



SEVENTH FRAMEWORK PROGRAMME  
INFORMATION AND COMMUNICATION TECHNOLOGIES

Project:  
Accessibility Assessment Simulation Environment for New  
Applications Design and Development  
(ACCESSIBLE, Grant Agreement No. 224145)



Deliverable number and title:

D 4.3 - A set of guidelines for the validation and integration of the implemented tools and methodologies

Lead beneficiary:	<b>FFCUL</b>
WP. no, title and activity type	WP4 – <b>Services and Content Knowledge Infrastructure</b> (RTD)
Contributing Task (s)	T4.4, T4.2, T4.3, T4.1
Dissemination level	PU - Public
Delivery date	September 2010
Status	Final Draft
File name and size	"D4.3-ACCESSIBLE_FFCUL-WP4-R1-V1.0.DOC", 3855 Kb

*(This page has been left empty in purpose.  
The same stands after each main section.)*

## Authors List

Leading Author ( <i>Editor</i> )				
	<i>Surname</i>	<i>Initials</i>	<i>Beneficiary Name (Short Name)</i>	<i>Contact email</i>
	Lopes	R.	FFCUL	rlopes@di.fc.ul.pt
Co-authors ( <i>In alphabetic order</i> )				
#	<i>Surname</i>	<i>Initials</i>	<i>Beneficiary Name (Short Name)</i>	<i>Contact email</i>
1	Tzovaras	D.	CERTH/ITI	Dimitrios.Tzovaras@iti.gr
2	Votis	K.	CERTH/ITI	kvotis@iti.gr
3	Carriço	L.	FFCUL	lmc@di.fc.ul.pt
4	Bandeira	R.	FFCUL	rbandeira@lasige.di.fc.ul.pt
5	Kalogirou	K.	CERTH/HIT	kalogir@certh.gr

## Peer Reviewers List

#	<i>Surname</i>	<i>Initials</i>	<i>Contact email</i>
1	Kehagias	D.K.	<a href="mailto:diok@iti.gr">diok@iti.gr</a>
2	Grudeva	P.G.	<a href="mailto:petia@marie-curie-bg.org">petia@marie-curie-bg.org</a>



## Executive Summary

The integration of the ACCESSIBLE Content and Knowledge Infrastructure (e.g., the ACCESSIBLE ontology) is crucial for the configurability and personalization characteristics of all the accessibility evaluation and simulation tools created within the context of the ACCESSIBLE project. These provide the brain for all the knowledge modeled into the ACCESSIBLE ontology, as specified in the ACCESSIBLE Harmonized Methodology.

All requirements for the ACCESSIBLE knowledge infrastructure are discussed in the light of their corresponding developments, thus providing a validation of all the methodologies and approaches taken in the course of Work Package 4. More specifically, the meta models will be validated regarding their semantics, through their coverage of different accessibility subjects. The rules inference engine and accompanying tools will be validated against a battery of tests.

Furthermore, this deliverable documents the updates and advances on the ACCESSIBLE Inference Engine, and provides example configurations and clients for the integration facilities provided by this component.



## Table of contents

Authors List .....	3
Peer Reviewers List .....	3
Executive Summary .....	5
Table of contents .....	7
List of figures .....	9
List of tables .....	10
List of abbreviations and acronyms .....	11
1 Introduction .....	13
2 Testing and Validation .....	14
2.1 Generic functional requirements .....	14
2.2 Performance requirements .....	27
2.3 Operational requirements .....	28
2.4 Reliability requirements .....	31
2.5 Maintainability & Interoperability requirements .....	31
3 OntologyQuery Server .....	33
3.1 Introduction and Context .....	33
3.2 Requirements .....	33
3.2.1 Functional Requirements .....	33
3.2.2 Non-functional requirements .....	36
3.3 OntologyQuery Server Component .....	36
3.4 Implementation .....	37
4 OntologyQuery Client .....	39
4.1 Introduction and Context .....	39
4.2 Requirements .....	39
4.2.1 Functional Requirements .....	39
4.3 OntologyQuery Client Component .....	40
4.4 Implementation .....	40
5 OntologyQuery General .....	42
5.1 Requirements .....	42
5.2 General Component .....	43
6 Testing and validation .....	44
6.1 Measurement goals and Criteria .....	44
6.1.1 WP4 Tools' Performance .....	44
6.1.2 Usability of the WP4 Tools .....	45
6.2 Performance evaluation of ACCESSIBLE ontologies .....	46
6.3 Performance Evaluation of Inference Engine on Context Information (ACCESSIBLE ontologies) .....	50
7 Conclusions .....	55
8 References .....	56



## List of figures

Figure 1: ACCESSIBLE ontology .....	15
Figure 2: ACCESSIBLE owl files .....	18
Figure 3: Edition process .....	25
Figure 4: Example of navigation.....	28
Figure 5: Example of navigation.....	29
Figure 6: ACCESSIBLE Rules User Interface .....	30
Figure 7: Example of Rules User Interface GUI .....	31
Figure 8: Starting the OntologyQuery Server.....	34
Figure 9: Querying the Ontology .....	35
Figure 10: OntologyQuery Server Architecture .....	36
Figure 11: OntologyQuery Server Implementation .....	37
Figure 12: OntologyQuery Client Use Case .....	39
Figure 13: OntologyQuery Client Architecture .....	40
Figure 14: OntologyQuery Client Implementation.....	42
Figure 15: OntologyQuery General Implementation.....	43
Figure 16: Using the Inference engine to assess test set adequacy and performance..	50

## List of tables

Table 1: Response time for loading GenericOntology.owl .....	46
Table 2: Response time for loading ScreenMagnifiers.owl.....	47
Table 3: Response time for loading DescriptionLanguage.owl.....	47
Table 4: Response time for loading ScreenReader.owl.....	47
Table 5: Response time for loading HTML.owl.....	48
Table 6: Response time for loading SpeechDevices.owl.....	48
Table 7: Response time for loading TextBrowsers.owl.....	48
Table 8: Response time for loading ListeningDevices.owl .....	48
Table 9: Response time for loading WCAG2.owl.....	49
Table 10: Response time for loading MWBP.owl.....	49
Table 11: Response time for loading WebService1.owl.....	49
Table 12: Response time for loading ScanningSoftware.owl.....	49
Table 13: Summary of query response time .....	53

## List of abbreviations and acronyms

*(in alphabetic order)*

JVM	Java Virtual Machine
OWL	Web Ontology Language
RDF	Resource Description Framework
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Query Language for RDF
SWRL	Semantic Web Rules Language



# 1 Introduction

The ACCESSIBLE Content and Knowledge Infrastructure (e.g., the ACCESSIBLE ontology) is a crucial component in the configurability and personalisation characteristics for all the tools created within the context of the ACCESSIBLE project, as specified by the ACCESSIBLE Harmonized Methodology.

This deliverable presents the results from the ACCESSIBLE Content and Knowledge Infrastructure, as thoroughly detailed in previous deliverables, D4.1 [1] and D4.2 [2].

Therefore, the goal of this deliverable is centred on ensuring that the ACCESSIBLE Ontology and associated tools have been designed and implemented according to the goals and requirements for the ACCESSIBLE Content and Knowledge Infrastructure, as delineated in Work Package 2 Deliverable D2.2 [3].

Furthermore, we present a new version of the ACCESSIBLE Inference Engine, hereinafter referred as OntologyQuery. This new version tackles the improvement of the Inference Engine detailed in Deliverable D4.2, as well as how its inner architecture can be leveraged towards its integration into other services.

The next Sections of this deliverable present an analysis of each requirement and how it has been fulfilled within Work Package 4 and associated Work Packages (e.g., WP3), and will also detail on the particular aspects of the inference engine.

## 2 Testing and Validation

In Deliverable D2.2 [3], a set of requirements were devised for the ACCESSIBLE Ontological Knowledge Resource, i.e., all the components specified and implemented in the context of Work Package 4. In the light of these requirements, we present next how they have been accomplished.

### 2.1 Generic functional requirements

*G-REQ3-1: The ontology will support the OWL Web Ontology Language.*

The ACCESSIBLE ontology was specified using Semantic Web technologies, including RDF, RDF-S, and OWL.

An example excerpt of the ACCESSIBLE ontology is presented below, where RDF, RDF-S, and OWL namespaces appear:

```
<owl:Class rdf:ID="Application"/>
  <owl:ObjectProperty rdf:ID="User_linksTo_Approach">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="Approach_linksTo_User"/>
    </owl:inverseOf>
    <rdfs:range rdf:resource="#Approach"/>
    <rdfs:domain rdf:resource="#User"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="Device_belongsTo_Disability">
    <rdfs:domain rdf:resource="#Device"/>
    <rdfs:range rdf:resource="#Disability"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="Disability_has_Device"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
```

Furthermore, we present below a renderisation of the ACCESSIBLE ontology

within the Protégé ontology modelling tool, detailing the namespaces within the Generic Ontology (including RDFS and OWL namespaces):

**INDIVIDUAL EDITOR** + - F T

For Individual: Ontology(<http://www.AccessibleOntology.com/GenericOntology.owl>) (instance of owl:Ontology, intern...

**Ontology URI**

<http://www.AccessibleOntology.com/GenericOntology.owl>

**Annotations**

Property	Value	Lang
rdfs:comment		

**Default Namespace**

<http://www.AccessibleOntology.com/GenericOntology.owl#>

**Namespace Prefixes**

Prefix	Namespace
xsd	<a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a>
Section508	<a href="http://www.AccessibleOntology.com/Section508.owl#">http://www.AccessibleOntology.com/Section508.owl#</a>
Braille	<a href="http://www.AccessibleOntology.com/Braille.owl#">http://www.AccessibleOntology.com/Braille.owl#</a>
swrlb	<a href="http://www.w3.org/2003/11/swrlb#">http://www.w3.org/2003/11/swrlb#</a>
rdfs	<a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a>
owl	<a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a>
WCAG1	<a href="http://www.AccessibleOntology.com/WCAG1.owl#">http://www.AccessibleOntology.com/WCAG1.owl#</a>
swrla	<a href="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#">http://swrl.stanford.edu/ontologies/3.3/swrla.owl#</a>
abox	<a href="http://swrl.stanford.edu/ontologies/built-ins/3.3/abox.owl#">http://swrl.stanford.edu/ontologies/built-ins/3.3/abox.owl#</a>
WCAG2	<a href="http://www.AccessibleOntology.com/WCAG2.owl#">http://www.AccessibleOntology.com/WCAG2.owl#</a>
query	<a href="http://swrl.stanford.edu/ontologies/built-ins/3.3/query.owl#">http://swrl.stanford.edu/ontologies/built-ins/3.3/query.owl#</a>
MVBP	<a href="http://www.AccessibleOntology.com/MVBP.owl#">http://www.AccessibleOntology.com/MVBP.owl#</a>
WebService1	<a href="http://www.AccessibleOntology.com/WebService1.owl#">http://www.AccessibleOntology.com/WebService1.owl#</a>
DL	<a href="http://www.AccessibleOntology.com/DescriptionLanguage#">http://www.AccessibleOntology.com/DescriptionLanguage#</a>
rdflib	<a href="http://www.w3.org/2000/02/rdf-schema#">http://www.w3.org/2000/02/rdf-schema#</a>

**Figure 1:** ACCESSIBLE ontology

***G-REQ3-2:*** *The ontology shall support the SWRL (a Semantic Web Rule Language) rules, which form of an implication between an antecedent (body) and consequent (head).*

The ACCESSIBLE ontology supports SWRL rules, in order to specify general production rules to establish interconnections between RDF triples.

Below we present an excerpt of a SWRL rule specified in the Generic Ontology:

```
<swrl:Imp rdf:ID="Rule_GuidelineToOutputResult">
  <swrl:head>
    <swrl:AtomList>
      <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
      <rdf:first>
        <swrl:IndividualPropertyAtom>
          <swrl:propertyPredicate
rdf:resource="#Guideline_linksTo_OutputResult"/>
          <swrl:argument1>
            <swrl:Variable rdf:ID="z"/>
          </swrl:argument1>
          <swrl:argument2>
            <swrl:Variable rdf:ID="e"/>
          </swrl:argument2>
        </swrl:IndividualPropertyAtom>
      </rdf:first>
    </swrl:AtomList>
  </swrl:head>
  <swrl:body>
    <swrl:AtomList>
      <rdf:rest>
        <swrl:AtomList>
          <rdf:first>
            <swrl:IndividualPropertyAtom>
              <swrl:argument2>
                <swrl:Variable rdf:ID="d"/>
              </swrl:argument2>
            </swrl:IndividualPropertyAtom>
          </rdf:first>
        </swrl:AtomList>
      </rdf:rest>
    </swrl:AtomList>
  </swrl:body>
</swrl:Imp>
```

***G-REQ3-3:*** *The ontology shall support SPARQL queries that consist of triplets, in order to narrow the information space of accessibility assessment according to specified semantics of usage scenarios.*

The ACCESSIBLE ontology is queryable through SPARQL, since it has been defined with Semantic Web technologies.

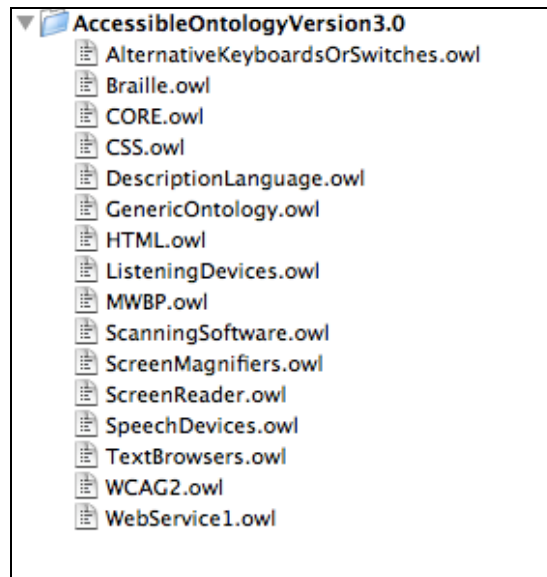
An example SPARQL query to retrieve all impairments specified in the Generic Ontology is presented below:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acc: <http://www.AccessibleOntology.com/GenericOntology.owl#>
SELECT ?impairment
WHERE {
    ?impairment rdf:type acc:Impairment
}
```

***G-REQ3-4:*** *In order to preserve the integrity and the maintenance of the ontology, the ACCESSIBLE ontology based knowledge resource shall include generic and domain-specific ontologies.*

The ACCESSIBLE ontology has been separated into several sub-ontologies, the Generic Ontology, and one Domain-Specific Ontology for each technological domain covered in ACCESSIBLE (Web, Mobile Web, Web services, and SDL), as well as miscellaneous auxiliary ontologies for domain-specific ontologies.

Below we present the ontology breakdown designed for the ACCESSIBLE Ontology:



**Figure 2:** ACCESSIBLE owl files

Furthermore, the inclusion of the Domain-Specific Ontologies within the Generic Ontology is made through the inclusion of namespaces and associated Semantic Web predicates.

***G-REQ3-5:*** *The ACCESSIBLE ontology shall contain a generic version that describes the general structure of accessibility attributes of guidelines, users, devices, and applications.*

The Generic Ontology provides structural elements to describe accessibility assessment domains, as presented next, in an excerpt from the ACCESSIBLE Ontology:

```
<owl:Class rdf:ID="Checkpoint"/>
  <owl:Class rdf:ID="Capability"/>
  <owl:Class rdf:ID="OutputResult"/>
  <owl:Class rdf:ID="Test"/>
  <owl:Class rdf:ID="Device"/>
  <owl:Class rdf:ID="Standard"/>
  <owl:Class rdf:ID="Approach"/>
  <owl:Class rdf:ID="User"/>
  <owl:Class rdf:ID="Guideline"/>
  <owl:Class rdf:ID="Impairment"/>
```

```

<owl:Class rdf:ID="FunctionalLimitation"/>
<owl:Class rdf:ID="Disability"/>
<owl:Class rdf:ID="Technique"/>
<owl:Class rdf:ID="Application"/>
<owl:ObjectProperty rdf:ID="User_linksTo_Approach">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="Approach_linksTo_User"/>
  </owl:inverseOf>
  <rdfs:range rdf:resource="#Approach"/>
  <rdfs:domain rdf:resource="#User"/>
</owl:ObjectProperty>

```

***G-REQ3-6:*** *The domain ontologies shall include different characteristics and instances of accessibility standards, disabilities, functional limitations, devices and application domains.*

Each Domain-Specific Ontology describes standards and/or guidelines for its corresponding technological domain (e.g., WCAG 2.0 for Web technologies), and maps these standards and guidelines into functional limitations or disabilities. These mappings reflect the work devised in Work Package 3, i.e., the ACCESSIBLE Harmonised Methodology.

As an example, this factor was achieved within the WCAG 2.0 ontology by specialising the generic terminology into WCAG 2.0 specific terms, thus bridging together both terminologies successfully:

```

<owl:Class rdf:ID="SuccessCriterion">
  <rdfs:subClassOf
rdf:resource="http://www.AccessibleOntology.com/GenericOntology.owl#C
heckpoint"/>
  </owl:Class>
  <owl:Class rdf:ID="Technique">
    <rdfs:subClassOf
rdf:resource="http://www.AccessibleOntology.com/GenericOntology.owl#T
echnique"/>
  </owl:Class>
  <owl:Class rdf:ID="Guideline">

```

```

    <rdfs:subClassOf
rdf:resource="http://www.AccessibleOntology.com/GenericOntology.owl#G
uideline"/>
  </owl:Class>

  <owl:Class rdf:ID="Test">
    <rdfs:subClassOf
rdf:resource="http://www.AccessibleOntology.com/GenericOntology.owl#T
est"/>
  </owl:Class>

  <owl:Class rdf:ID="Approach">
    <rdfs:subClassOf
rdf:resource="http://www.AccessibleOntology.com/GenericOntology.owl#A
pproach"/>
  </owl:Class>

```

Here we can perceive the terminology bridge between both ontologies, e.g., by mapping the generic term SuccessCriterion into the WCAG-specific term Checkpoint.

Regarding the specificities of ontologies for each domain, this factor was also successfully achieved, by mapping terminology and evaluation guidelines' structures into proper Semantic Web concepts, such as Classes and corresponding instances.

One example detailed below concerns the Checkpoint STYLE\_SHEETS\_SUPPORT from the W3C Mobile Web Best Practices 1.0, and corresponding technique:

```

<j.0:Checkpoint rdf:ID="Checkpoint_48">
  <Checkpoint_has_Technique>
    <j.0:Technique rdf:ID="Technique_48">
      <hasId rdf:datatype="xs:int">48</hasId>
      <hasName
rdf:datatype="xs:string">STYLE_SHEETS_SUPPORT</hasName>
      <Technique_belongsTo_Checkpoint
rdf:resource="#Checkpoint_48"/>
    </j.0:Technique>
  </Checkpoint_has_Technique>
  <hasDescription rdf:datatype="xs:string">Organize documents so
that if necessary they may be read without style
sheets</hasDescription>
  <hasName rdf:datatype="xs:string">Style sheets support</hasName>

```

```
<hasId rdf:datatype="xs:int">48</hasId>
```

Regarding functional limitations, the ontologies properly define these concepts according to the ACCESSIBLE Harmonised Methodology (which, in turn, is based on ICF), and map these into the different accessibility guidelines. Below we present an example for WCAG 2.0 and how the Success Criterion 1.2.3 maps into the Conductive Hearing Loss disability:

```
<FunctionalLimitation_belongsTo_Disability>
  <Disability rdf:ID="Conductive_Hearing_Loss">
    <Disability_belongsTo_Checkpoint>
      <rdf:Description
rdf:about="http://www.AccessibleOntology.com/WCAG2.owl#SuccessCriteri
on_1.2.3">
        <Checkpoint_has_Disability
rdf:resource="#Deaf_Blindness"/>
      </rdf:Description>
    </Disability_belongsTo_Checkpoint>
    <Disability_belongsTo_Checkpoint>
      <rdf:Description
rdf:about="http://www.AccessibleOntology.com/WCAG2.owl#SuccessCriteri
on_1.2.4">
        <Checkpoint_has_Disability
rdf:resource="#Deaf_Blindness"/>
      </rdf:Description>
    </Disability_belongsTo_Checkpoint>
    <Disability_belongsTo_Checkpoint>
      <rdf:Description
rdf:about="http://www.AccessibleOntology.com/WCAG2.owl#SuccessCriteri
on_1.2.2">
        <Checkpoint_has_Disability
rdf:resource="#Deaf_Blindness"/>
      </rdf:Description>
    </Disability_belongsTo_Checkpoint>
    <Disability_has_FunctionalLimitation>
      <FunctionalLimitation rdf:ID="Sound_Discrimination">
        <hasName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Sound discrimination</hasName>
        <hasDescription
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
>Sensory functions relating to sensing the presence of
sound involving the differentiation of ground and binaural synthesis,
separation and blending. </hasDescription>
```

```
<FunctionalLimitation_belongsTo_Disability
rdf:resource="#Conductive_Hearing_Loss"/>
```

***G-REQ3-7: All the domain ontologies shall comply with the generic ontology.***

Each Domain-Specific Ontology follows the OWL structures (e.g., Classes, Properties, etc.) defined in the Generic Ontology, by defining all technology-specific concepts as corresponding OWL instances or subclasses.

We present below a small subset of the Description Languages Ontology, where a specific checkpoint and corresponding guideline association follow the Generic Ontology structures, but specify the domain concepts as instances of the generic classes:

```
<j.0:Checkpoint rdf:ID="Checkpoint_17">
  <hasName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >DL Checkpoint_17</hasName>
  <Checkpoint_belongsTo_Guideline>
    <j.0:Guideline rdf:ID="Guideline_17">
      <hasName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >DL Guideline 17</hasName>
      <Guideline_has_Checkpoint rdf:resource="#Checkpoint_17"/>
      <hasId rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >17</hasId>
    </j.0:Guideline>
  </Checkpoint_belongsTo_Guideline>
  <hasPriority rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >1</hasPriority>
  <hasDescription
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Usage of comboboxes instead of textboxes helping the user to
choose a value from old or proposed values must be used where
possible. </hasDescription>
```

***G-REQ3-8: The ontology shall include widely used web content accessibility guidelines.***

The ACCESSIBLE ontology includes WCAG 2.0, the Mobile Web Best Practices (MWBP) from W3C, the defined guidelines for Web services and Description Languages .

***G-REQ3-9:*** *The ontology shall include the most well known disabilities that people face and for which accessibility guidelines provide clues on how they should be supported in software applications.*

The ACCESSIBLE ontology specifies a thorough set of disabilities based on the ICF categorisation, as well as its mappings into accessibility guidelines, based on the Barrier Walkthrough method. These instances of the ACCESSIBLE ontology were devised according to the information specified in the ACCESSIBLE Harmonised Methodology.

A small sample of the specified disabilities within the ACCESSIBLE Ontology is presented below:

```
<Disability rdf:ID="Extreme_Light_Sensitivity"/>
<Disability rdf:ID="Rett_Syndrome"/>
<Disability rdf:ID="Absent_Limb_Or_Reduced_Limb_Function" />
```

***G-REQ3-10:*** *The ontology shall include a variety of assistive devices, which are used from people with disabilities.*

The ACCESSIBLE ontology specifies a set of assistive devices, and their relationship to disabilities and functional limitations. These are particularly reflected in the ACCESSIBLE accessibility simulation tools. A small example of such specification within the ontology is presented below, where the specification of a Braille display is formalised in RDF/OWL:

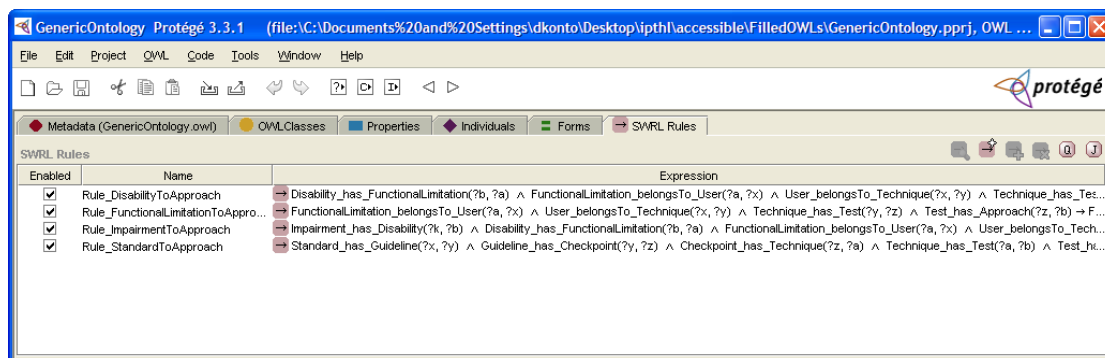
```
<owl:Class rdf:ID="Braille">
  <rdfs:subClassOf
rdf:resource="http://www.AccessibleOntology.com/GenericOntology.owl#Device"/>
</owl:Class>
<Braille rdf:ID="Alva_BC640">
  <j.0:hasName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Alva BC640</j.0:hasName>
  <j.0:hasDescription
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >The ALVA BC640 combines powerful features with a compact and
lightweight design. High quality Optelec Braille cells and easy-to-
operate keys allow for effortless reading and smart navigation. The
optional Braille Audio Feature Pack makes the ALVA BC640 uniquely
versatile. The ergonomically designed Braille input keys and
integrated high quality audio speakers makes you operate your ALVA
BC640 efficiently and comfortably. Upon request the ALVA BC640 can
even be equipped with internal memory, allowing you to store your
documents or host your preferred screenreader on-
board.</j.0:hasDescription>
</Braille>
```

**G-REQ3-11:** *The ontology shall incorporate different types of applications (such as web services, HTML) that can be verified for their accessibility status.*

The ACCESSIBLE ontology includes Domain-Specific Ontologies that cover different application domains, and their reflection into accessibility assessment guidelines, as detailed before.

**G-REQ3-12:** *For the implementation of the ontologies, the Protégé ontology editor will be used as well as the SWRL supporting tool.*

As presented in Deliverable D4.1, A set of formalisms and taxonomies for accessibility assessment procedures and their inherent meta-models, the Protégé ontology editor was used to define all OWL ontological resources and associated languages. A screenshot of such edition process is presented below:



**Figure 3:** Edition process

**G-REQ3-13:** *Adequate semantic description shall be provided from all the characteristics that are included in the generic as well as in the domain ontologies.*

All OWL classes and properties in the Generic Ontology, as well as in all Domain-Specific Ontologies, reflected the naming schemes recurrently used in accessibility guidelines (e.g., Guideline, Checkpoint, Technique, etc.) As

detailed previously, the naming schemes are compliant with the expected terminology. Furthermore, all checkpoints and associated instances of accessibility guidelines provide a textual description of their goal, in order to be more readable by users that will interact directly or indirectly with the ontologies, as exemplified below:

```
<Guideline_has_Checkpoint>
  <j.0:Checkpoint rdf:ID="Checkpoint_I6.3">
    <hasId rdf:datatype="http://www.w3.org/2001/XMLSchema#int "
      >34</hasId>
    <Checkpoint_belongsTo_Guideline
rdf:resource="#Guideline_I06"/>
    <hasPriority
rdf:datatype="http://www.w3.org/2001/XMLSchema#int "
      >1</hasPriority>
    <hasParent rdf:resource="#Checkpoint_I6"/>
    <hasDescription
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >The inputs and outputs of operations should have the minimum
possible level of abstraction </hasDescription>
    <hasName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >WebService1 Checkpoint_I6.3</hasName>
  </j.0:Checkpoint>
</Guideline_has_Checkpoint>
```

***G-REQ3-14:*** *The execution of queries and rules shall be supported through the integration of an open source inference engine that shall support the usage of SPARQL as well as SWRL.*

The ACCESSIBLE Inference Engine provides support to the integration of the ACCESSIBLE Ontology into all technology evaluators. Furthermore, the Jena and Pellet open source Semantic Web technology libraries provide the ground basis for the mentioned integrations. By using Jena and Pellet, support is provided to the SPARQL and SWRL languages, as well as core Semantic Web languages, such as RDF, RDF-S, and OWL.

More details about this integration has been specified in deliverable D4.2 and further explained in subsequent sections of this deliverable.

## **2.2 Performance requirements**

***P-REQ3-1:*** *Response times in communication of the ACCESSIBLE knowledge resource with the assessment system shall be in line with those specified in the communication protocol selected.*

The ACCESSIBLE Inference Engine is defined as both an embeddable library and SOAP-based Web service, in order to allow different integration scenarios into technology evaluators. The selection of its usage is made according to the limitations imposed by required response times.

***P-REQ3-2:*** *In the same way, the inference engine in the complex rules and queries execution shall also take into account the performance.*

The average loading time of all ontologies devised (the Generic Ontology and all Domain-Specific Ontologies) is around 2.5 seconds (more details on performance can be found in section 6).

## 2.3 Operational requirements

*O-REQ3-1: The user shall be able to navigate through the ACCESSIBLE loaded ontology.*

The ACCESSIBLE ontology is navigable in different ways, including: Protégé, the ACCESSIBLE Rules User Interface, as well as technology-specific evaluators and the ACCESSIBLE Assessment Portal. Such examples are presented below:

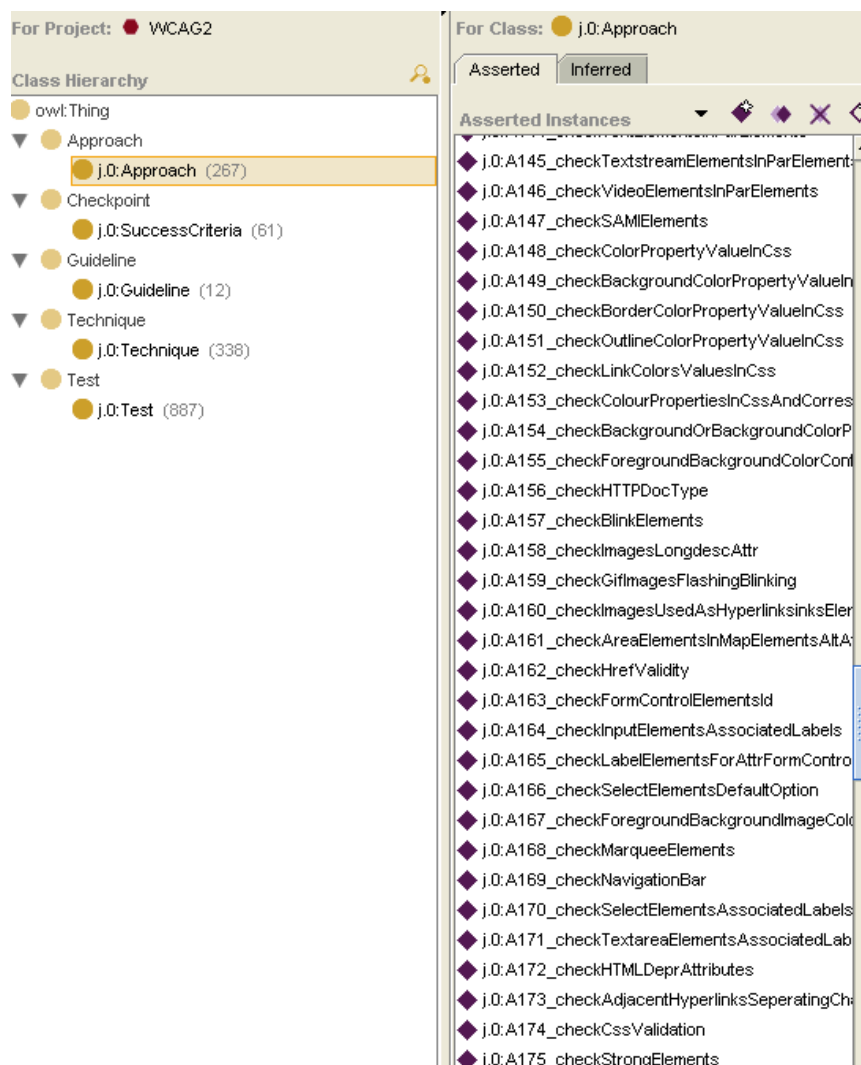
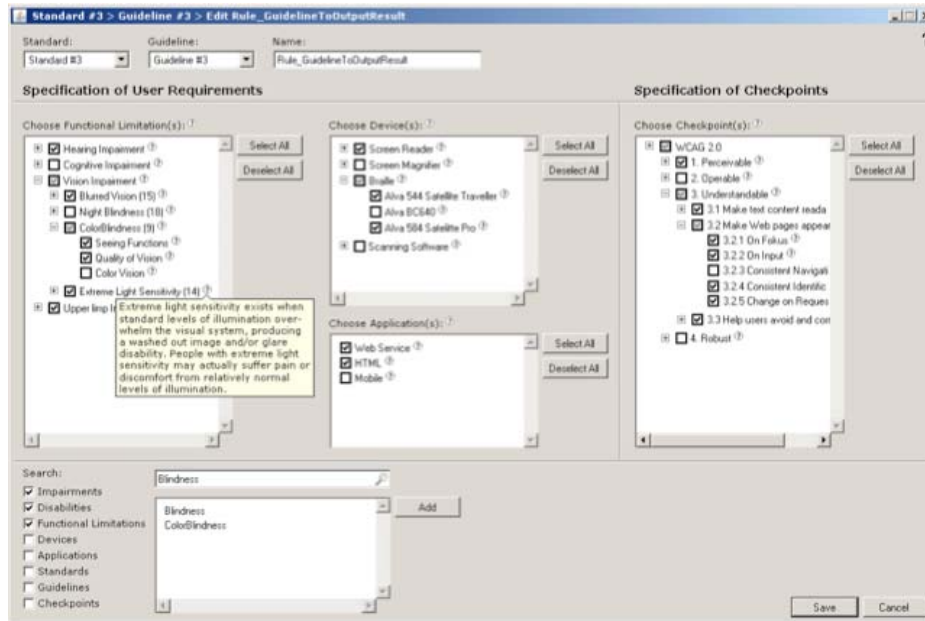


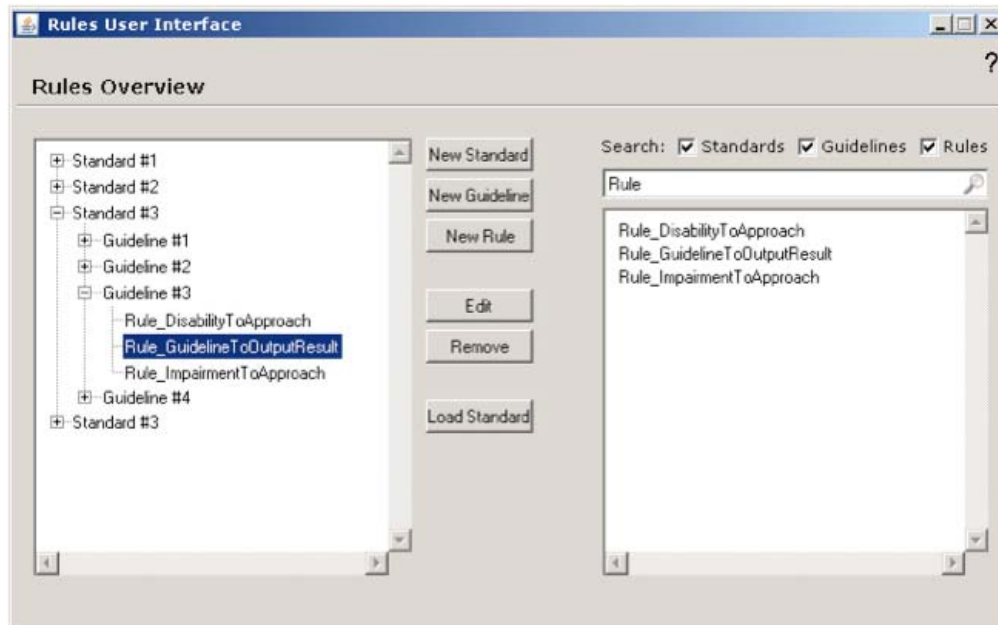
Figure 4: Example of navigation



**Figure 5:** Example of navigation

***O-REQ3-2:*** The user shall be able to search through the ontology in classes, properties and instances.

The ACCESSIBLE ontology includes standardised terminology and associated meta-information that affords searchability. Below we present a screenshot of the ACCESSIBLE Rules User Interface depicting this situation:



**Figure 6:** ACCESSIBLE Rules User Interface

***O-REQ3-3:** When displaying ontologies to the user via GUI, elements must be ordered according to the part of the ontology being displayed.*

The ACCESSIBLE ontology is presented in technology-specific evaluators and related GUIs, according to the associated accessibility assessment contexts.

***O-REQ3-4:** Developers or other users that have access to the ontology shall be familiar with the basic ideas of OWL language.*

The ACCESSIBLE ontology extensibility requires a minimum level of conceptualisations of OWL languages, which is leveraged through the ACCESSIBLE Rules User Interface.

***O-REQ3-5:** Developers or other users that wish to add new terms (such as a new accessibility guideline) to the ontology shall use the independent domain ontologies and comply with the generic ontology attributes.*

The ACCESSIBLE ontology provides normalised terminology that can be leveraged to implement technology-specific ontologies. The application of the Generic Ontology into the four Domain-Specific Ontologies implemented in the ACCESSIBLE ontology provides further support for this assertion.

## 2.4 Reliability requirements

**R-REQ3-1:** *The ontology-based resource shall provide answers just for the knowledge embedded in ontologies (both general and domain-specific). All knowledge not represented will be regarded as unknown, not as false (i.e. open-world assumption).*

The Semantic Web technologies used in the definition of the ACCESSIBLE ontology (e.g., RDF, RDF-S, OWL) provide support for this assertion.

## 2.5 Maintainability & Interoperability requirements

**M-REQ3-1:** *Developers shall be able to specify new SWRL rules and/or SPARQL queries through a special purpose GUI.*

The ACCESSIBLE Rules User Interface implements these features. Below we present a small example of the GUI and its application into the definition of a new rule.

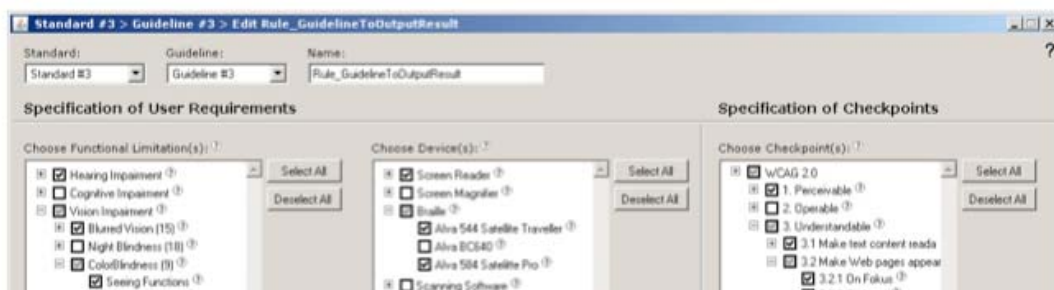


Figure 7: Example of Rules User Interface GUI

***M-REQ3-2:*** All ontologies should be developed in a way that knowledge can be interchanged between different Operating Systems and different knowledge management systems. For this, the ontological resource will be developed with standardised and open technologies and languages, including RDF, OWL, SWRL, and SPARQL.

All technologies used in the ACCESSIBLE ontology are runtime-agnostic (i.e., operating system, KMSs, etc.)

***M-REQ3-3:*** The knowledge inference engine must be interoperable and cope with open technologies. For this, it will be developed using open source knowledge software components developed in portable languages (e.g. Java).

All the libraries and software developed in the context of Work Package 4 are open-source and implemented using open-source and cross-platform languages and technologies. Java, Groovy, and JVM provide support for the implementation, and Jena/Pellet provide open-source inference mechanisms for Semantic Web technologies and languages.

***M-REQ3-4:*** In this same way, the ACCESSIBLE knowledge resource and the developed inference engine as well as the rules editor shall be enough documented in order to be understandable for users external to the development.

This deliverable, D4.3, as well as D4.2, provide documentation to support the extension and integration of the ACCESSIBLE Knowledge Resources into external software and accessibility evaluation processes.

## **3 OntologyQuery Server**

This Section presents the details of the OntologyQuery Server component. Its goal lays on providing a web service SOAP compliant server for querying a chosen OWL2-DL compliant ontology (in our case i.e. the ACCESSIBLE Generic Ontology or single .owl ontologies).

### ***3.1 Introduction and Context***

The ACCESSIBLE Ontological Knowledge – according to the D4.1 deliverable - “acts as a registry of knowledge for all the other components of the ACCESSIBLE Project. All ACCESSIBLE components access the knowledge base by interacting with the ontological knowledge resource through available interfaces.” Ontology in both ACCESSIBLE components designed and that are being designed is the essential core and interacts through external libraries/components or ACCESSIBLE built-in functionalities. All these ways of interacting have in common the SPARQL query language to access the accessible knowledge. A big amount of SPARQL queries are used and will be used to accomplish the goal expected then.

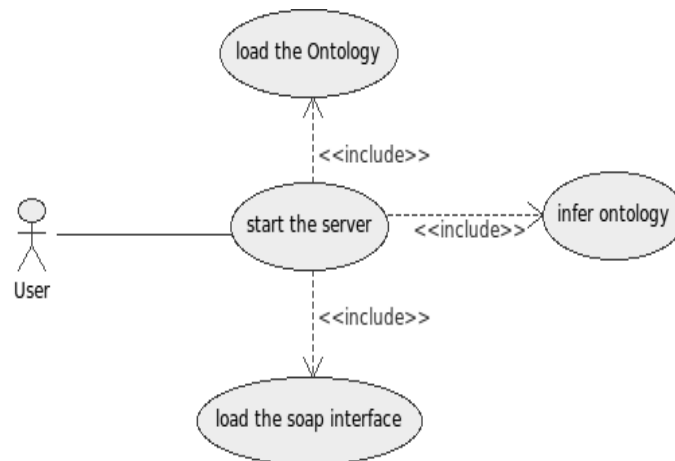
For the reasons explained above, a simple and straightforward tool for querying the ontology through SPARQL is a boost to the development of new queries. Every new SPARQL query can be checked and debugged, concentrating the attention on whether the given output is what we expect.

### ***3.2 Requirements***

Here are explained the main functional and non-functional requirements that have lead to the development of this component.

#### **3.2.1 Functional Requirements**

**Function: Start the server**



**Figure 8:** Starting the OntologyQuery Server

Starting conditions: This functionality occurs when the server component is loaded by another component (i.e. a general component that load both client and server part)

Normal flow of the functionality:

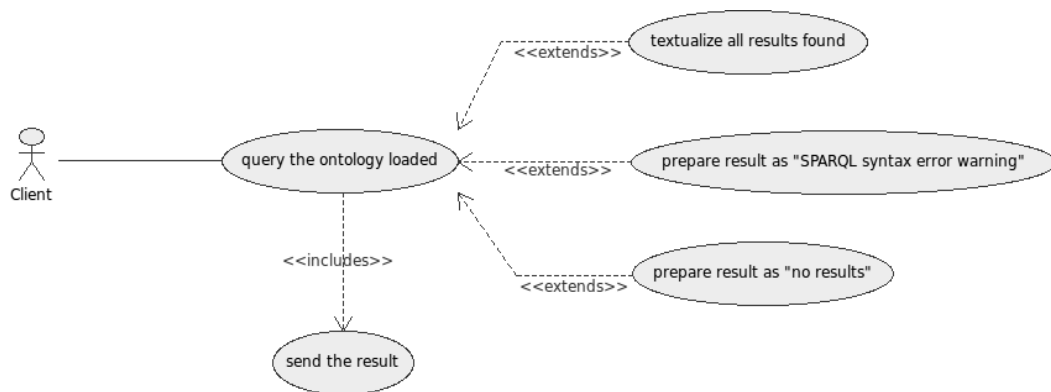
The server tries to load and infer the ontology indicated, using the alternative local file indicated into the locationmapper file. [E1]

Finally, it tries to set up the SOAP interface [E2]

Exceptional flow:

E1. In case some file needed is missing or it's impossible to continue loading the ontology, the component stops loading and generates an exception.

E2. In case some error occurs when it tries to set the soap interface up, the component stops working and generates an exception.

**Function: Query the ontology loaded****Figure 9:** Querying the Ontology

Starting conditions: This functionality occurs when the QueryOntology client or a generic SOAP client sends a query to the server address using an operation provided by the server.

Normal flow of the functionality:

A SOAP Message with the request is sent by the client

Query the SPARQL query on the ontology loaded when the server was opened. [E1][E2]

Parses all the results and prepares a textual row for each result [E3]

Send the result through SOAP protocol to the client.

Exceptional flow:

E1. In case the operation doesn't exist, a SOAP error message will be sent to the client

E2. In case the SPARQL query has some error, a SOAP warning message will be sent to the client as result.

E3. If the SPARQL query has zero result, a SOAP message telling there are zero result will be sent.

### 3.2.2 Non-functional requirements

Besides the main function requirements presented in the previous Section, we have defined a set of non-functional requirements that must be taken into account in the architecture and implementation decisions for the OntologyQuery Server, as follows:

NF1	This component must be as portable as possible
NF2	The component must be able to load every OWL2-DL ontology
NF3	The component must be able to make inference through the Pellet reasoner
NF4	The SPARQL queries must be done on the inferred ontology
NF5	It must be possible to specify as a parameter a path for a location-mapper file used to load alternative local files, in case the specified ontology whether imports or refers to other files from the Web that are impossible to fetch (i.e., no connections or services unavailable)
NF6	It's mandatory to specify as a parameter the complete URI that will be used from this component to create an access point to the services.

### 3.3 OntologyQuery Server Component

The component's architecture was devised according to the constraints and requirements presented in the previous Sections. This architecture is presented below:

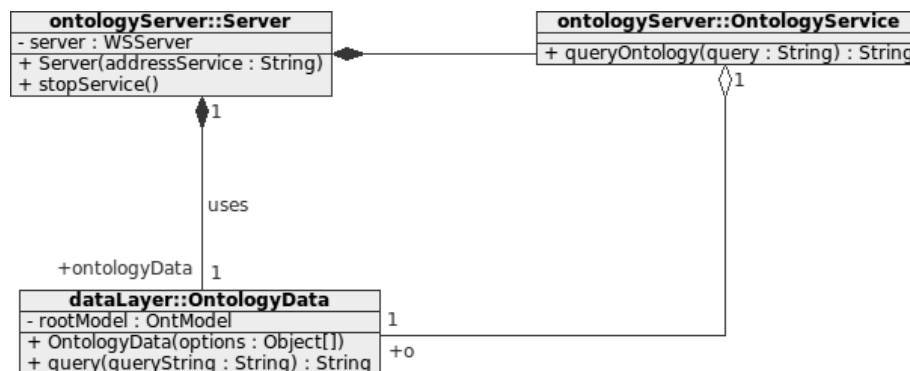
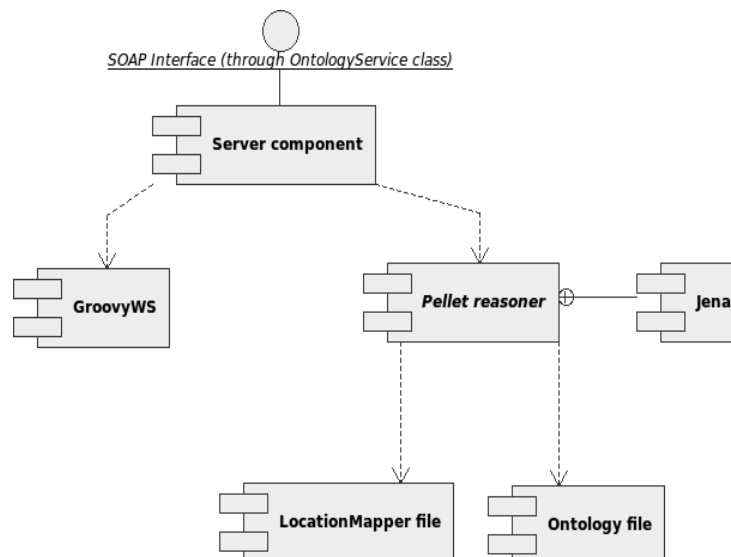


Figure 10: OntologyQuery Server Architecture

The components have been divided into two packages: a business layer named `ontologyServer` and a `dataLayer`. Further detailing these packages, this component comprises the following entities:

- **`dataLayer::ontologyData`**: it manages the actions related to the ACCESSIBLE Ontology. This class enables loading, inferring and querying the different aspects of the ACCESSIBLE Ontology.
- **`OntologyServer::OntologyService`**: this Bean-like entity is used for creating a SOAP service through GroovyWS library (c.f. the implementation Section below). Every public method declared creates a new service available.
- **`OntologyServer`**: thanks to the `WSServer` type property and the `OntologyService` type class, it offers a SOAP service for querying the ontology.

### 3.4 Implementation



**Figure 11:** OntologyQuery Server Implementation

The `Server component` is the core of the program. It has to load and manage the ontology, infer it, perform SPARQL queries and provide a SOAP

interface. To accomplish these tasks are being used these external components and libraries:

- **Groovy programming language:** Groovy is an open-source, agile programming language (akin to Ruby, Python, etc.) that sits on top of the JVM, providing a set of high abstractions and concepts – both at the language and libraries levels – to facilitate readability and flexibility of implementing software components;
- **JVM (Java Virtual Machine) technologies and libraries:** the foundation of the implementation of the Rules Inference Engine is the JVM and related libraries. We opted for this technology due to its pervasiveness (available in all relevant execution platforms) and openness (licensed as open-source). Furthermore, the software libraries that are made available right from the start by the JVM provide a lower entry barrier in the development of software components;
- **Pellet:** Pellet3 is an open-source, state-of-the-art OWL 2 reasoner, made available as a Java library;
- **GroovyWS:** All the Web services have been realized with GroovyWS4, an external library of the Groovy ecosystem, which provides a simple way to implement and use Web Services.

## 4 OntologyQuery Client

This Section presents the details of the OntologyQuery Client component. Its goal lays on providing a program that can query the ACCESSIBLE ontology through the services offered from the server component previously documented.

### 4.1 Introduction and Context

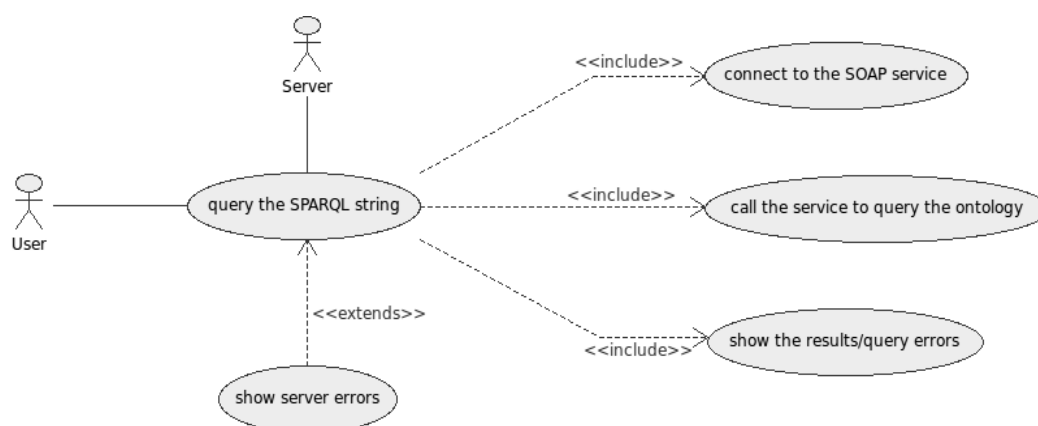
This component creates a simple-to-use and straightforward graphical interface to use the server component. It realizes the missing presentation layer in the previous component, by interacting with the web service realized. This division has been made to modularize and make possible future uses by third parts.

### 4.2 Requirements

Here are explained the main functional and non-functional requirements that have lead to the development of this component.

#### 4.2.1 Functional Requirements

##### Function: Query the SPARQL string



**Figure 12:** OntologyQuery Client Use Case

Starting conditions: This functionality occurs when the user click on “query”.

Normal flow of the functionality:

The client tries to create a connection with the server, by using the URI address specified at the loading of the program as a parameter [E1]

The client send the query request with the string written by the user using the graphical interface;

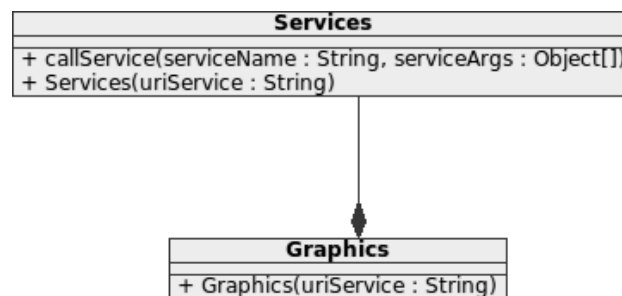
The result is showed in the text field;

Exceptional flow:

E1. In case the URI is unavailable or the server doesn't permit the operation required, a popup error message appears.

### 4.3 *OntologyQuery Client Component*

The component's architecture was devised according to the constraints and requirements presented in the previous Sections. This architecture is presented below:



**Figure 13:** OntologyQuery Client Architecture

- **Services:** it is the entity designed to connect the graphical interface to the Web service offered by the server component.
- **Graphics:** it is the entity designed to build the graphic interface.

### 4.4 *Implementation*

The Client component can be seen as the Presentation Layer of this tool. It has to interact graphically with the end users and the business layer performed by the server component, infers it, performs SPARQL queries and provides a SOAP interface. To accomplish these tasks are being used these external components and libraries:

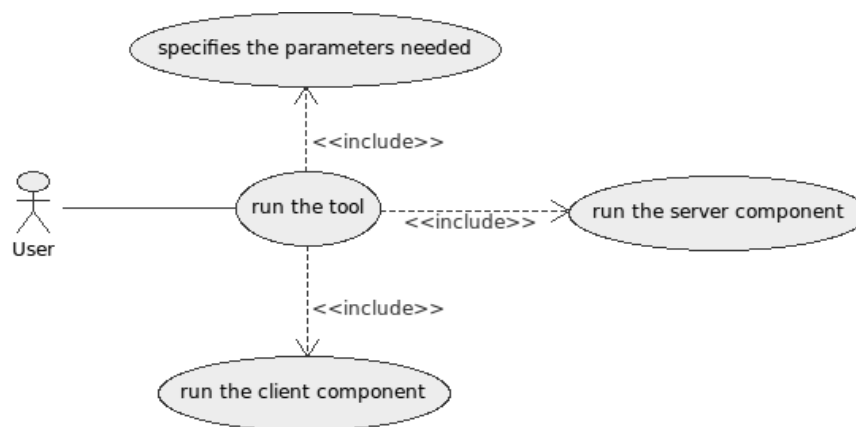
- **Groovy programming language:** Groovy is an open-source, agile programming language (akin to Ruby, Python, etc.) that sits on top of the JVM, providing a set of high abstractions and concepts – both at the language and libraries levels – to facilitate readability and flexibility of implementing software components;
- **JVM (Java Virtual Machine) technologies and libraries:** the foundation of the implementation of the Rules Inference Engine is the JVM and related libraries. We opted for this technology due to its pervasiveness (available in all relevant execution platforms) and openness (licensed as open-source). Furthermore, the software libraries that are made available right from the start by the JVM provide a lower entry barrier in the development of software components;

## 5 OntologyQuery General

This small component aims at loading the two main components as if they were bound together. This allows the quick and simple use of the client and server features, especially for testing and integration purposes.

### 5.1 Requirements

Here are explained the main functional and non-functional requirements that have lead to the development of this component.



**Figure 14:** OntologyQuery Client Implementation

Starting conditions: This functionality occurs when the user tries to open the tool.

Normal flow of the functionality:

1. The user opens the program through the shell script `execute.sh` that will open the tool with the default parameters. Otherwise the user can run the tool with personalized parameters by typing

```
java -jar "ontologyQuery.jar" <server-address> <path-ontology> <path-location-mapper-file>
```

where the `server-address` will be the complete URI that will be used to create an access point to the services (e.g., `http://localhost:6980/ontologyService`), the `path-ontology` will be the

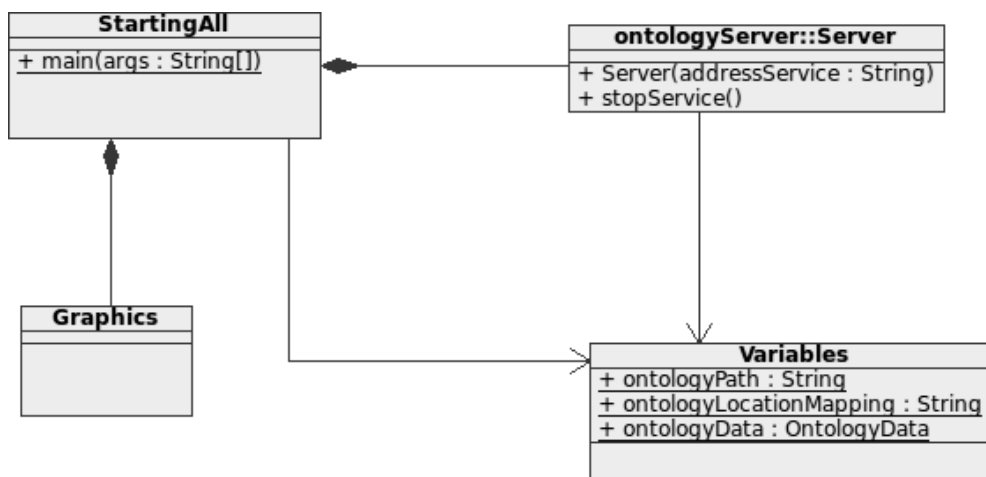
path used to load the ontology, and the path-location-mapper-file will be the path used to load alternative local files, in case the specified ontology <path-ontology> whether imports or refers to other files from the web that are impossible to fetch (no connections or services unavailable) [E1];

2. The program will execute the server and client component individually by passing the parameters needed.

#### Exceptional flow:

E1. In case some parameter is missing, a textual warning message will be displayed and the program will be terminated.

## 5.2 General Component



**Figure 15:** OntologyQuery General Implementation

- **Variables:** entity used for providing information to all the components of the tool.
- **StartingAll:** entity used to manage the execution of the two components and to set the necessary environment variables. This is the delegated class to run the tool thanks to its public static void main method.

## 6 Testing and validation

This part intends to give a guideline for evaluation and validation of methods and tools that have been implemented in WP4. It particularly aims to define measurement goals and for each of them, an associated set of tests and criteria. The purpose of this part is to provide a starting point for the evaluation of the ACCESSIBLE tools across the Pilot phase (WP6, WP7).

In fact, the next step, detailed in a future deliverable of the Work package 6 (D6.1b Pilot plans) will develop the evaluation procedure for the pilot applications and will determine the optimal evaluation methodology to properly carry out the system's performance test. Consequently, the Validation Plan will be established and will give specific details about the validation phase.

### 6.1 Measurement goals and Criteria

Considering the WP4 ACCESSIBLE objective(s) and goals, as described in the work plan of WP4, we can then define a set of measurement goals (focused on performance and usability issues), each one containing a set of criteria used for measurement.

During Pilots we aim to measure:

- WP4 Tool Performance
- Usability of WP4 Tools and acceptance by users

For each of these measurement goals, we provide a table containing a corresponding set of criteria. Considering [4], [5], [6] and [7], we also present a definition for every criterion.

#### 6.1.1 WP4 Tools' Performance

##### 6.1.1.0 System (software) Performance and Quality

Criteria	Definition
<b>Interoperability</b>	Examining interoperability issues between WP4 tools and other WP5 implementations
<b>Reliability</b>	The ability of a system or component to perform its required functions under stated conditions for a specified period of time

<b>Criteria</b>	<b>Definition</b>
<b>Testability</b>	The degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met
<b>Functional scope</b>	The range or scope to which a system component is capable of being applied
<b>Efficiency</b>	The degree to which a system or component performs its designated functions with minimum consumption of resources.
<b>Robustness</b>	The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environment conditions

### 6.1.1.1 Interface Quality

<b>Criteria</b>	<b>Definition</b>
<b>Homogeneity</b>	Stability of the design choices (presentation of the information, the same procedure leads to the same results)
<b>User-friendly</b>	The degree to which a system or component contains enough information to explain its objectives and properties
<b>Explicit control</b>	The means allowing the user to control the effects of orders (to identify the active options in the menu, to be able to stop an impression)
<b>Error management (handling)</b>	The function that identifies and responds to user errors to maintain normal or to correct its operations
<b>Conciseness</b>	The degree to which the interface has no excessive information present. Reduction of the activities of perception and memorisation (icons, default options)

### 6.1.2 Usability of the WP4 Tools

#### 6.1.2.0 Semantic aspects of querying

<b>Criteria</b>	<b>Definition</b>
<b>Simplicity</b>	The degree to which a system or component has a design and implementation that is straightforward and easy to understand
<b>Efficiency</b>	The degree to which a system or component performs its designated search functions with minimum consumption of resources

#### 6.1.2.1 Results of interaction

Criteria	Definition
<b>Significance</b>	The quality of the results according to the user request.
<b>Understandability</b>	The degree to which the presentation of the results is clear to the user
<b>Latency</b>	The length of time it takes to respond to an event

## 6.2 Performance evaluation of ACCESSIBLE ontologies

The ACCESSIBLE generic and domain ontologies were designed with the open source ontology editor Protege. In the initial implementation of the system we had not many real data available, so we could only test the basic structure of the ontologies. For that reason basic SWRL tests have been performed by running methods for creating Jena model in RDB and initialising knowledge from OWL files for detecting conflicts in the ontologies.

After the finalization of the ACCESSIBLE ontologies the usage of Logical and rule-based approaches to ontology validation and quality evaluation has been performed though the implemented inference engine.

As it was presented in previous section the average loading time of all ontologies devised (the Generic Ontology and all Domain-Specific Ontologies) is around 2.5 seconds (more details on performance can be found in section 6). We present below the timings (in nanoseconds) for loading each ontology (domain as well as the generic ontology), showing 10 loading attempts and corresponding average times:

GenericOntology.owl	
Loading Attempt	Time (nanoseconds)
1	3915200283
2	1468300961
3	427083843
4	355854921
5	326917680
6	309591132
7	309558353
8	307069046
9	306600343
10	320505899
Average	804668246

**Table 1:** Response time for loading GenericOntology.owl

**ScreenMagnifiers.owl**

Loading Attempt	Time (nanoseconds)
1	48579793
2	37021491
3	19861643
4	33945027
5	23667397
6	28766675
7	19085709
8	14782317
9	28655402
10	9874991
Average	26424045

**Table 2:** Response time for loading ScreenMagnifiers.owl**DescriptionLanguage.owl**

Loading Attempt	Time (nanoseconds)
1	799082597
2	357082555
3	220911797
4	310817936
5	144726907
6	113738619
7	197657327
8	83962668
9	69255347
10	73824047
Average	237105980

**Table 3:** Response time for loading DescriptionLanguage.owl**ScreenReader.owl**

Loading Attempt	Time (nanoseconds)
1	39400967
2	14095323
3	17253040
4	17961108
5	12426096
6	21067709
7	26604650
8	10318821
9	10236815
10	10214364
Average	17957889

**Table 4:** Response time for loading ScreenReader.owl**HTML.owl**

Loading Attempt	Time (nanoseconds)
1	113884381
2	153794780
3	58265469
4	120397320
5	49297494
6	84575086
7	57722596
8	35101296
9	28820612
10	46555585

Average	74841462
---------	----------

**Table 5:** Response time for loading HTML.owl**SpeechDevices.owl**

Loading Attempt	Time (nanoseconds)
1	36251357
2	30667282
3	30628313
4	29097952
5	23184601
6	21430202
7	10803512
8	11094028
9	10254899
10	17767296
Average	22117944

**Table 6:** Response time for loading SpeechDevices.owl**TextBrowsers.owl**

Loading Attempt	Time (nanoseconds)
1	95392025
2	31523274
3	11193822
4	23333898
5	17582455
6	29371559
7	23008788
8	13892517
9	15517752
10	8224217
Average	26904031

**Table 7:** Response time for loading TextBrowsers.owl**ListeningDevices.owl**

Loading Attempt	Time (nanoseconds)
1	33192042
2	17932715
3	23292745
4	18480933
5	9870311
6	9307585
7	18173742
8	8963847
9	8298268
10	11822250
Average	15933444

**Table 8:** Response time for loading ListeningDevices.owl**WCAG2.owl**

Loading Attempt	Time (nanoseconds)
1	1652583795
2	1372793924
3	411352289
4	404863320
5	413322760
6	336031265

7	292876576
8	287841391
9	291531739
10	284716903
Average	574791396

**Table 9:** Response time for loading WCAG2.owl**MWBP.owl**

Loading Attempt	Time (nanoseconds)
1	457519084
2	193839505
3	257726457
4	102464581
5	111618873
6	117309487
7	163601832
8	139156146
9	131697985
10	80564911
Average	175549886

**Table 10:** Response time for loading MWBP.owl**WebService1.owl**

Loading Attempt	Time (nanoseconds)
1	866943518
2	727619047
3	296845505
4	334775115
5	257375931
6	145288971
7	122994905
8	130304228
9	132303315
10	180124837
Average	319457537

**Table 11:** Response time for loading WebService1.owl**ScanningSoftware.owl**

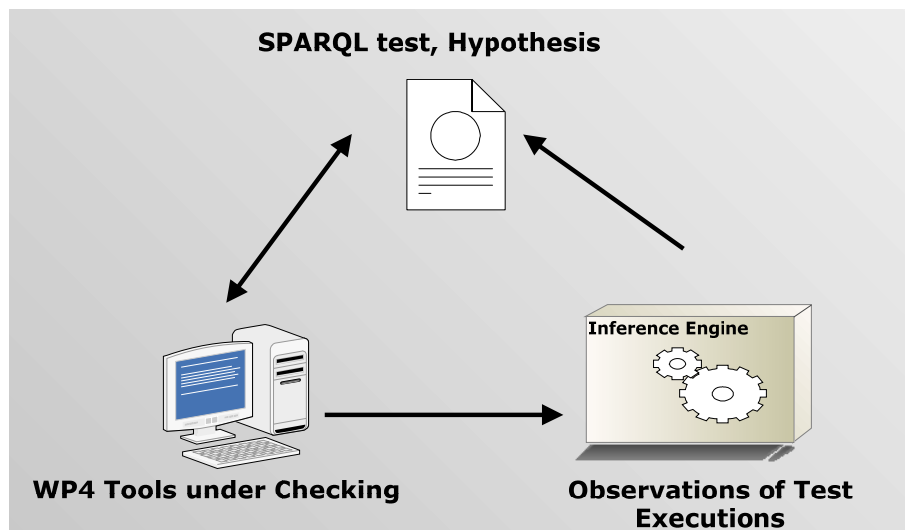
Loading Attempt	Time (nanoseconds)
1	32057065
2	11372929
3	15563411
4	10639942
5	18426601
6	11435835
7	8509125
8	8376951
9	15478975
10	8619982
Average	14048082

**Table 12:** Response time for loading ScanningSoftware.owl

### 6.3 Performance Evaluation of Inference Engine on Context Information (ACCESSIBLE ontologies)

Speed is a significant factor in the implementations of rule-based systems, and many inference engines slow dramatically as the size of the problem increases.

The implemented inference engine through the integration of Jena and Pellet open source Semantic Web technology libraries provide the ground basis for supporting the SPARQL and SWRL languages. For the performance evaluation on information, we placed our focus on scalability and subsequent performance issue which was the important issue for the successfully implementation. Specifically, in evaluating the performance of query processing, we considered 9 sets of SPARQL queries that were generated through the ACCESSIBLE ontologies.



**Figure 16:** Using the Inference engine to assess test set adequacy and performance

In order to evaluate handling of context information, SPARQL is used as follows (Sparql example for the selection of impairments):

```
SELECT *
WHERE {
    ?impairment rdf:type acc:Impairment ; acc:hasName ?name
}
```

The following sets of SPARQL queries were selected for the performance evaluation:

### **(1) Get applications query**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acc: <http://www.AccessibleOntology.com/GenericOntology.owl#>

SELECT *
WHERE {
    ?application rdf:type acc:Application ; acc:hasName ?name
}
}
```

### **(2) Get approaches and links query**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acc: <http://www.AccessibleOntology.com/GenericOntology.owl#>

SELECT *
WHERE
{
    ?approach rdf:type acc:Approach ; acc:hasDescription ?description .
    OPTIONAL { ?approach acc:Approach_linksTo_Standard ?standard . }
    OPTIONAL { ?approach acc:Approach_belongsTo_Checkpoint ?checkpoint . }
    OPTIONAL { ?approach acc:Approach_linksTo_Impairment ?impairment . }
    OPTIONAL { ?approach acc:Approach_linksTo_Disability ?disability . }
    OPTIONAL { ?approach acc:Approach_belongsTo_Test ?test . }
    OPTIONAL { ?approach acc:Approach_has_OutputResult ?outputResult }
}
ORDER BY ?approach
```

### **(3) Get Techniques query**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acc: <http://www.AccessibleOntology.com/GenericOntology.owl#>

SELECT *
WHERE {
    ?technique rdf:type acc:Technique ; acc:hasName ?name ;
    acc:Technique_belongsTo_Checkpoint ?checkpoint .
    OPTIONAL { ?technique acc:areAllTestsImplemented ?areAllTestsImplemented }
}
ORDER BY ?technique
```

### **(4) Get Standards query**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acc: <http://www.AccessibleOntology.com/GenericOntology.owl#>

SELECT *
WHERE {
    ?standard rdf:type acc:Standard ; acc:hasName ?name
}
}
```

**(5) Get Personas query**

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acc: <http://www.AccessibleOntology.com/GenericOntology.owl#>

SELECT *
WHERE {
    ?persona rdf:type acc:User ; acc:hasName ?name ; acc:hasAge ?age ;
    acc:hasMaritalStatus ?maritalStatus ; acc:hasJob ?job ; acc:hasLocation ?location ;
    acc:User_has_Disability ?disability ; acc:Meet ?meet ; acc:TechnologyUsage
    ?technologyUsage
}

```

**(6) Get Functional limitations**

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acc: <http://www.AccessibleOntology.com/GenericOntology.owl#>

SELECT *
WHERE
{
    ?functionalLimitation rdf:type acc:FunctionalLimitation; acc:hasName ?name .
    OPTIONAL { ?functionalLimitation acc:FunctionalLimitation_belongsTo_Disability
    ?disability }
}
ORDER BY ?functionalLimitation

```

**(7) Get Disabilities**

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acc: <http://www.AccessibleOntology.com/GenericOntology.owl#>

SELECT *
WHERE
{
    ?disability rdf:type acc:Disability ; acc:hasName ?name .
    OPTIONAL { ?disability acc:Disability_belongsTo_Impairment ?impairment }
}
ORDER BY ?disability

```

**(8) Get Devices**

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acc: <http://www.AccessibleOntology.com/GenericOntology.owl#>

SELECT *
WHERE {
    ?device rdf:type acc:Device ; acc:hasName ?name
}

```

**(9) Get Checkpoints**

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX acc: <http://www.AccessibleOntology.com/GenericOntology.owl#>

```

```
SELECT *  
WHERE {  
    ?checkpoint rdf:type acc:Checkpoint ; acc:hasName ?name ; acc:hasPriority ?priority  
}
```

We select query response time as the performance measure. The summary of response time is listed in the following Table 13.

Query #	Inference Engine Response Time (ms)
1	10,20
2	3043,21
3	2502,20
4	2225,34
5	250,00
6	520,30
7	35,70
8	221,30
9	980,50

**Table 13:** Summary of query response time



## 7 Conclusions

In this deliverable we discussed the compliance with the requirements for all components and knowledge defined at Work Package 4. Each requirement was faced with statements regarding each one of the components and knowledge infrastructure, from the ACCESSIBLE Inference Engine, the ACCESSIBLE Rules User Interface, as well as the ACCESSIBLE Ontology.

All of these requirements were successfully met which results on the correctness of the concepts present within the ACCESSIBLE Ontology, and their integration into the ACCESSIBLE Inference Engine. The technologies and languages used for the definition of these components and knowledge were based on existing – and thoroughly disseminated – standards, which affords also the main properties required for these components: their application and integration into different accessibility assessment and evaluation tools (such as those created within the scope of the ACCESSIBLE project), and their extensibility into new application domains, new accessibility standards, etc.

This deliverable also presented the advancements on the ACCESSIBLE Inference Engine, regarding their optimization and integration values towards its inclusion into the ACCESSIBLE Architecture and corresponding assessment and simulation tools. The new version of the ACCESSIBLE Inference Engine was presented, along the side of configurations and example client features, in order to afford a seamless integration of the Inference Engine services into different assessment tools.

The execution of the Inference Engine, whether if integrated as a Web service or a library within accessibility assessment tools, provides the required guidance for the expedite evaluation of accessibility in different application domains, and, consequently, provide evidence of the compliance with the elicited requirements for the ACCESSIBLE Content and Knowledge Infrastructure – the overarching goal of Work Package 4.

## 8 References

- [1] D4.1: “A set of formalisms and taxonomies for accessibility assessment procedures and their inherent meta-models”
- [2] D4.2: “A software package containing a set of modelling tools, rules inference engine, and the rules graphical editor”
- [3] D2.2: “User needs and System Requirements Specification”
- [4] [Evans 87] Evans, Michael W. & Marciniak, John. *Software Quality Assurance and Management*. New York, NY: John Wiley & Sons, Inc., 1987.
- [5] [IEEE 90] Institute of Electrical and Electronics Engineers. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, NY: 1990
- [6] [Nyberg et.al.94] Eric H. Nyberg, Teruko Mitamura & Jaime G. Carbonell. *Evaluation Metrics for Knowledge Based Machine Translation*, Proceedings of COLING-94, Kyoto, Japan, August 5-9, 1994
- [7] [Lompré 01] Nicole Lompré. *Démarche pour l'évaluation ergonomique*. 2001