D2.1.1 Evaluation of the Natural Language Processing Requirements set by the Use Cases

ModelWriter

Text & Model-Synchronized Document Engineering Platform

Work Package: WP2
Task: T2.1 – Data Collection

Edited by:

Claire Gardent <claire.gardent@loria.fr> (CNRS / LORIA)
Samuel Cruz-Lara <samuel.cruz-lara@loria.fr> (University of Lorraine / LORIA)
Anne Monceaux (Airbus)
Marwa Rostren (OBEO)
Ferhat Erata (UNIT)

Date: 16-Sept-2015
Version: 1.0.1

Document History

Version	Author(s)	Date	Remarks
0.1.0	Claire Gardent (CNRS/LORIA), Anne Monceaux (Airbus), Marwa Rostren (Obeo), Anne Monceaux (Airbus)	27-Jan-2015	Draft
0.1.1	Samuel Cruz-Lara (University of Lorraine/LORIA)	10-Feb-2015	Minor modifications to the initial draft
1.0.0	Claire Gardent (CNRS/LORIA), Anne Monceaux (Airbus), Marwa Rostren (Obeo), Anne Monceaux (Airbus)	08-Sep-2015	Initial Release
1.0.1	Ferhat Erata (UNIT)	16-Sep-2015	Minor Edits

Table of Contents

1. Introduction

This deliverable describes the software requirements for the Natural Language Processing (NLP) modules developed within ModelWriter.

These software requirements for the Natural Language Processing (NLP) modules developed within ModelWriter follow from:

- the use cases defined in deliverable D1.2.1 and
- the reversible semantic processing engine to be developed within WP2

The ModelWriter use cases require the development of two main applications:

- A Semantic Annotation module which annotates text with model elements such as Java classes, methods, attributes, Ecore models elements[1], etc. thereby allowing for a two way synchronisation between model and documentation. The Semantic Annotation module will be used and evaluated by Use Cases UC-FR1, UC-FR2, UC-FR4 and UC-FR5.
- A Semantic Analysis module which converts text or elements of a text to a knowledge representation format which can be queried and/or reasoned with (e.g., RDF or OWL)

In addition, the reversible semantic processing engine planned for WP2 ModelWriter should support both text analysis and text generation. That is, it should be able both to convert text to a knowledge or a model representation (analysis) and to produce text from models (generation).

Terms, abbreviations and definitions

Abbreviation	Definition
RDF	Resource Description Framework
RDFS	RDF Schema
OWL	Web Ontology Language
NLP	Natural Language Processing
EMF	Eclipse Modelling Framework
WP	Work Package
UC	Use Case

[1] Eclipse Modelling Framework. http://eclipse.org/modeling/emf/

2. Application Overview

As mentioned above, three main NLP tools will be developed within the framework of the ModelWriter project: a semantic annotator, a semantic parser and a text generator. This section presents the "big picture" for these NLP tools developed within ModelWriter. That is, it describes the objectives of these tools, how they fit into the business process of companies and how they relate to other software systems.

Objectives

In this section, we specify the commonly accepted objectives for the NLP tools developed within ModelWriter. Specifically, we state what business objectives these tools will help achieve.

2.1.1. Semantic Annotation

- Improve customer service (the synchronisation between document and model is automated rather than having to be performed manually and is thereby faster and more rigorous)
- Reduce costs (automate revisions that were previously performed manually by OBEO engineers and developers)
- Replace obsolete technology (use state of the art NLP techniques rather than manual update)

2.1.2. Semantic Analysis

- Reduce costs (automate normalisation of text that were previously done manually by AIRBUS engineers)
- Improve user service (by converting text to a a knowledge representation format that supports querying and reasoning)
- Piloting a new technology (deep semantic reasoning using automating text-to-knowledge conversion and OWL querying/reasoning)
- Enable the integration with the knowledge representation coming from the synchronized models (through the foreseen query or reasoning mechanisms)

2.1.3. Reversible Semantic Engine

- Pilote a new technology which potentially supports propagating model changes to text and vice versa.

Business Process

This section describes how the NLP tools will be used in the context of the ModelWriter project and more specifically, in the context of the ModelWriter usecases.

2.1.4. Semantic Annotation

- In the context of OBEO, the Semantic Annotation Process will help users to explicitly and semi-automatically define mappings between some specific parts of textual documents and model elements. It's an upstream process of the synchronization since text ↔ model synchronization cannot be handled without handling mappings first.

- In the context of the Airbus Group, Semantic Annotation will enable synchronizing between OWL models produced by Airbus and associated documentation such as system installation design principles (SIDP) document or similar.

2.1.5. Semantic Analysis

- In the context of the Airbus Group, Semantic Analysis will enable the retrieval and representation of rules (from documents about system installation rules) currently expressed in an unstructured way inside text. This enables new synchronization means to connect the text to the models' elements it refers to. In this context, it is foreseen that a given rule might be actually documented partially in a text and partially in a model. For example the text part might identify an equipment (e.g., a given pipe type) and express some installation condition (in a wet zone) while a model (e.g., a table or a 2D drawing) might contain other relevant information (e.g., required distance between the equipment and another object).

2.1.6. Reversible Semantic Engine

- Stand Alone Application

UserRoles and Responsabilities

This section describes who the users are and how the systems fits into what they do. In essence, the NLP tools developed within ModelWriter will help company engineers and developers maintain an up to date synchronisation between documents and models.

2.1.7. Semantic Annotation

Developers:

- Search for text (resp. model) that is related to a change in the model (resp. text).
- Maintain synchronisation between program and documentation.

2.1.8. Semantic Analysis

Developers:

- Convert documentation to normalised format
- Search documentation for applicable rules

2.1.9. Reversible Semantic Engine

Developers

- Construct model from text
- Generate text from model

Interactions with Other Systems

In this section, we describe how the NLP tools related to other existing systems in the ModelWriter approach. A more detailed specification of these interactions is provided in Deliverable D6.2.1 Architectural Design.

2.1.10. Semantic Annotation

The semantic annotation interacts with the Text Connectors, the Knowledge Semantic Services and Knowledge Base Repository as described in Deliverable D6.2.1.

The text connectors provides the conversion between the text content of a document (from the corresponding authoring tool) and the plain text format processed by the NLP tools.

The Knowledge Base Repository manages

1. The mapping between texts and models, by relying on the annotation chosen by users. The mapping must be explicit and must link a part of the text to one or more models elements.
2. A Bi-directional synchronization, based on the mapping between texts and models. The synchronization must be established by comparing the current state of texts and models with the previous state before the modification.
3. Consistency checks between the texts and the models to provide information about asynchronous parts.

Input: The input of the semantic annotation module is a document that can be in different formats (html, doc, txstile, xlsx, xml, etc.)

Output: The output of the semantic annotation module is an annotated text file where text fragments are annotated with model elements.

Semantic Annotation: annotates the input text document with useful information to:

1. Help users at the editor level to choose the model annotation which conforms the best with the current context;
2. To provide at the knowledge level the mapping between the annotated part of the text and the model element.

Automated reasoner: checks consistency of knowledge store or of new knowledge against existing knowledge store.

2.1.11. Semantic Analysis

The interactions between the semantic analysis module, the input text and the WP3 ModelWriter reasoning modules are as follows.

Input: The input of the semantic analysis module is a text in ASCII format derived from some input document by the Convertor module. As indicated above, the input document may be in different formats (html, xlsx, xml, …). Most probably the user editor is MSWord, but as there will be a connector (developed by OBEO) the text format shall be decided by the technical partners.

Output: The output of the semantic analysis module is either natural language rules that have been normalised or knowledge that was extracted from the input text and is represented in a standard knowledge representation language.

Convertor: maps the input document into text. The convertor might include several submodules depending on the number of input formats that should be considered.

Semantic Analysis: converts text to normalised rules (natural language) or to knowledge (formal language)

Query Engine: interpret formal queries against knowledge store and returns all answers that satisfies the query given the knowledge store

Automated reasoner: checks consistency of knowledge store or of new knowledge against existing knowledge store.

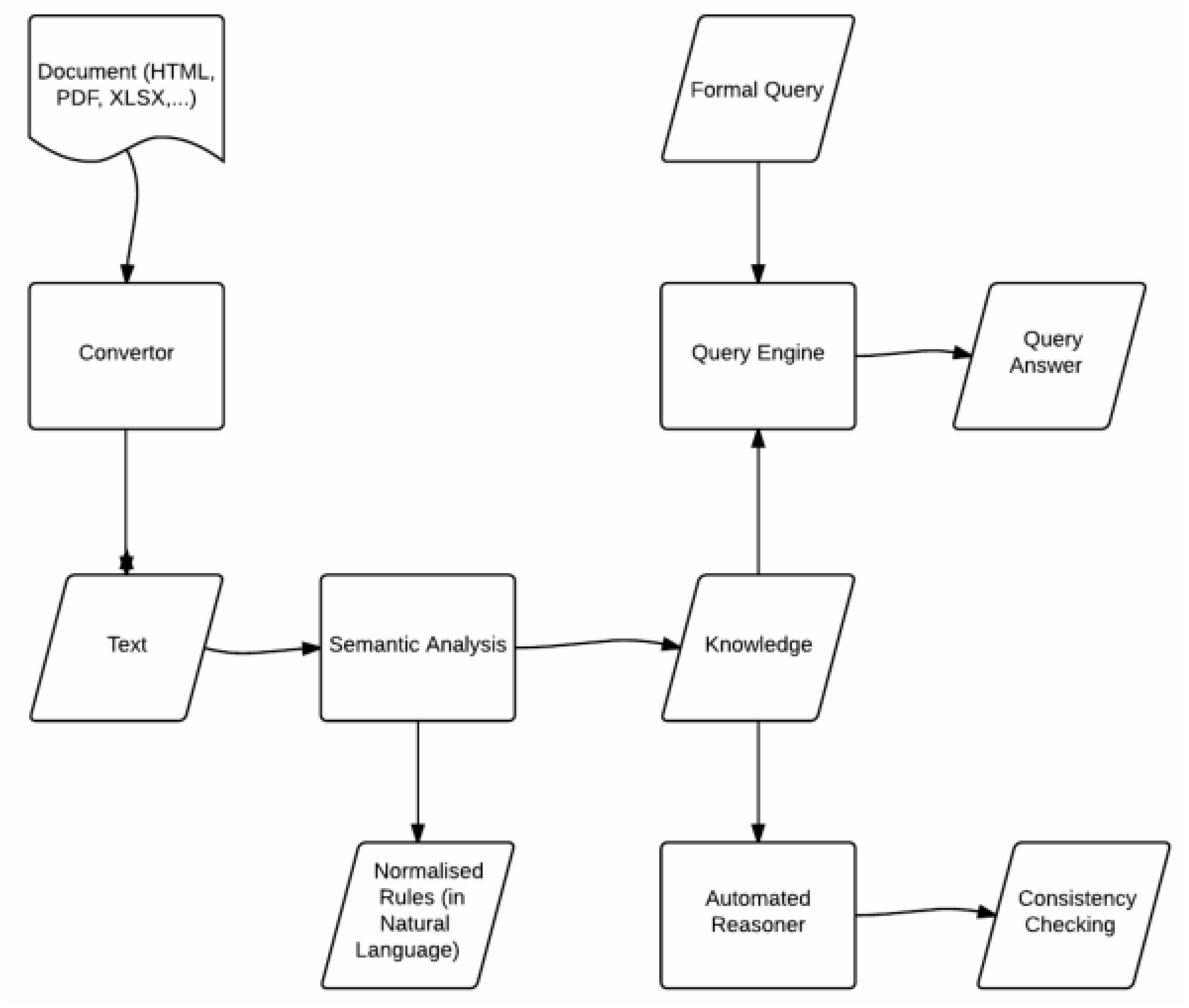

Figure 2: Semantic Analysis.

2.1.12. Reversible Semantic Parsing

The flowchart (see Figure 3) for the reversible semantic processing engine is shown below.

Input: a document (html, doc, txt, xlsx, xml …) or some data.

Output: some data (analysis) or some text (generation).

Lexicon: a data structure mapping words and data to syntactic rules.

Grammar: a set of syntactic rules and some rule combination operations.

Algorithms: the reversible engine includes both a parsing algorithm (to map text to data) and a generation algorithm (to map data to text).

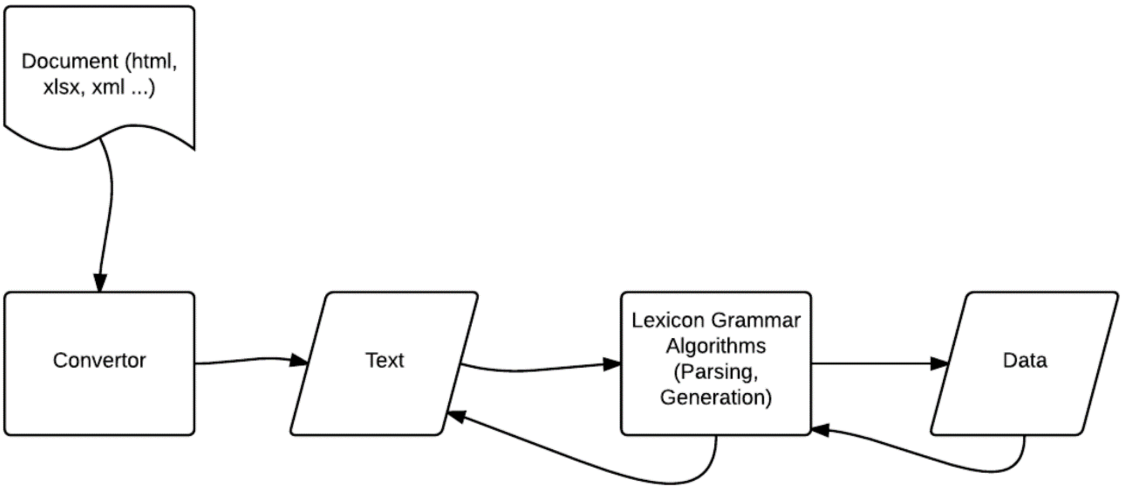
Document (html, xlsx, xml ...)
Convertor
Text
Lexicon Grammar Algorithms (Parsing, Generation)
Data

Figure 3: Reversible Semantic Parsing.

3. Functionnal Requirements

The functional requirements of the NLP modules are categorized in three areas: semantic annotation, semantic analysis and reversible semantic processing.

Each of these are further decomposed into components (convertor, preprocessing, parsing, etc.), all of which are MUST do. Each of these components can be developed with very simple models at first, which can then be made more complex.

Identifier	Semantic-Processor
Type	Functional
Importance	MUST
Relations	Requires: Semantic-Annotator, Semantic-Analyser, Reversible-Semantic-Engine
Description	The NLP modules developed by ModelWriter for mapping text to data or data to text
Source	Development, Research, Assets

Semantic Annotator

Identifier	Semantic-Annotator
Type	Functional
Importance	MUST
Relations	Required by Semantic-Processor
Description	An independent system which annotates documents with model elements
Source	Development, Research, Available Software

Identifier	Convertor
Type	Functional
Importance	MUST
Relations	Required by Semantic-Annotator

Description	Converts document (html, xlsx, xml, etc.) to text
Source	Development, Research, Available ontoloSoftware

Identifier	Sentence-Segmenter
Type	Functional
Importance	MUST
Relations	Required by Semantic-Annotator
Description	Segments text into sentences
Source	Development, Research, Available Software

Identifier	Tokenizer
Type	Functional
Importance	MUST
Relations	Required by Semantic-Annotator
Description	Segment sentences into words/collocations
Source	Development, Research, Available Software

Identifier	Annotator
Type	Functional
Importance	MUST
Relations	Required by Semantic-Annotator
Description	Labels words/collocation with matching model elements
Source	Development, Research, Available Software

Identifier	Model or Knowledge Base

Type	Functional
Importance	MUST
Relations	Required by Semantic-Annotator
Description	A list of model or KB elements with which the text must be annotated
Source	Obeo SIRIUS model and Java Code ; Airbus KB of components and SIDP rules

Semantic Analysis

Identifier	Semantic-Analyser
Type	Functional
Importance	MUST
Relations	Required by Semantic Processor
Description	An independent system which maps text to data that can be queried and reasoned with
Source	Development, Research, Available Software

Identifier	Convertor
Type	Functional
Importance	MUST
Relations	Required by Semantic-Analyser
Description	Converts document (html, xlsx, xml, etc.) to text
Source	Development, Research, Available Software

Identifier	Sentence-Segmenter
Type	Functional
Importance	MUST

Relations	Required by Semantic-Analyser
Description	Segments text into sentences
Source	Development, Research, Available Software

Identifier	Tokenizer
Type	Functional
Importance	MUST
Relations	Required by Semantic-Analyser
Description	Segment sentences into words/collocations
Source	Development, Research, Available Software

Identifier	Term-Detector
Type	Functional
Importance	MUST
Relations	Required by Semantic-Analyser
Description	Identifies terms i.e., words/collocations which denote elements (concepts, relations) of the domain ontology or of the input model
Source	Development, Research, Available Software

Identifier	Relation-Extractor
Type	Functional
Importance	MUST
Relations	Required by Semantic-Analyser
Description	Extracts (subject,relation,object) data triples from text
Source	Development, Research, Available Software

Identifier	Rule-Identifier

Type	Functional
Importance	COULD
Relations	Required by Semantic-Analyser
Description	Identifies natural language segments in free text which encode SIDP rules
Source	Development, Research, Available Software

Identifier	Term-Extractor
Type	Functional
Importance	COULD
Relations	Required by Semantic-Analyser
Description	Extracts domain specific terms from text and groups them into synonymic sets
Source	Development, Research, Available Software

Identifier	Domain-Ontology
Type	Non Functional
Importance	MUST
Relations	Required by Semantic-Analyser
Description	Domain specific ontology
Source	AIRBUS

Reversible Semantic Engine

Identifier	Reversible-Semantic-Engine
Type	Functional
Importance	MUST
Relations	Required by Semantic Processor

Description	An independent system which maps text to data and data to text
Source	Development, Research, Available Software

Identifier	Lexicon
Type	Functional
Importance	MUST
Relations	Required by Reversible-Semantic-Engine
Description	A data structure mapping words and model elements to grammar units
Source	Development, Research, Available Software

Identifier	Lexicon-Extractor
Type	Functional
Importance	COULD
Relations	Alternative to Lexicon
Description	Extracts Lexicon from semantically annotated, parsed text
Source	Development, Research, Available Software

Identifier	Grammar
Type	Functional
Importance	MUST
Relations	Required by Reversible-Semantic-Engine
Description	A data structure describing the syntax of natural language
Source	Development, Research, Available Software

Identifier	Grammar-Extractor
Type	Functional
Importance	COULD

Relations	Required by Reversible-Semantic-Engine
Description	Induces Grammar from semantically annotated, parsed text
Source	Development, Research, Available Software

Identifier	Parser
Type	Functional
Importance	MUST
Relations	Required by Reversible-Semantic-Engine
Description	Maps text to data using Lexicon and Grammar
Source	Development, Research, Available Software

Identifier	Generator
Type	Functional
Importance	MUST
Relations	Required by Reversible-Semantic-Engine
Description	Maps data to text using Lexicon and Grammar
Source	Development, Research, Available Software