4. First XML validation report [download file for example 4.4].

```
<earl:Software rdf:about="http://example.org/coolvalidator/20/">
        <dct:title xml:lang="en">Cool Validator</dct:title>
        <dct:hasVersion>2.0</dct:hasVersion>
        <dct:description xml:lang="en">The best XML validator of the world.</dct:description>
    </earl:Software>

    <earl:TestCase rdf:about="http://example.org/customers/schemas/order.xsd">
        <dct:title xml:lang="en">Ordering Web Service Schema</dct:title>
    </earl:TestCase>

    <earl:TestResult rdf:about="#result">
        <earl:info>The end-tag for element type "quantity" must end with a '&gt;' delimiter.</earl:info>
        <earl:pointer rdf:resource="#pointer" />
        <earl:outcome rdf:resource="http://example.org/ns/xmlval#FatalError" />
    </earl:TestResult>

    <ptr:LineCharPointer rdf:ID="pointer">
        <ptr:charNumber>9</ptr:charNumber>
        <ptr:lineNumber>7</ptr:lineNumber>
        <ptr:reference rdf:resource="#order" />
    </ptr:LineCharPointer>
```

This customer is aware of the EARL extensions of the Cool Validator, and can interpret the results from the perspective of the XML specification, correcting accordingly her SOAP client. Customer Y, who sent the following payload:

Example 4.5. SOAP payload for Customer Y [download file for example 4.5].

```
    <order xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation = "order.xsd"
    orderid = "oj_490" customer = "67890">

    <item itemid = "cat_30922">
        <quantity>4.0</quantity>
        <unitprice currency = "euro">783.30</unitprice>
    </item>
</order>
```

cannot interpret this extension of the vocabulary sent in another report. However, by supporting the EARL standard and standard subclassing mechanisms of Semantic Web vocabularies, this customer is still in the position of interpreting the outcome of the error messages and can act accordingly.

Example 4.6. Second XML validation report translated to standard EARL [download file for example 4.6].

```
<earl:TestResult rdf:ID="result1">
        <earl:pointer rdf:resource="#pointer1" />
        <earl:outcome rdf:resource="http://www.w3.org/ns/earl#Fail" />
        <earl:info>The value '4.0' of element 'quantity' is not valid.</earl:info>
    </earl:TestResult>

    <earl:TestResult rdf:ID="result2">
        <earl:pointer rdf:resource="#pointer2" />
        <earl:outcome rdf:resource="http://www.w3.org/ns/earl#Fail" />
        <earl:info>The value 'euro' of attribute 'currency' on element 'unitprice' is not valid with respect to its type, 'currencyTyp
    </earl:TestResult>

    <ptr:LineCharPointer rdf:ID="pointer1">
        <ptr:charNumber>33</ptr:charNumber>
        <ptr:lineNumber>6</ptr:lineNumber>
        <ptr:reference rdf:resource="#order" />
    </ptr:LineCharPointer>

    <ptr:LineCharPointer rdf:ID="pointer2">
        <ptr:charNumber>38</ptr:charNumber>
        <ptr:lineNumber>7</ptr:lineNumber>
        <ptr:reference rdf:resource="#order" />
    </ptr:LineCharPointer>
```

## 3.5.2 Merging reports from different sources

Using EARL, reports from different sources can be combined to obtain more information or refine existing ones. We take as starting point an XHTML file, which contains two images. One of them lacks of an alternative text attribute. In the other one, the attribute is present, but it reflects the size of the image in bytes.

Example 4.7. An XHTML document to be tested [download file for example 4.7].

```
    <html lang="en" xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
  <title>My photo album</title>
  </head>
  <body>
  <h1>My photo album<</h1>
  <p>These are two nice photos I took yesterday:</p>
  <ul>
  <li>Image of a cat who likes
    <acronym title="Evaluation and Report Language">EARL</acronym>, although it seems quite tired:
    <img src="../images/cat.jpg" />
  </li>
  <li>Image of a fir tree:
    <img src="../images/fir_tree.jpg" alt="98211 bytes" />
  </li>
  </ul>
  </body>
</html>
```

An accessibility evaluator who wants to verify the compliance of this page against success criteria 1.1.1 from WCAG 2.0 [WCAG20] is using for its accessibility test two tools:

- The already known Cool Tool checker (see Example 3.14) and
- The Exemplary Compliance checker (see Example 4.8 below).

Example 4.8. Exemplary Compliance as a Software assertor [download file for example 4.8].

```
    <earl:Software rdf:about="http://example.org/excompliance/">
        <dct:title xml:lang="en">Exemplary Compliance checker</dct:title>
        <dct:hasVersion>3.2</dct:hasVersion>
        <dct:description xml:lang="en">The compliance checker for Web Accessibility</dct:description>
    </earl:Software>
```

The selected tools test, among others, the following WCAG 2.0 techniques:

- Common Failure F65
- Common Failure F30

The Cool Tool checker provides the following report:

Example 4.9. Extract from the Cool Tool report [download file for example 4.9].

```
    <earl:TestResult rdf:ID="result1">
        <earl:pointer rdf:resource="#pointer1" />
        <earl:outcome rdf:resource="http://www.w3.org/ns/earl#failed" />
    </earl:TestResult>
    <earl:TestResult rdf:ID="result2">
```

```
        <earl:pointer rdf:resource="#pointer2" />
        <earl:outcome rdf:resource="http://www.w3.org/ns/earl#cantTell" />
    </earl:TestResult>

    <ptr:LineCharPointer rdf:ID="pointer1">
        <ptr:lineNumber>17</ptr:lineNumber>
        <ptr:charNumber>5</ptr:charNumber>
        <ptr:reference rdf:resource="http://example.org/resource/index.html" />
    </ptr:LineCharPointer>
    <ptr:LineCharPointer rdf:ID="pointer2">
        <ptr:lineNumber>20</ptr:lineNumber>
        <ptr:charNumber>5</ptr:charNumber>
        <ptr:reference rdf:resource="http://example.org/resource/index.html" />
    </ptr:LineCharPointer>
```

In this section of the report, we can observe that the tool is able to identify correctly the error in the first image, but it is unable to discern whether the alternative attribute of the second image corresponds to its size. However, the Exemplary Compliance checker is able to download the image, check its size, and compare it to the content of the alternative attribute. This tool produces the following report:

Example 4.10. Extract from the Exemplary Compliance checker report [download file for example 4.10].

```
    <earl:TestResult rdf:ID="result1">
        <earl:pointer rdf:resource="#pointer1" />
        <earl:outcome rdf:resource="http://www.w3.org/ns/earl#failed" />
    </earl:TestResult>
    <earl:TestResult rdf:ID="result2">
        <earl:pointer rdf:resource="#pointer2" />
        <earl:outcome rdf:resource="http://www.w3.org/ns/earl#failed" />
    </earl:TestResult>

    <ptr:LineCharPointer rdf:ID="pointer1">
        <ptr:lineNumber>17</ptr:lineNumber>
        <ptr:charNumber>5</ptr:charNumber>
        <ptr:reference rdf:resource="http://example.org/resource/index.html" />
    </ptr:LineCharPointer>
    <ptr:LineCharPointer rdf:ID="pointer2">
        <ptr:lineNumber>20</ptr:lineNumber>
        <ptr:charNumber>5</ptr:charNumber>
        <ptr:reference rdf:resource="http://example.org/resource/index.html" />
    </ptr:LineCharPointer>
```

Finally, our evaluator creates a new assertor group, which members include the evaluator and the two tools. The report that she delivers to her customer contains only the assertions that are final, substituting the undefined outcomes by those from the tool that is able to verify adequately the technique. Our evaluator can take decisions on this regard because the use of the EARL Pointers vocabulary allows her to compare exactly the location of the assertion.

Example 4.11. Extract from the final accessibility report [download file for example 4.11].

```
    <foaf:Group rdf:ID="assertgroup">
        <dct:title>John Doe and the W3C HTML Validator</dct:title>
        <earl:mainAssertor rdf:resource="http://example.org/persons/jdoe/" />
        <foaf:member rdf:resource="http://example.org/excompliance/" />
        <foaf:member rdf:resource="http://example.org/cooltool/" />
    </foaf:Group>
    <foaf:Person rdf:about="http://example.org/persons/jdoe/">
        <foaf:mbox rdf:resource="mailto:jane@example.org" />
        <foaf:name>Jane Doe</foaf:name>
    </foaf:Person>
    <earl:Software rdf:about="http://example.org/cooltool/">
        <dct:title xml:lang="en">Cool Tool accessibility checker</dct:title>
        <dct:hasVersion>1.0.c</dct:hasVersion>
        <dct:description xml:lang="en">A reliable compliance checker for Web Accessibility</dct:description>
    </earl:Software>
    <earl:Software rdf:about="http://example.org/excompliance/">
        <dct:title xml:lang="en">Exemplary Compliance checker</dct:title>
        <dct:hasVersion>3.2</dct:hasVersion>
        <dct:description xml:lang="en">The compliance checker for Web Accessibility</dct:description>
    </earl:Software>

    <earl:TestCase rdf:about="http://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/F65">
        <dct:description xml:lang="en">Failure due to omitting the alt attribute on img elements, area elements, and input elements of
        <dct:title xml:lang="en">Failure of Success Criterion 1.1.1 from WCAG 2.0</dct:title>
    </earl:TestCase>
    <earl:TestCase rdf:about="http://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/F30">
        <dct:description xml:lang="en">Failure of Success Criterion 1.1.1 and 1.2.1 due to using text alternatives that are not altern
        <dct:title xml:lang="en">Failure of Success Criterion 1.1.1 and 1.2.1 from WCAG 2.0</dct:title>
    </earl:TestCase>

    <earl:TestResult rdf:ID="result1">
        <earl:pointer rdf:resource="#pointer1" />
        <earl:outcome rdf:resource="http://www.w3.org/ns/earl#failed" />
    </earl:TestResult>
    <earl:TestResult rdf:ID="result2">
        <earl:pointer rdf:resource="#pointer2" />
        <earl:outcome rdf:resource="http://www.w3.org/ns/earl#failed" />
    </earl:TestResult>

    <ptr:LineCharPointer rdf:ID="pointer1">
        <ptr:lineNumber>17</ptr:lineNumber>
        <ptr:charNumber>5</ptr:charNumber>
        <ptr:reference rdf:resource="http://example.org/resource/index.html" />
    </ptr:LineCharPointer>
    <ptr:LineCharPointer rdf:ID="pointer2">
        <ptr:lineNumber>20</ptr:lineNumber>
        <ptr:charNumber>5</ptr:charNumber>
        <ptr:reference rdf:resource="http://example.org/resource/index.html" />
    </ptr:LineCharPointer>
```

This example demonstrates how the use of simple Semantic Web technologies enables the combination of EARL assertions to produce improved and more accurate reports.

## 4. Conformance for EARL 1.0 Tools and Reports

[Editor's note: ERT WG is looking for feedback on this entire section.]

This section defines conformance requirements for software tools and processes, to ensure a consistent implementation and exchange of the EARL 1.0 vocabulary. The following applies to tools conforming with EARL 1.0:

A. Conforming EARL 1.0 reports adhere to the requirements listed in 4.1 Conforming EARL 1.0 Reports
B. Software tools that produce conforming EARL 1.0 reports can provide them in valid RDF/XML notation
C. Software tools that process conforming EARL 1.0 reports can accept them in valid RDF/XML notation

### 4.1 Conforming EARL 1.0 Reports

Conforming EARL 1.0 reports are valid RDF graphs with:

1. At least one Assertion
2. Exactly one Assertor, referenced by earl:assertedBy, for each Assertion
    a. Exactly one identifying name (per language), referenced by dct:title, foaf:name, or doap:name for each Assertor
    b. At most one description (per language), referenced by dct:description or doap:description for each Assertor
    c. Any number of attributes, referenced by foaf:nick, foaf:mbox, or foaf:homepage for each Assertor that is also of type foaf:Agent
    d. Any number of members, referenced by foaf:member, for each Assertor that is also of type foaf:Group
    e. At most one main assertor, referenced by earl:mainAssertor, for each Assertor that is also of type foaf:Group
3. Exactly one Test Subject, referenced by earl:subject, for each Assertion
    a. Exactly one identifying title (per language), referenced by dct:title, foaf:name, or doap:name for each Test Subject
    b. At most one description (per language), referenced by dct:description or doap:description for each Test Subject
    c. At most one date (as defined by XML Datatypes), referenced by dct:date, for each Test Subject
    d. Any number of relationships, referenced by dct:hasPart or dct:isPartOf, between any instances of Test Subject
4. Exactly one Test Criterion, referenced by earl:test, for each Assertion
    a. Exactly one identifying title (per language), referenced by dct:title, for each Test Criterion
    b. At most one description (per language), referenced by dct:description or doap:description for each Test Criterion
    c. Any number of relationships, referenced by dct:hasPart or dct:isPartOf, between any instances of Test Criterion
5. Exactly one Test Result, referenced by earl:result, for each Assertion
    a. Exactly one date (as defined by XML Datatypes), referenced by dct:date, for each Test Result
    b. Exactly one Outcome Value, referenced by earl:outcome, for each Test Result
        i. Exactly one identifying title (per language), referenced by dct:title, for each Outcome Value
        ii. Exactly one description (per language), referenced by dct:description, for each Outcome Value
    c. At most one identifying title (per language), referenced by dct:title, for each Test Result
    d. At most one description (per language), referenced by dct:description or doap:description for each Test Result
    e. At most one additional information (per language), referenced by earl:info for each Test Result
    f. Any number of pointer methods, referenced by earl:pointer for each Test Result
6. At most one Test Mode, referenced by earl:mode, for each Assertion
    a. Exactly one identifying title (per language), referenced by dct:title, for each Test Mode
    b. Exactly one description (per language), referenced by dct:description, for each Test Mode
7. Exactly one identifying name (per language), referenced by doap:name, for each Software
8. Conforming HTTP-in-RDF graphs, for each Test Subject that is also of type http:Response
9. Conforming Content-in-RDF graphs, for each Test Subject that is also of type cnt:Content

Note: subclasses or subproperties of terms share the same type. They are therefore considered to be equivalent entities in adhering to any of the above requirements. Also, instances in multiple languages of the same entity (such as title or description) are considered to be a single occurrence of the entity.

In addition, it is strongly recommended that EARL 1.0 reports are also valid RDF graphs with:

1. Each Assertor is also one of the following types:
    - earl:Software
    - foaf:Agent
    - foaf:Person
    - foaf:Organization
    - foaf:Group
2. Each Test Subject is also one of the following types:
    - earl:Software
    - cnt:Content
    - http:Response
    - foaf:Document
3. Each Test Criterion is also one of the following types:
    - earl:TestRequirement
    - earl:TestCase
4. Each Test Mode is one of the following instances:
    - earl:automatic
    - earl:manual
    - earl:semiAuto
    - earl:undisclosed
    - earl:unknownMode

5. Each Outcome Value is one of the following instances:
   - earl:passed
   - earl:failed
   - earl:cantTell
   - earl:inapplicable
   - earl:untested

   Or is an instance of one of the following classes (or sublcases thereof):
   - earl:Pass
   - earl:Fail
   - earl:CannotTell
   - earl:NotApplicable
   - earl:NotTested

### 4.2 Conforming HTTP-in-RDF Graphs

Conforming HTTP-in-RDF graphs are valid RDF graphs with:

1. Exactly one connection authority, specified by http:connectionAuthority, for each Connection
2. At most one RDF collection, referenced by http:requests, with any number of Request instances, for each Connection
3. Exactly one HTTP version, specified by http:httpVersion, for each Message
4. At most one RDF collection, referenced by http:headers, with any number of Message Header instances, for each Message
5. Exactly one message body, referenced by http:body, for each Message
6. At most one date, specified by dct:date, for each Message
7. Exactly one method name, specified by http:methodName, for each Request
8. Exactly one request URI, specified by http:requestURI, for each Request
9. At most one Method, referenced by http:mthd, for each Request
10. At most one Response, referenced by http:resp, for each Request
11. Exactly one status code value, specified by http:statusCodeValue, for each Response
12. Exactly one reason phrase, specified by http:reasonPhrase, for each Response
13. At most one Status Code, referenced by http:sc, for each Response
14. Exactly one field name, specified by http:fieldName, for each Message Header
15. Exactly one field value, specified by http:fieldValue, for each Message Header
16. At most one Header Name, referenced by http:hdrName, for each Message Header
17. At most one RDF collection, referenced by http:headerElements, with any number of Header Element instances, for each Message Header
18. Exactly one header element name, specified by http:elementName, for each Header Element
19. At most one header element value, specified by http:elementValue, for each Header Element
20. At most one RDF collection, referenced by http:params, with any number of Parameter instances, for each Header Element
21. Exactly one parameter name, specified by http:paramName, for each Parameter
22. Exactly one parameter value, specified by http:paramValue, for each Parameter

### 4.3 Conforming Content-in-RDF Graphs

Conforming Content-in-RDF graphs are valid RDF graphs with:

1. At most one character encoding, specified by cnt:characterEncoding, for each Content
2. Any number of relationships, referenced by dct:hasFormat or dct:isFormatOf, between any instances of Content
3. Exactly one byte sequence, specified by cnt:bytes, for each ContentAsBase64
4. Exactly one character sequence, specified by cnt:chars, for each ContentAsText
5. Exactly one XML rest, specified by cnt:rest, for each ContentAsXML
6. At most one leadingMisc, specified by cnt:leadingMisc, for each ContentAsXML
7. At most one document type delcaration, referenced by cnt:dtDecl, for each ContentAsXML
8. Exactly one XML version, specified by cnt:version, for each ContentAsXML
9. At most one XML character encoding, specified by cnt:declaredEncoding, for each ContentAsXML
10. At most one XML standalone declaration, specified by cnt:standalone, for each ContentAsXML
11. Exactly one document type name, specified by cnt:doctypeName, for each DoctypeDecl
12. At most one public identifier, specified by cnt:publicId, for each DoctypeDecl
13. At most one system identifier, specified by cnt:systemId, for each DoctypeDecl
14. At most one internal subset, specified by cnt:internalSubset, for each DoctypeDecl

## 5. Serializations of EARL Reports

Note: this section will be added to refer the reader to best practices and existing references in RDF/XML serializations (possibly providing a DTD or XML Schema for EARL); RDF->JSON conversion (in particular if we do end up providing an XML Schema or DTD); binary RDF (work in progress at W3C); or other formats that may be useful to tool developers.

## Appendix A: References

[Content-RDF]
Representing Content in RDF. http://www.w3.org/TR/Content-in-RDF10/

[DCMISCHEMAS]
DCMI term declarations represented in RDF schema language. http://dublincore.org/schemas/rdfs/

[EARL-Schema]
Evaluation and Report Language 1.0 Schema. http://www.w3.org/TR/EARL10-Schema/

[FOAF]
FOAF Vocabulary Specification 0.91. Namespace Document 2 November 2007 - OpenID Edition. http://xmlns.com/foaf/spec/

[HTTP-RDF]
HTTP Vocabulary in RDF. http://www.w3.org/TR/HTTP-in-RDF10/

[IEEE-829]
IEEE Standard for Software Test Documentation (IEEE Std 829-1998). ISBN 0-7381-1444-8 SS94687. http://ieeexplore.ieee.org/servlet/opac?punumber=5976

[ISO15836]
Information and documentation - The Dublin Core metadata element set. ISO 15836:2009. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=52142

[NISOZ3985]

ANSI/NISO Z39.85 - The Dublin Core Metadata Element Set. NISO, May 2007. http://www.niso.org/standards/z39-85-2007/
[Pointers-RDF]
Pointer Methods in RDF. http://www.w3.org/TR/Pointers-in-RDF10/
[RDF]
Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, 22 February 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
[RDF-PRIMER]
RDF Primer. W3C Recommendation, 10 February 2004. http://www.w3.org/TR/rdf-primer/
[RDFS]
RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, 10 February 2004. http://www.w3.org/TR/rdf-schema/
[RDF-XML]
RDF/XML Syntax Specification (Revised). W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-syntax-grammar/
[RDF-XML-DIFFS]
Why RDF model is different from the XML model. Paper by Tim Berners-Lee, September 1998. http://www.w3.org/DesignIssues/RDF-XML
[RFC2119]
Key words for use in RFCs to Indicate Requirement Levels. IETF RFC, March 1997. http://www.ietf.org/rfc/rfc2119.txt
[RFC5013]
The Dublin Core Metadata Element Set. IETF RFC, August 2007. http://www.ietf.org/rfc/rfc5013.txt
[OWL]
OWL Web Ontology Language. W3C Recommendation, 10 February 2004. http://www.w3.org/TR/owl-features/
[WCAG10]
Web Content Accessibility Guidelines 1.0. W3C Recommendation, 5 May 1999. http://www.w3.org/TR/WCAG10/
[WCAG20]
Web Content Accessibility Guidelines 2.0. W3C Recommendation, 11 December 2008. http://www.w3.org/TR/WCAG20/
[XML]
Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation 26 November 2008. http://www.w3.org/TR/REC-xml/
[URI]
Uniform Resource Identifier (URI): Generic Syntax. IETF RFC, January 2005. http://www.ietf.org/rfc/rfc3986.txt
[DOAP]
[HTTP]
[DCMI]
[RFC5013]
[CNT]
[PTRS]
[XMLS]

## Appendix D: Contributors

Shadi Abou-Zahra, Carlos Iglesias, Michael A Squillace, Johannes Koch and Carlos A Velasco.

## Appendix C: Document changes

The following is a list of changes with respect to the previous internal version:

- Adopted common definitions of EARL.
- Disambiguated the links to examples.
- Minor editorial corrections.
- Examples clarified.