D5.2: Synergetic Semantic Annotation Environment

Deliverable Form

<table>
<thead>
<tr>
<th>Project Reference No.</th>
<th>ICT FP7318497</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable No.</td>
<td>D5.2b</td>
</tr>
<tr>
<td>Relevant Workpackage:</td>
<td>WP5: Semantic Infrastructure</td>
</tr>
<tr>
<td>Nature:</td>
<td>R, P</td>
</tr>
<tr>
<td>Dissemination Level:</td>
<td>PU</td>
</tr>
<tr>
<td>Document version:</td>
<td>V1.3</td>
</tr>
<tr>
<td>Date:</td>
<td>21/11/2014</td>
</tr>
<tr>
<td>Authors:</td>
<td>NCSR-D</td>
</tr>
<tr>
<td>Document description:</td>
<td>The SemaGrow Synergetic Semantic Annotation Environment is the visual tool that will assist data providers in adding and maintaining their data source as a member of a SemaGrow federation. This report documents the prototype and provides a user guide.</td>
</tr>
</tbody>
</table>
Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author (Partner)</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draft v0.1</td>
<td>7/4/2014</td>
<td>NCSR-D</td>
<td>Document structure</td>
</tr>
<tr>
<td>Draft v0.8</td>
<td>24/4/2014</td>
<td>NCSR-D</td>
<td>ELEON 3 Developers Guide and Users Guide</td>
</tr>
<tr>
<td>Draft v0.9</td>
<td>9/5/2014</td>
<td>UAH</td>
<td>Internal review</td>
</tr>
<tr>
<td>Final v1.0</td>
<td>16/6/2014</td>
<td>NCSR-D</td>
<td>Delivered as D5.2</td>
</tr>
<tr>
<td>Draft v1.1</td>
<td>29/9/2014</td>
<td>NCSR-D</td>
<td>Updated with text regarding visualizing and editing self-tuning histograms</td>
</tr>
<tr>
<td>Draft v1.2</td>
<td>6/10/2014</td>
<td>UAH</td>
<td>Internal review</td>
</tr>
<tr>
<td>Final v1.3</td>
<td>21/11/2014</td>
<td>NCSR-D</td>
<td>Delivered as D5.2b</td>
</tr>
</tbody>
</table>
EXECUTIVE SUMMARY

The SemaGrow Synergetic Semantic Annotation Environment, the visual tool that allows data providers to describe the content and technical specifications of the data sources they contribute to the SemaGrow federation. The SemaGrow Synergetic Semantic Annotation Environment is developed as an extension of the ELEON Ontology Enrichment Environment: a metadata authoring environment designed allowing non-technical domain experts to annotate ontologies with application-specific meta-information.

The SemaGrow Synergetic Semantic Annotation Environment, as well as ELEON itself, is open source software developed on a public BitBucket repository. This report documents the version tagged as ELEON v3.0 on the repository.

We foresee two modes of operation of the SemaGrow Synergetic Semantic Annotation Environment: (a) when new data sources are added to the federation, or the schema employed in a data source is updated, or several instance-level changes (dataset size, entity URI patterns) have accumulated, the data provider can use ELEON to provide or update the manually authored description of the data source; In addition, (b) metadata that is automatically maintained by the Resource Discovery component of the SemaGrow Stack can be inspected and corrected.

In general, the SemaGrow Synergetic Semantic Annotation Environment is the visual tool that will assist data providers with implementing the guidelines for adding and maintaining their data source as a member of a SemaGrow federation.
TABLE OF CONTENTS

LIST OF FIGURES .......................................................................................................................... 5
LIST OF TABLES .............................................................................................................................. 6
LIST OF TERMS AND ABBREVIATIONS .......................................................................................... 7

1. INTRODUCTION .......................................................................................................................... 8
   1.1 Purpose and Scope ...................................................................................................................... 8
   1.2 Approach to Work Package and Relation to other Deliverables .............................................. 8
   1.3 Methodology and Structure of the Deliverable ......................................................................... 10
   1.4 Big Data Aspects .................................................................................................................... 10

2. DEVELOPERS GUIDE .................................................................................................................. 11
   2.1 Overview .................................................................................................................................. 11
   2.2 Presentation Layer .................................................................................................................... 11
   2.3 Persistence Layer and Importing/Exporting ............................................................................ 11
   2.4 Codebase Structure .................................................................................................................. 12

3. USER GUIDE ................................................................................................................................ 14
   3.1 Core concepts ......................................................................................................................... 14
   3.2 Installation and execution ........................................................................................................ 15
   3.3 Starting an annotation session ................................................................................................. 16
   3.4 Faceted browsing ..................................................................................................................... 19
       3.4.1 Per Property annotation ..................................................................................................... 19
       3.4.2 Per Entity annotation .......................................................................................................... 23
       3.4.3 Histogram annotation ....................................................................................................... 25
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Position of the ELEON Annotation Environment in the overall SemaGrow Ecosystem</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>Dependencies with other tasks and WPs</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Schematic overview of how classes and interfaces extend, implement, or point to each other</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>Loading and saving annotations</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Login menu</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Annotation Schema</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>Data Schema menu</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>Dataset browsing facets</td>
<td>18</td>
</tr>
<tr>
<td>9</td>
<td>Insert dataset label</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>Dataset label</td>
<td>19</td>
</tr>
<tr>
<td>11</td>
<td>New dataset inserted</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>Property editor</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>Schema selection</td>
<td>21</td>
</tr>
<tr>
<td>14</td>
<td>Schema added</td>
<td>21</td>
</tr>
<tr>
<td>15</td>
<td>Remove from tree</td>
<td>22</td>
</tr>
<tr>
<td>16</td>
<td>Insert dataset</td>
<td>22</td>
</tr>
<tr>
<td>17</td>
<td>Insert new subset</td>
<td>23</td>
</tr>
<tr>
<td>18</td>
<td>Subject and Object pattern definition</td>
<td>23</td>
</tr>
<tr>
<td>19</td>
<td>Subset added</td>
<td>24</td>
</tr>
<tr>
<td>20</td>
<td>Add existing dataset or subset as child</td>
<td>24</td>
</tr>
<tr>
<td>21</td>
<td>Multiple inheritance</td>
<td>25</td>
</tr>
<tr>
<td>22</td>
<td>Histogram Tree Structure</td>
<td>26</td>
</tr>
<tr>
<td>23</td>
<td>Histogram Properties Editor</td>
<td>26</td>
</tr>
<tr>
<td>24</td>
<td>Editing URI regular expressions</td>
<td>27</td>
</tr>
<tr>
<td>25</td>
<td>Editing numerical ranges</td>
<td>27</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 1: ELEON dependencies ................................................................. 15
## LIST OF TERMS AND ABBREVIATIONS

<table>
<thead>
<tr>
<th>Term/Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotator</td>
<td>A user role in ELEON: the user who uses ELEON to edit metadata.</td>
</tr>
<tr>
<td>Data curator</td>
<td>A user role in the SemaGrow Synergetic Semantic Annotation Environment: the specialization of the ELEON annotator role to a user who edits metadata about data sources. The data curator might be the same person or organization as the data provider, but can also be a data consumer who annotates publicly available third-party data sources in order to add them to a SemaGrow federation.</td>
</tr>
<tr>
<td>ELEON</td>
<td><strong>ELEON Ontology Enrichment Environment</strong>: a metadata authoring environment developed by NCSR-D. ELEON focuses on allowing non-technical domain experts to annotate ontologies with application-specific meta-information. ELEON was assumed as the foundation for the development of the SemaGrow Synergetic Semantic Annotation Environment (this deliverable).</td>
</tr>
<tr>
<td>Histogram</td>
<td>A structure capable of holding information about data, without continuously accessing them, but only once for initial refinement. It can be interpreted as a tree structure, with the root being the more general node, containing information about the whole dataset. Each child of the root focuses on a smaller area of the data and their children so on. In our case the histograms contain metadata about a dataset, and their Sevod serializations can loaded in ELEON for visualization and editing.</td>
</tr>
<tr>
<td>Jena</td>
<td>Apache Jena is an open source Java framework for building Semantic Web and Linked Data applications (<a href="http://jena.apache.org">http://jena.apache.org</a>)</td>
</tr>
<tr>
<td>Resource Discovery</td>
<td>The component of the SemaGrow Stack that maintains and serves metadata about the contents of each data source that is federated under the SemaGrow Stack.</td>
</tr>
<tr>
<td>SemaGrow Stack</td>
<td>The SemaGrow Stack integrates the SemaGrow components needed in order to offer a single SPARQL endpoint that federates a number of large-scale, heterogeneous data sources, also exposed as SPARQL endpoints.</td>
</tr>
<tr>
<td>Sevod</td>
<td><strong>Sevod</strong> is the extension of the VoID vocabulary developed in the SemaGrow project. Sevod extends VoID with properties addressing how different datasets can be joined in responses to queries that combine information from multiple data sources.</td>
</tr>
<tr>
<td>SWT</td>
<td>The Standard Widget Toolkit (SWT) is a graphical widget toolkit for use with the Java platform. It is now maintained by the Eclipse Foundation. SWT is used for the visual elements of ELEON.</td>
</tr>
<tr>
<td>VoID</td>
<td>The Vocabulary of Interlinked Datasets (VoID) specifies how to provide information about the size and semantics of a dataset, defining properties such as the number of triples in a dataset and its subsets, the number of distinct entities mentioned in these triples, the classes that characterize these entities, etc.</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

1.1 Purpose and Scope

This report documents the SemaGrow Synergetic Semantic Annotation Environment, the visual tool that allows data providers to describe the content and technical specifications of the data sources they contribute to the SemaGrow federation. The SemaGrow Synergetic Semantic Annotation Environment is developed as an extension of the ELEON Ontology Enrichment Environment: a metadata authoring environment designed allowing non-technical domain experts to annotate ontologies with application-specific meta-information. ELEON has been previously used to annotate semantic data with the linguistic knowledge needed in order to realize multi-lingual descriptions of ontology entities. Re-using the core ELEON mechanisms, the SemaGrow Synergetic Semantic Annotation Environment views datasets and their inclusion relationships as the data that is to be annotated with knowledge concerning:

- The kinds of entities in the datasets, their classes, naming conventions they follow (as regular expressions over their URIs), and the properties that link them
- The number of triples in each dataset; and
- Relations across datasets, such as the join selectivity

We foresee two modes of operation: (a) when new data sources are added to the federation, or the schema employed in a data source is updated, or several instance-level changes (dataset size, entity URI patterns) have accumulated, the data provider can use ELEON to provide or update the manually authored description of the data source; In addition, (b) metadata that is automatically maintained by the Resource Discovery component of the SemaGrow Stack can be inspected and corrected.

In general, the SemaGrow Synergetic Semantic Annotation Environment is the visual tool that will assist data providers with implementing the guidelines for adding and maintaining their data source as a member of a SemaGrow federation. ELEON interacts with the SemaGrow Stack components via Resource Discovery, through which it both populates the data source metadata repository with manually provided metadata and reads the automatically extracted metadata in order to present them (Figure 1).

1.2 Approach to Work Package and Relation to other Deliverables

The aim of WP5 is to carry out all development, integration, and deployment required in order to produce a complete and robust SemaGrow system based on the methodological and research prototyping work in WP3 and WP4; including both the SemaGrow Stack and the associated off-stack tools.

With respect to this deliverable in particular, this approach is refined as follows (Figure 2):

- Task 5.2 receives as input the architecture developed in Task 2.3, in order to implement the appropriate interfaces for interacting with the Resource Discovery component.
- Task 5.2 receives as input the data source annotation vocabulary developed in Task 3.1, in order to implement the appropriate functionalities for visualizing and editing data source descriptions. Task 3.1 has already taken into account the metadata requirements set by the query optimization method (Task 3.4) and has translated such requirements into the Sevod vocabulary, and extension of the VoID vocabulary for describing Semantic Web datasets (cf. Section 3.1).
- Task 5.2 provides the SemaGrow Synergetic Semantic Annotation Environment to the integrated prototype prepared in Task 5.4.
Figure 1: Position of the ELEON Annotation Environment in the overall SemaGrow Ecosystem

Figure 2: Dependencies with other tasks and WPs
1.3 Methodology and Structure of the Deliverable

Besides this introductory section, this document includes a *Developers’ Guide* (Section 2) that explains the structure of the codebase of the *SemaGrow Synergetic Semantic Annotation Environment*; including interfaces and classes from the underlying ELEON Environment used in the SemaGrow extension and *not* including ELEON interfaces and classes that are not relevant to SemaGrow. It then proceeds to include the *Users’ Guide* (Section 3) that is distributed to participants of SemaGrow pilots and hackathons. And then concludes with an outline of the steps to take until final delivery (M24).

Please note that the original schedule on delivering D5.2 on M18 has been updated as a response to comments by the Expert Steering Group and discussion during the first review indicating the importance of automatically extracted metadata. In the updated schedule, the original scope of delivery (manual annotations) is delivered as part of the M18 delivery batch and the new functionality (presenting and editing automatically extracted metadata) will be delivered in tandem with development in Task 3.1 of the metadata extraction method.

1.4 Big Data Aspects

There are no big data issues to be addressed in the context of this deliverable. The manually authored metadata is, naturally, not going to present a scalability issue. The automatic extraction of data source annotations could pose a big data problem to Task 3.1 for queries that bring back large results and the extracted metadata is exploited in Task 3.4 to address big data challenges, but the metadata itself is not big data.
2. DEVELOPERS GUIDE

2.1 Overview

ELEON is published as open source software under the GPLv2 license. ELEON was originally developed on CVS and then on an internal git installation at NCSR-D. Work has now been moved to a public Bitbucket repository, with some of the pre-Bitbucket history retained. Development takes place on the master branch. The latest release is ELEON v3.0; bug fixes and minor releases are maintained on the eleon3 branch. To retrieve a copy of the repository:

```
# git clone https://bitbucket.org/semagrow/eleon.git
```

Development takes place in Eclipse and the Eclipse .project and .classpath files are included in the repository. ELEON dependencies are listed in Table 1, Users Guide (Section 4 of this document). The versions shown on that table are the ones used during development and testing of ELEON v3.0; other versions might also work but have not been tested.

2.2 Presentation Layer

The ELEON presentation layer has been developed using the WindowBuilder plug-in for the Eclipse IDE and the Standard Widget Toolkit (SWT). Specifically, the panels that make up the presentation layer have been developed using WindowBuilder SWT Designer, a visual designer and layout tools to create from simple forms to complex windows. WindowBuilder uses GEF to display and manage the visual presentation.

Generated code does not require any additional custom libraries to compile and run: all of the generated code can be used without having WindowBuilder installed. WindowBuilder can read and write almost any format and reverse-engineer most hand-written Java GUI code. It also supports free-form code editing and most user refactorings.

The Standard Widget Toolkit (SWT) is a graphical widget toolkit for use with the Java platform. It was originally developed by Stephen Northover at IBM and is now maintained by the Eclipse Foundation in tandem with the Eclipse IDE.

To display GUI elements, the SWT implementation accesses the native GUI libraries of the operating system using Java Native Interface (JNI) in a manner that is similar to those programs written using operating system-specific APIs. Programs that use SWT are portable, but the implementation of the toolkit, despite being partly written in Java, is unique for each platform.

SWT is a wrapper around native code objects, such as GTK+ objects, Motif objects, etc. In cases where native platform GUI libraries do not support the functionality required for SWT, SWT implements its own GUI code in Java, similar to Swing. In essence, SWT is a compromise between the low level performance and look and feel of AWT and the high level ease of use of Swing.

SWT widgets have the same "look and feel" as native widgets because they often are the same native widgets. This is in contrast to the Swing toolkit where all widgets are emulations of native widgets. In some cases the difference is distinguishable. For example the OS X tree widget features a subtle animation when a tree is expanded and default buttons actually have an animated pulsing glow to focus the user's attention on them. The default Swing versions of these widgets do not animate.

Since SWT is simply a wrapper around native GUI code, it does not require large numbers of updates when that native code is changed, provided that there is no change in the API between operating system updates. The same cannot be said of Swing: Swing supports the ability to change the look and feel of the running application with "pluggable look and feels" which enable emulating the native platform user interface using themes, which must be updated to mirror operating system GUI changes (such as theme or other look and feel updates).

2.3 Persistence Layer and Importing/Exporting

ELEON uses the Apache Jena Framework to handle ontology creation and annotation. Jena is an open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs. The graphs are represented as an abstract model. A model can be sourced with data from files, databases, URLs or a combination of these. A model can also be queried through SPARQL and updated through SPARQL. Jena supports serialisation of RDF graphs to a relational database, RDF/XML, Turtle, and Notation 3.
2.4 Codebase Structure

The following interfaces and classes have been developed for the SemaGrow application of annotating Semantic Web datasets. Figure 3 gives a schematic overview of how classes and interfaces extend, implement, or point to each other.

package gr.demokritos.iit.eleon.facets

package-info: This package contains interfaces and abstract classes relevant to how data will be organized and visualized in order to be presented to the user.

interface Facet: The Facet is one of the core concepts of ELEON; each Facet is a different way to visualize and present the data that will be annotated. This interface is the most abstract generalization of anything that can present to the user entities in a Semantic Web representation so that the user can select one to annotate.

interface TreeFacet: This interface is an abstraction over all facets that are based on the SWT widget for drawing trees, org.eclipse.swt.widgets.Tree.

interface TreeFacetNode: This interface is an abstraction over the classes of objects that are attached to each node of the tree in a TreeFacet. Each org.eclipse.swt.widgets.TreeItem in the org.eclipse.swt.widgets.Tree of this facet holds a pointer to a TreeFacetNode instance, accessible through the TreeItem::setData() and TreeItem::getData() methods.

gr.demokritos.iit.eleon.facets.dataset

package-info: This package contains an implementation of the TreeFacet interface suitable for annotating datasets with the metadata needed for query optimization.

abstract class DatasetFacet: This is the base class for all TreeFacet implementation in this package.

abstract class DatasetNode: This is the base class for all TreeFacetNode implementation in this package.

class PropertyTreeFacet: An instance of this class presents a dataset as the inclusion hierarchy of the subsets that hold all triples with a given predicate. The inclusion tree is automatically built from rdfs:subPropertyOf statements.

class PropertyTreeNode: An instance of this class represents a subset where all triples use the same predicate.

class TriplePatternFacet: An instance of this class presents a dataset as an inclusion hierarchy of the subsets that hold all triples that match a given triple pattern. The inclusion hierarchy is not inferred from these patterns, but is explicitly stated by the user.

class TriplePatternNode: An instance of this class represents a subset where all triples match a pattern. Triple patterns are (subject predicate object) triplets where: (a) the subject and object can be left as a variable or specified by a regular expression that matches their URIs and/or the RDFS class they are instances of; and (b) the predicate can be left as a variable or specified by a regular expression that matches its URI.

class HistogramPatternNode: An instance of this class presents a dataset that is automatically generated by a histogram. The subsets of this dataset hold triples that match an arbitrary pattern, given by a user of the SemaGrow endpoint.

class HistogramTreeNode: An instance of this class represents a subset where all triples match a pattern. The Eleon user does not specify the pattern, for it's been automatically generated by the histogram.

gr.demokritos.iit.eleon.annotations

package-info: This package contains classes that represent the annotation schemas and the users’ access rights.

class AnnotationVocabulary: Each instance of this class knows how to access the annotation properties that are necessary in order to present the data (e.g., which is the property that assign a human-readable label, which is the property that assigns a hierarchical structure, etc.) as well as the properties that should be made available to the user for annotating the data.

class Annotator: Each granule of annotations (typically the annotation properties of an instance that is presented as a node in the facet) is assigned an Annotator instance. All users can view all annotations, but they can only edit their own annotations.
**class AnnotatorList:** The Annotator instances known to the system, and information about their authentication.

![Diagram](image_url)

**Figure 3:** Schematic overview of how classes and interfaces extend, implement, or point to each other.

---

**gr.demokritos.iit.eleon.persistence**

**package-info:** This package contains classes that implement functionality related to persisting and importing/exporting serializations of annotations created in ELEON.

**interface PersistenceBackend:** An abstraction over all persistence back-ends.

**class ELEONXML:** This class implements a back-end based on a native ELEON XML schema.

**class OWLFile:** This class implements a back-end based on Jena.

**gr.demokritos.iit.eleon.ui**

**package-info:** This package contains the classes implementing dialog boxes and similar SWT elements.
3. USER GUIDE

3.1 Core concepts

ELEON is a tool for the creation and editing of dataset descriptions. These descriptions comprise metadata regarding the structure, semantics, and volume of RDF data repositories. Such metadata is used by distributed querying engines in order to optimize query execution.

In ELEON, the structure of a data repositories (and of collections comprising multiple data repositories) is perceived as a hierarchy of datasets and their subsets: a dataset contains all of the RDF triples contained in all of its subsets; these subsets are a dataset in their own right, and can be broken down into smaller subsets that have semantics and volume properties of their own. Each dataset is represented as a node in a tree-like hierarchy. Each node is associated with values for the annotation properties that are foreseen by the annotation schema in use. The distinction should be noted between:

- The schemas used in the datasets themselves. Here, we will call these data schemas; and
- The schemas used in order to describe datasets. Here, we will call these annotation schemas.

With respect to annotation schemas, ELEON supports the Vocabulary of Interlinked Datasets (VoID) and Sevod, the extension of VoID developed in the SemaGrow project. A fuller presentation of these properties can be found below, so we will at this point only say that VoID properties provide information about size and semantics, such as the number of triples in a dataset, the number of distinct entities mentioned in these triples, the classes that characterize these entities, etc. Sevod extends VoID with more detailed information mostly addressing how different datasets can be joined in responses to queries that combine information from multiple data sources.

In VoID and Sevod, as well as in their visual presentation in ELEON, there is no class distinction between the different levels of datasets in the hierarchy: a dataset can be anything from a huge collection to a handful of triples that is a tiny part of the data served by a data source. Conceptually, however, we will make the following distinctions in this document:

- We will call data source a dataset that is served by a SPARQL endpoint. All data in subsets of a dataset are assumed to be accessible via the data source’s endpoint. The importance of data sources is that they are the entities that can be accessed by the query engine; all information about their subsets is only relevant for deciding whether a data source’s endpoint should be contacted for a given query or not.
- We will call collection a dataset that comprises several data sources that are known by collective name and that are provided by a single creator entity, follow the same data schemas, or have some other unifying attributes. The importance of collections is that they offer a human-readable title and other provenance information that can be used to prefer or exclude collections from the data considered for a given query.
- We will call subset or dataset a dataset that is part of a data source.
- We will call a dataset complete if it either (a) has no subsets or (b) among its direct subsets, there is a set of subsets that form a partitioning of the dataset. The importance of complete datasets is that besides knowing what can be found at a data source, we can also infer that some resources cannot be possibly found at a data source.

Although this three-level organization is not foreseen by either VoID or Sevod, and neither enforced by ELEON, it should be considered a best practice recommendation for maximizing the usefulness of the descriptions for the purpose of query engine optimization.

---

1 Multiple inheritance is supported, but it is very rarely useful when annotating datasets.
2 cf. K. Alexander, R. Cyganiak et al., Describing Linked Datasets with the VoID Vocabulary. W3C Interest Group Note, 3 March 2011. Available at http://www.w3.org/TR/void
3.2 Installation and execution

ELEON is published as open source software under the GPLv2 license and is developed on a public Bitbucket repository. This document describes ELEON v3.0 which is maintained on branch “eleon3” of the semagrow/eleon repository. To checkout ELEON 3.0 please issue:

```
# git clone https://bitbucket.org/semagrow/eleon.git && cd eleon && git fetch && git checkout tags/v3-0
```

To checkout the latest version on the ELEON 3.X branch please issue:

```
# git clone https://bitbucket.org/semagrow/eleon.git && cd eleon && git fetch && git checkout eleon3
```

ELEON dependencies are listed on Table 1. The versions shown on that table are the ones used during development and testing of ELEON v3.0; other versions might also work but have not been tested. Besides dependencies, Java JDK 7 and Apache Ant are also needed in order to compile.

The binary distribution of ELEON only depends on Java SE Runtime Environment 7; all libraries used by ELEON are included in the distribution. The binary distribution is available at http://iit.demokritos.gr/~eleon

After extracting the binary distribution, the following directory structure is created:

```
./ Contains eleon.jar and eleon.sh
./resources/ Contains schemas and vocabularies that ELEON needs for its operation.
./resources/schemas/ Contains testcases for different ELEON domains and applications.
./resources/testcases/ Contains testcases for the Semagrow data source annotation use cases.
```

This structure should not be removed or relocated. The tool is executed by issuing:

```
# java -jar eleon.jar
```

from the ./ directory. In order to have an executable that does not depend on execution path, please modify eleon.sh to set the ELEON installation path. In order to run the tool, Java Runtime Environment version 7 must be installed. JRE 7 can be installed as follows on Debian-based systems:

```
# apt-get install openjdk-7-jre
```

<table>
<thead>
<tr>
<th>Library and version</th>
<th>Available at</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWT 4.3</td>
<td><a href="http://www.eclipse.org/swt">http://www.eclipse.org/swt</a></td>
</tr>
<tr>
<td>Jena 2.11.1, which comprises several jars: jena-core 2.11.1, jena-aqr 2.11.1, jena-iri 1.0.1</td>
<td><a href="http://jena.apache.org">http://jena.apache.org</a></td>
</tr>
<tr>
<td>From Jena: the XML API, xml-apis 1.4.01</td>
<td><a href="http://xerces.apache.org/xml-commons">http://xerces.apache.org/xml-commons</a></td>
</tr>
<tr>
<td>From the XML API: the Xerces imlementation, xercesImpl 2.11.0</td>
<td><a href="http://xerces.apache.org/xerces2-j">http://xerces.apache.org/xerces2-j</a></td>
</tr>
<tr>
<td>From Jena: SLF4J logging, slf4j-api 1.6.4</td>
<td><a href="http://www.slf4j.org">http://www.slf4j.org</a></td>
</tr>
<tr>
<td>From Jena: httpclient 4.2.3</td>
<td><a href="http://hc.apache.org/httpcomponents-client-ga">http://hc.apache.org/httpcomponents-client-ga</a></td>
</tr>
<tr>
<td>From httpclient: httpcore 4.2.2</td>
<td><a href="http://hc.apache.org/httpcomponents-core-ga">http://hc.apache.org/httpcomponents-core-ga</a></td>
</tr>
<tr>
<td>From httpclient: Apache commons logging, commons codec 1.6</td>
<td><a href="http://commons.apache.org">http://commons.apache.org</a></td>
</tr>
<tr>
<td>From HTTPclient/SLF4J: jcl-over-slf4j, 1.6.4</td>
<td><a href="http://www.slf4j.org/legacy.html">http://www.slf4j.org/legacy.html</a></td>
</tr>
</tbody>
</table>
### 3.3 Starting an annotation session

To resume a previously saved annotation session the user has to load the annotations from the file selection dialog at the *File->Open* menu (Figure 4). When the file is loaded the previous annotations will appear in the UI. After the annotation process has finished the user can save the annotations from the *File->Save* and *File->SaveAs* menus. *File->Save* overwrites the file most recently opened or saved-as. When ELEON terminates, all changes since the last save are lost.

After loading an annotations repository, users must authenticate in order to gain write access to a particular segment of the overall annotations repository. Users have read-only access to the complete repository, but write access only to annotations owned by their own username. The level of security provided depends on the backend, and in the case of RDF files discussed here, it amounts to simply declaring the user’s identity in the *Author* menu by either selecting an existing user name or adding a new one (Figure 5). User identities persist in the annotations repository so user names from previous sessions are available after a repository has been loaded.

When starting from scratch, the annotation schema must be selected. The *Annotation Schema* menu offers the list of schemas known to ELEON and allows selecting *exactly one* of the available options (Figure 6). At the moment, the VoID and Sevod schemas are available; Sevod is the pre-set default.

Then the user must select the schemas that will be known to ELEON as the possible schemas used in the data sources being annotated. This can be done from the *Data Schemamenu*. The annotator can select one or more schemas from the menu or load a new one by selecting an OWL, RDF/XML, or TTL file from the local filesystem (Figure 7). The schemas that are per-installed and do not need loading are in the resources/schemas/directory of the distribution and should not be removed or relocated.

It should be noted that the data schemas selected in this menu to not refer to the data schema used in any particular dataset; they are the schemas relevant to this session as a whole and the schemas among which the user may select the schema that any specific dataset follows.
Figure 5: Login menu

Figure 6: Annotation Schema
Figure 7: Data Schema menu

Figure 8: Dataset browsing facets
3.4 Faceted browsing

The annotator then can start the annotation process. To do so he has to select one of the available facets from the Facet panel (Figure 8). A tree will be drawn next to the facet list containing a node named root. This node cannot be deleted.

The annotation process now depends from the type of facet the user selected. First we will describe the process for the “per property” facet and then for the “per entity” facet.

3.4.1 Per Property annotation

Before clicking on the “per property” facet an Annotator (Figure 5) and at least one data schema (Figure 7) have to be selected.

To begin the annotator has to define a new dataset to annotate. To do so he has to right-click the “root” node of the tree and the click on the “Insert dataset label” menu item (Figure 9). A new dialog will appear in which the user must put the name of the dataset (Figure 10). The dataset with the given name will then appear in the tree under the root node (Figure 11). To annotate this dataset the user must click on the node. A table will appear next to the tree containing in one column the properties available for annotation and the in the second column a text field to insert values for those properties (Figure 12).
Figure 11: New dataset inserted

Figure 12: Property editor
Figure 13: Schema selection

Figure 14: Schema added
Figure 15: Remove from tree

Figure 16: Insert dataset
The user then must click on the “void:vocabulary” field. A new window will appear containing a list with all the available schemas, as selected from the Data Schema menu. The user has to select one or more from the list and click the “OK” button (Figure 13). After that the properties from the selected schemas will appear under the selected data source (Figure 14). The annotator can also delete a node from the tree by right-clicking of the node and the click on the “Remove” button. The node and all its children will be removed from the tree (Figure 15).

The annotator can then insert values for the rest of the properties by editing the fields in the property table (Figure 12). Some properties, depending on the Annotation Schema selected (Figure 6), are auto-filled by the tool and cannot be edited by the annotator.

### 3.4.2 Per Entity annotation

The annotation process for the “per entity” facet is similar to that of the “per property” facet. Again the user has to click on the facet, a tree will appear next to the facet’s list containing a root node and the user can insert a dataset to annotate (Figure 16). Then the annotator can insert a new subset by right-clicking on a dataset and selecting “Insert new Dataset” from the menu (Figure 17). A new dialog will appear in which the annotator has to input the regex pattern for the subjects and/or the objects (Figure 18) of this subset. The new subset will then appear as child of the dataset with label “<subject_pattern> ?p <object_pattern>” (Figure 19).
Figure 19: Subset added

Figure 20: Add existing dataset or subset as child
The annotator can also select a previously created dataset or subset to insert by right-clicking on any tree node and selecting “Insert existing dataset or subset as child” (Figure 20).

A dialog will appear and the annotator can choose one dataset or subset to be copied under the selected node (Figure 21). The annotation cannot copy a subset under the root node. Only datasets are permitted under that node. The annotator can add values to the properties of each dataset or subset by using the property editor (Figure 12).

### 3.4.3 Histogram annotation

The annotation process for the “histogram” facet is similar to that of the “per property” and “per entity” facet. Again the user has to click on the facet, a tree will appear next to the facet’s list containing a root node and the user can insert a dataset to annotate (Figure 22). Make sure to select an Annotator and a Schema in order to be able to act on the facet.

The difference from the other facets is that the dataset is automatically generated by a histogram so the user must load an existing dataset by clicking File->Open and selecting a file to load.

After the successful loading of the histogram and selection of the proper facet, ELEON will visualize it based on a tree structure. The automatically generated nodes of the tree are colored in blue and can be edited by any user of the system (Figure 22). A user can click on a node and the available meta-data will appear on a right table, available for editing (Figure 23). Nodes that have been edited by another user of the system can’t be edited by anyone else, based on ELEON authorization policy, and will appear in red color. The user that is currently logged-in will see the nodes, edited by him, with green color. Proper validity checks exist to help user insert valid information. A user can edit URI ranges (Figure 24) as well as numerical ranges (Error! Reference source not found.).
Figure 22: Histogram Tree Structure

Figure 23: Histogram Properties Editor
Figure 24: Editing URI regular expressions

Figure 25: Editing numerical ranges