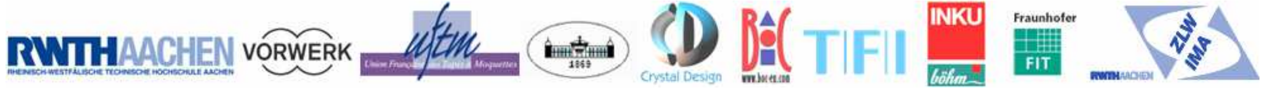


AsIsKnown

A semantic-based knowledge flow system for

the European home textiles industry

Work package 3:
Common sense ontology engineering

Deliverable D10 “Software Tool for Engineering Common Sense Ontology”

Lead participant: IPP-BAS

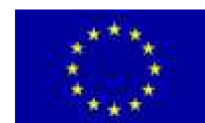
Nature: Software

Dissemination level: PU

Delivery date: 10 PM

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.



This document has been produced in the context of the AsIsKnown Project. The AsIsKnown project is part of the European Community's Sixth Framework Program for research and development and is as such funded by the European Commission. All information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability is respect of this document, which is merely representing the authors view.

Context

WP 3	Common sense ontology engineering
Task 3.5	Implementation of the software tools.
Dependencies	This deliverable requires user requirement input from 3.3

Contributors:	***Reviewers:***
Alexander Simov (IPP-BAS) Anelia Tincheva (IPP-BAS) Borislav Kirilov (IPP-BAS) Kiril Simov (IPP-BAS) Simeon Manchev (IPP-BAS)	Kiril Simov (IPP-BAS)

Approved by: Kiril Simov, Bulgaria as WP3 Head

EN 14041:2004
Notified Body
No. 1658

Testing laboratory, inspection and certification body recognized according to §28 of the “Landesbauordnung NRW” for flame resistant floor coverings

Deutscher Akkreditierungs Rat
DAP-PL-3457.00
Accredited for the

Executive Summary

This report presents the Application Program Interface to the AsIsKnown Onto System (AIKOS). The API reflects the ontology related services described in Deliverable 7 and depends on the tools described in Deliverable 9. The main objective is to create a software system able to support the ontology related services necessary for the AsIsKnown system with maximum reuse of existing systems.

EN 14041:2004
Notified Body
No. 1658

Testing laboratory, inspection and certification body recognized according to §28 of the “Landesbauordnung NRW” for flame resistant floor coverings

Deutscher Akkreditierungs Rat
DAP-PL-3457.00
Accredited for the

Table of Contents

EXECUTIVE SUMMARY....4

TABLE OF CONTENTS....5

TABLE OF FIGURES....6

LIST OF ABBREVIATIONS....7

1 INTRODUCTION AND PROBLEM STATEMENT....8

2 IMPLEMENTATION OF THE ONTOLOGY RELATED SERVICES....9

3 CONCLUSIONS AND OUTLOOK....20

REFERENCES....21

EN 14041:2004
Notified Body
No. 1658

Testing laboratory, inspection and certification body recognized according to §28 of the “Landesbauordnung NRW” for flame resistant floor coverings

Deutscher Akkreditierungs Rat
DAP-PL-3457.00
Accredited for the

Table of Figures

Figure 1: JenaDBImport settings tab... 11

CE
EN 14041:2004
Notified Body
No. 1658

Testing laboratory, inspection and certification body
recognized according to §28 of the “Landesbauordnung
NRW” for flame resistant floor coverings

Deutscher
Akkreditierungs
Rat
DAR
DAP-PL-3457.00
Accredited for the

List of Abbreviations

AIKOS	AsIsKnown Ontology System
API	Application Programming Interface
DIG	Iterface for Description Logic Reasoners
OWL	Web Ontology Language
OWL-DL	Web Ontology Language – Description Logic sublanguage
SWRL	Rule Language based on a combination of the OWL DL and OWL Lite with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language. SWRL includes a high-level abstract syntax for Horn-like rules in both the OWL DL and OWL Lite sublanguages of OWL. A model-theoretic semantics is given to provide the formal meaning for OWL ontologies including rules written in this abstract syntax. An XML syntax based on RuleML and the OWL XML Presentation
XML	Extensible Markup Language

EN 14041:2004
Notified Body
No. 1658

Testing laboratory, inspection and certification body recognized according to §28 of the “Landesbauordnung NRW” for flame resistant floor coverings

Deutscher Akkreditierungs Rat
DAP-PL-3457.00
Accredited for the

1 Introduction and Problem Statement

Here we discuss the implementation of the services that are necessary to support the incorporation of the Ontology Management System within the overall AsIsKnown Architecture. The AsIsKnown Onto System (AIKOS) provides the services necessary for supporting the information flows within AsIsKnown System. The AIKOS services are:

- Editing of Ontologies
- Storage of Ontologies
- Inference Services (Consistency, Realization, Classification, Navigation)
- Mapping of Ontologies and Content Systems
- Semantic Annotation with Ontological Information

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.

2 Implementation of the Ontology Related Services

In this section we present a summary of the requirements for the services which the AIKOS minimally will have to provide. For each service we describe the component that is responsible for implementation of the service and the functional interface to it.

Editing of Ontologies

- *Analytical Ontology Editing* – these services have to support the editing of the ontology directly in some serialization of the ontological language. For example, XML representation of OWL;
- *Visual Ontology Editing* – these services have to support visualization of a portion of an ontology and support editing via forms or via direct manipulation of the graphical representation;
- *Specific to AsIsKnown Editing* – these services have to support designing and editing of templates for reflecting the ontology structure;
- *Lexicon Editing* – these services have to support the creation of lexical entries and their alignment to the ontology.

The first two services are supported by Protégé (http://protege.stanford.edu). The advantages of Protégé compared with other Ontology Editors include the following:

Open source and easy to install

Protégé is an open source platform for ontology creation and visual ontology manipulation. The installation process is easy.

- *Plug-in Architecture*

  The most valuable advantage of Protégé is its Plug-in architecture. As described before, the implementation of AIKOS will need some project specific modules and views of the ontology. Using Protégé in that direction provides us with a platform for the development of additional functionalities specific for the project. The developed modules will interoperate with the available ones in Protégé. The most convenient way for our purpose is the development of “tab plug-ins”. The tabs may be switched on and off depending on the user’s needs.

- *Supports OWL-DL Ontologies*

  The ontology that AIKOS will create and manipulate is OWL-DL (*Description Logic*) ontology. It means that the chosen Ontology Editor must support the OWL-DL subset of OWL definition language as an input and output format. Protégé supports that sublanguage via Protégé-OWL Plug-in which is configured to be switched on when the editor is started. The OWL-DL subset was chosen as suitable format for the ontology because it uses first order logic, i.e. axioms, rules, restrictions and inference capabilities can be used for manipulation of the domain. Protégé supports all these actions and when tested, Protégé showed very satisfactory results.

- *Manipulation of huge ontologies*

  When we tested a number of ontology tools Protégé showed the most satisfactory results on loading different ontologies.

- *Protégé Maintenance*

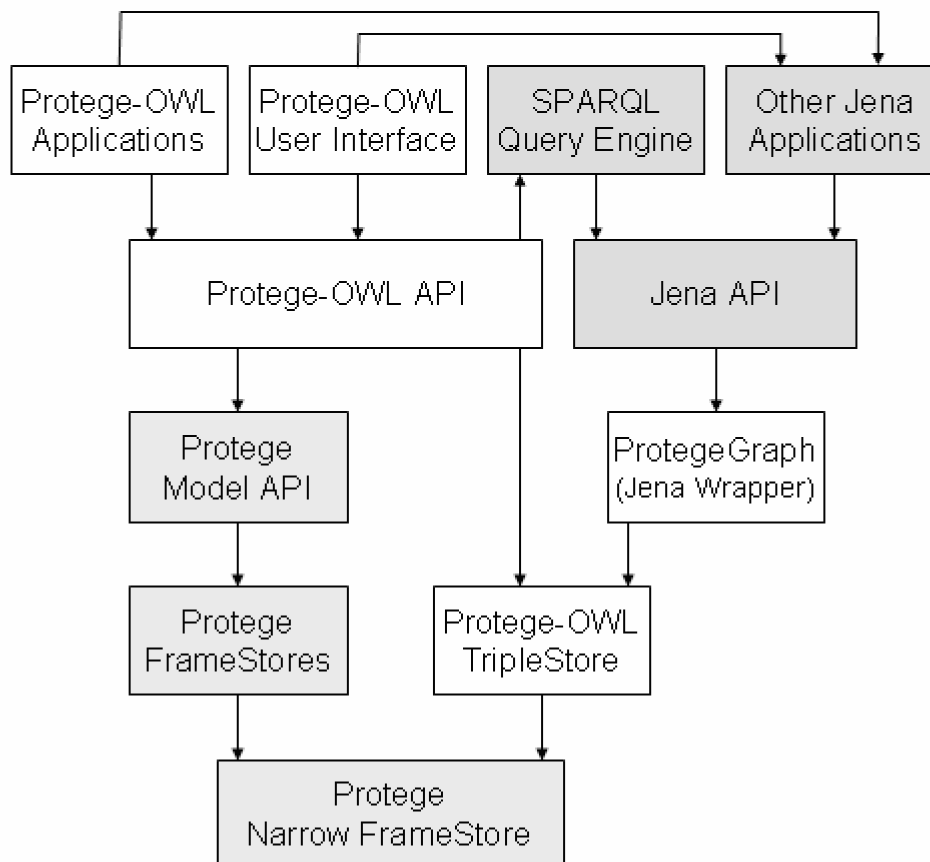
  Protégé has a satisfactory maintenance environment. There are forums, developer email list in case of a problem, FAQ list, Plug-in library that contains a lot of plug-ins separated by their functionality, tutorials for using Protégé and developing modules for Protégé, updates and new versions are also available. For the developers it is a good feature that there is a javadoc documentation generated for the API.

- *Interoperability with Jena Model and Pellet Reasoner*

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.

Protégé has a high level interoperability with Jena model. The Protégé-OWL Plug-in uses Jena OntModel. They are integrated in the following way:



Using the Protégé-Jena integration, our team created a plug-in that loads an ontology from a remote Jena repository, saved in MySQL or Oracle RDMS and exports it back to the repository.
Protégé communicates with a reasoner by the DIG protocol. The ontology editor has integration with a variety of reasoners but our team chose Pallet for the purpose. The main reason is that Pallet is open source and supports DL reasoning. The inference with Pallet and Protégé is described in details in the next points.
For the moment editing services specific to AsIsKnown are not identified. If necessary they will be added on the later stages of the project. Lexicon editing is part of the language processing system which will be described below.
The lexicon editing will be done in CLaRK system, which is an XML-based system for language resources creation and manipulation. It is described below.

Storage of Ontologies

In the deliverable 7 one of the requirements is the ontology to be represented in a modular way. But after the analysis of information representation we have decided that information about instances will be stored separately from the conceptual information. Thus, we have not checked the selected system if it handles large instance repositories. The supported services are:

- *Repository Management* – these services have to support the creation of a new repository, deletion, renaming, coping of an existing one. When these modifications happen to have impact on other repositories appropriate messages will be presented to the user;

Information Society Technologies
AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.

- *Ontology Management* – these services have to support the manipulation of the ontological information within one repository – the standard operations of adding, deleting and updating on the level of class, property and instance descriptions. Corresponding checks for validity and consistency will be performed;
- *Modularization* – the services of this group have to support the interaction between repositories via the import operators.

For repository maintenance we have chosen the Jena API. One of the main reasons for our choice is that the Jena Model is fully integrated with Protégé-OWL. The other reason is that Pellet inference functionality is able to infer the Jena Model. Testing Jena basic functions, sufficient for the project, showed satisfactory results.

- *Repository Management*

Our tests were concentrated on surveying the repository capabilities of Jena with RDMS and more specifically with MySQL. We chose this kind of RDMS because it is free, it is not as heavy as the Oracle RDMS and additionally it is reliable enough. It satisfies the purposes of the project. While testing, for management, we used MySQL Query Browser for the creation of a new repository, deletion, renaming, coping of an existing one. For the purpose of AIKOS we developed a configuring tool that sends queries to MySQL RDMS for base repositories manipulation. In that tool we can show appropriate messages to the user which are specific for AIKOS actions over the repository.

- *Ontology Management*

For our purpose we developed a plug-in that integrates the Protégé and Jena. The scenario is as follows:

1. Start Protégé editor and configure “JenaDBImport” plug-in. A new tab “JenaDBImport” is visualized.

2. Select the tab and fill in or load the last user’s setting as connection and repository parameters. There are two types of RDMS possible: MySQL and Oracle. Specify the repository and the model name for the ontology to be loaded.

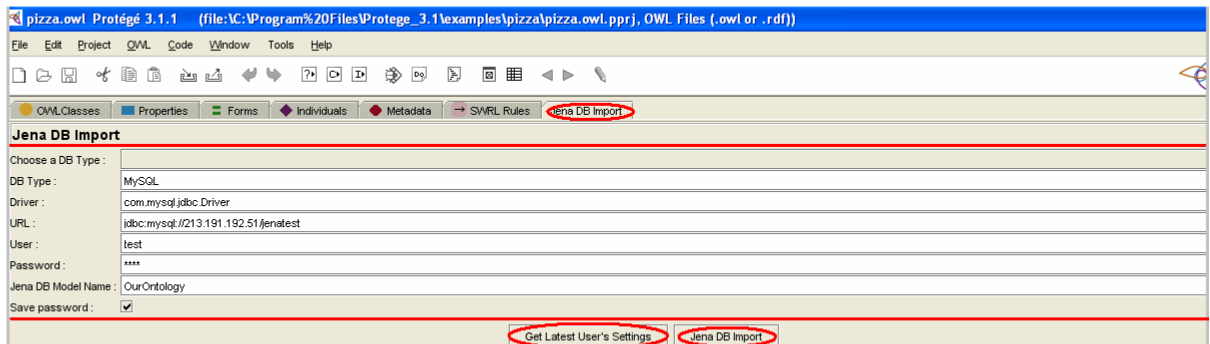


Figure 1: JenaDBImport settings tab.

3. If all the parameters are valid the current model is closed and saved (if needed) and the new one is loaded and the user can manipulate it in the Protégé Editor.

4. For exporting the manipulated ontology back to the repository, the user has to select “Main menu” → “Export Formats” → “Export to JenaDBmodel”. A modal dialog is visualized where the connection and repository parameters have to be specified. There is an option to replace the model in the RDMS before export. It is useful, because in that way no conflicts in the repository can occur. A suitable message concerning the result of the export action is shown to the user.

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.

- *Modularization*

  The conceptual model (classes’ taxonomy and relations between them) and the instances will be stored in different repositories. Taking that fact into consideration, the content that has to be stored is not so much and development of special modularization functionality is not necessary for the moment. We rely on the import functionality of OWL-DL to support the different modules of the ontology. If it is necessary to incorporate also modularization of the conceptual information in the process of integration with the other components of AsIsKnown system we will add such functionality.

Inference Services

Our work on supplying inference actions over ontology was done in two directions. The first one was to provide inference in the Protégé Editor, which to be used when the knowledge engineer is creating or modifying the ontology, and the second one was to develop inference API without a GUI presentation, which to be used by the other components of AsIsKnown system, when they need access to the ontology. Both approaches had to cover the specifications for inference services needed in AIKOS. These requirements are as follows:

- *Navigation* – these services have to support traversing of explicit information represented in the ontology;
- *Consistency* – these services have to support the check for contradiction within one repository. At least the consistency of the whole ontology and the consistency of a class description with respect to an ontology have to be supported;
- *Classification* – this service has to support for a class ***C*** the finding of the minimal classes in the ontology that are more general than ***C*** and the maximal classes in the ontology that are more specific than ***C***.
- *Realization* – this service has to support for an instance ***I*** the finding of the minimal classes in the ontology that describe ***I***.
- *Instance Checking* – this service has to support for an instance ***I*** and a class ***C*** the checking whether ***I*** is an instance of ***C***.
- *Retrieval* – this service has to support for a class C the finding of all of its instances in the ontology.
- *Rules* – this service has to support the creation of user defined rules. User-defined rules are a very powerful mechanism for changing the ontology. There will be a mechanism for defining the scope of such rules to one or few repositories.

Using Protégé Editor integrated with Pellet reasoner

As mentioned before Protégé can communicate with a reasoner via the DIG protocol. To use inference services in Protégé, the user has to install a reasoner first. The installation of Pellet can be found at http://www.mindswap.org/2003/pellet/download.shtml. Protégé should be configured from “OWL menu” → “OWL Preferences” to listen to port: 8081. Before using inference services the reasoner has to be started. In the case of Pellet the pellet-dig.bat file has to be started. It is available in the Pellet installation package. Here is a description of reasoning services:

- *Navigation*

  list of all subclasses

  This service is available by getting the context menu for a given class, from the subclass hierarchy tree, and selecting “Search and view” → “Show list of subclasses”.

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.

The result is shown in a result panel at the bottom of the main panel, visualizing the subclasses and their axiomatic description. When double clicking on a class from the result panel, navigation to the selected class is accomplished.

list of all super classes

It is obvious from the UI subclass hierarchy. All super classes are visible in the subclass hierarchy tree.

list of inferred super classes

It’s available by getting the context menu for a given class, from the subclass hierarchy tree, and selecting “Get Inferred Super classes”. The result is shown in a modal dialog. The inferred super classes are not only the explicitly defined direct ones but the inferred by the reasoner on the base of axiomatic description of the explicitly definedones.

- *Consistency*

In Protégé version which is integrated with Pellet, the reasoner can checked the consistency of a selected class or the whole ontology. The inconsistent data is visualized. “Classify taxonomy” service is also available. It opens a new subclass hierarchy tree that contains the inferred classes and relations on the basis of the axiomatic descriptions. The inconsistencies are also shown. The actions of the reasoner for the classification are visualized in a result panel.

- *Classification*

list of direct subclasses

The list of direct subclasses is available from the subclass hierarchy tree view, so accomplishing such functionality is not necessary.

list of direct super classes

The list is available from the hierarchy visualization on the base of the sub-super class relation

- *Realization*

The following reasoning services are implemented for *instance retrieval*. In Protégé the idea that an instance is closely related to a class is implemented. It means that a new instance can be included in the ontology in the context of a member of a particular class and its super classes. No other way for defining instances is possible. Selecting instances is also in the context of a class. Get all direct and indirect instances for a class. For a visualized instance, in the context of a specific class, one can get all the types of concepts that cover the instance. Another way of selecting an instance by name search in the whole Knowledge Base is by using “StringSearch” plug-in. Then information about the class to which the instance belongs is available.

- *Instance Checking*

It is possible to check that indirectly. By using “StringSearch” plug-in, the user enters a string that represents instance name and the minimal class for the specified instance is found.

- *Retrieval*

The first way to retrieve the instances for a particular class is by using the reasoners service “Compute individuals belonging to class” from the context menu for a

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.

particular class from the subclass hierarchy. The result contains also the inferred instances.

Another way to get the instances for a class is to use the “Individual Browser” as described above.

In Protégé Plug-in library contains a plug-in “InstanceXL”. The instances are extracted by specifying the class and if there are any instances for the specified concept they are visualized in a table view. The table view reflects the values of different properties specified as columns of the table including system ones like “:NAME” and “rdf:type”. The result can be filtered for properties in which the user is interested. The current table can be extracted in different formats and the user is able to search in it by specifying string values and regular expressions.

The other way for retrieving instances for a given class is by using “StringSearch” plug-in. It is available with the Protégé installation package. The search is in the whole KB and all the matching values are displayed in a result panel. The result is based on the matched value and it can be sorted and displayed. The other information specified is the type of the found resource: class, property, instance and the concrete name for the found item. Modification of the result panel reflects the whole model.

- *Rules*

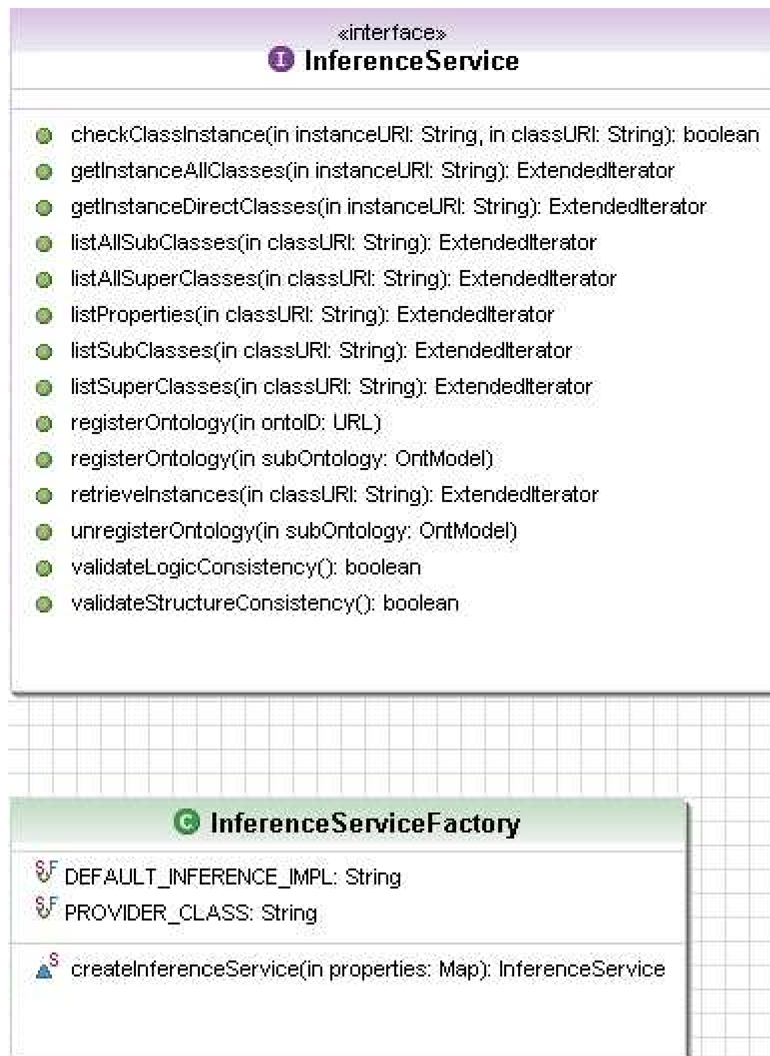
  Rules can be defined by the “SWRL Tab” available in the installation package of Protégé The user first should load SWRL ontology from the World Wide Web and afterwards a table view is available where rules can be defined. SWRL rules are saved as OWL individuals with their associated OWL file. There is a user-friendly editor for rule definition. Initially SWRL has no inference capabilities so it’s integrated with the Jess rule engine within Protégé-OWL to perform inference with SWRL rules. Jess is a Java-based rule engine. Jess system consists of a rule base, fact base, and an execution engine. Using “SWRL Tab” rules can be saved. If there exist defined rules for a resource they can be extracted. If there are saved rules in ontology, while loading it in Protégé, the rules and “SWRL Tab” are loaded automatically. The SWRL rules cannot be evaluated by Pellet reasoner. The reasoner sends warnings concerning interpretation problems when “Check Consistency” option is chosen.

Inference service API

Apart from the integration of the Pellet (http://www.mindswap.org/2003/pellet) reasoner within Protégé (http://protege.stanford.edu/), we provide an application independent API supporting the functionalities needed for the AIKOS inference services. Its main purpose is to offer a uniform interface for integration of other components of the AIKOS architecture. The interface supports a set of methods for each required area of functionalities.
The following UML diagram reveals the structure of the interface:

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.



The interchange data model used in the API reuses the native Jena (http://jena.sourceforge.net) object model, which is also integrated in the majority of the knowledge management systems. In practice this model has become a standard for conceptual model representation.

Additionally, we provide a factory class for instantiating custom implementations of the inference API in a uniform way. This gives an isolation layer for third party components not to depend on concrete implementations and also prevents them in case of implementation changes or replacement. The classes in this package do not expose any dependency to any interface implementation.

Usage

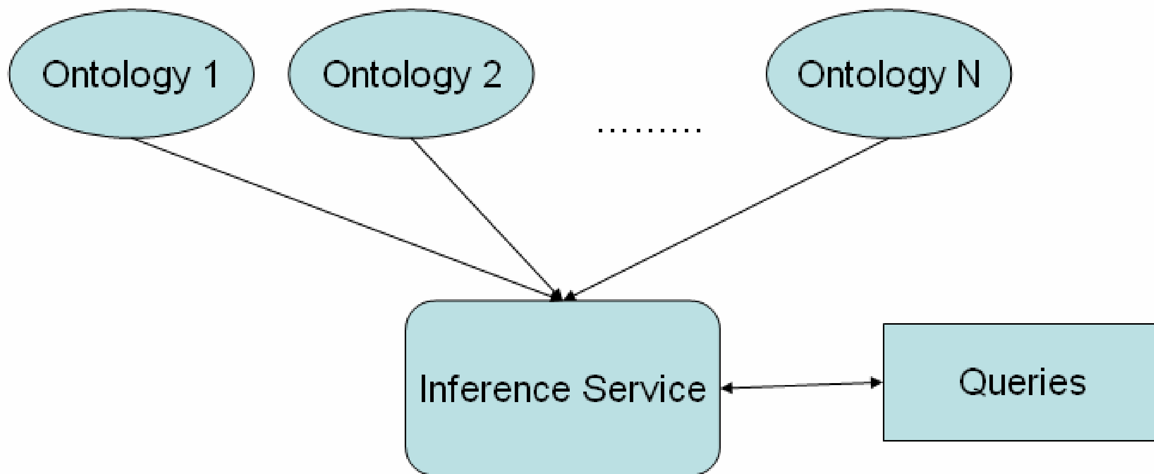
Instances of the API implementation can be created by the supplied factory method (*InferenceServiceFactory#createInferenceService*()). By supplying a null or empty properties argument, the factory creates a default implementation instance (described in the next section). If a specific implementation is required, a particular provider property (*PROVIDER_CLASS*) has to be set in the properties map. The value of this property is a fully quantified class name (string) of implementation of InferenceService API, which must be available at runtime in the classpath of the current virtual machine instance.

Having initialized the inference service, the user can dynamically register/un-register conceptual models (ontologies or fragments of them) on which certain inference queries may

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.

be performed. The registration can be performed either by specifying a concrete model or by URL where the model is located. Initially there are no models registered.



Supplied Default Implementation

Along with the API we provide a default base implementation containing Pellet (http://www.mindswap.org/2003/pellet/) reasoner. Pellet is an open-source OWL-DL reasoner, widely accepted in the various knowledge-based systems and frameworks.

Mapping of Ontologies and Content Systems

- *Mapping Creation* – this service has to support the creation of mapping rules from an external with respect to the AsIsKnown Core Component conceptualization to the AsIsKnown ontology. The rules will be determined on the basis of the classes and properties within the ontology. If there are missing classes or properties they could be added to the ontology with the Editing services. The result of the application of a mapping rule has to be valid (consistent) OWL statements about some instances;
- *Mapping Rule Application* – this service has to support the application of the mapping rules from a given mapping and storing of the resulting OWL statements in a given repository. Reaction of inconsistency or other problems will be supported by the Storage of Ontologies services;
- *Lexicon-based Mapping* – this service has to support the mapping from the ontology to a system of menus and back. The items of the menus are defined with the help of the lexicons aligned to the ontology.

Because the mapping rules are working only for *instance data* which is manipulated by Smart Profiler the Mapping Creation and Mapping Rule Application services will be implemented within Smart Profiler.
Lexicon-based Mapping will be part of CLaRK system which is described below.

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.

Semantic Annotation with Ontological Information

- *Text Analysing* – these services have to support the whole range of processing of the multimedia documents from converting them from original format into a common (XML-based) format to partial parsing (if necessary). We cannot provide such services for all languages (English, German, French, Bulgarian), but we will provide the basic tools for supporting such services and we will provide some for Bulgarian and English.
- *Semantic Annotation* – these services have to support the annotation with ontological information of the already processed text. Minimally, the services have to provide mechanisms for annotation of phrases with ontological classes and for annotation of relations between entities.

Both of these services together with lexicon management will be supported by CLaRK system. The necessary functionality will be tuned to the requirements of AsIsKnown system when the integration of CLaRK with the other components (mainly the Trend Analyzer and AIKOS) is done during the next 8 months of the project. The following is a description of the main technologies behind CLaRK and its tools.

CLaRK System – XML-based system for natural language processing

The CLaRK System [1], [2], [3] and [4] is an XML-based system for natural language processing. It incorporates several technologies:

- XML technology;
- Unicode;
- Regular Cascade Grammars;
- Constraints over XML Documents.

On the basis of these technologies the following tools are implemented: XML Editor, Unicode Tokeniser, Sorting tool, Removing and Extracting tool, Concordancer, XSLT tool, Cascaded Regular Grammar tool, etc. The system is implemented in Java and it is freely available at the project site http://www.bultreebank.org/clark/index.html.

Unicode tokenization

From the point of view of corpora development the textual nodes usually are considered as a list of textual elements, such as wordforms, punctuation and other tokens. In order to provide possibility for imposing constraints over the textual node and to segment them in meaningful way, the CLaRK System supports a user-defined hierarchy of tokenisers. At the very basic level the user can define a tokeniser in terms of a set of token types. In this basic tokeniser each token type is defined by a set of UNICODE symbols. Above this basic level tokenisers, the user can define other tokenisers, for which the token types are defined as regular expressions over the tokens of some other tokeniser, the so called parent tokeniser. For each tokeniser an alphabetical order over the token types is defined. This order is used for operations like comparison between two tokens, sorting and similar.

Cascaded Regular Grammars

The regular grammars are the basic mechanism for linguistic processing of the content of an XML document within the system. The regular grammar processor applies a set of rules over

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.

the content of some elements in the document and incorporates the categories of the rules back in the document as XML mark-up. The content is processed before the application of the grammar rules in the following way: textual nodes are tokenized with respect to some appropriate tokeniser, the element nodes are textualized on the basis of XPath expressions that determine the important information about the element. The recognized word is substituted by a new XML mark-up, which can or can not contain the word.

Constraints

The constraints that we implemented in the CLaRK System are generally based on the XPath language. We use XPath expressions to determine some data within one or several XML documents and thus we evaluate some predicates over the data. Generally, there are two modes of using a constraint. In the first mode the constraint is used for validity check, similar to the validity check, which is based on DTD or XML schema. In the second mode, the constraint is used to support the change of the document in order it to satisfy the constraint. In this section we present the constraints that are implemented in the system.

The constraints in the CLaRK system are defined in the following way:

(Selector, Condition, Event, Action)

where the selector defines to which node(s) in the document the constraint is applicable; the condition defines the state of the document when the constraint is applied. The condition is stated as an XPath expression, which is evaluated with respect to each node, selected by the selector. If the result fom the evaluation is approving (a non-empty list of nodes, a non-empty string, the true Boolean value, or a positive number), then the constraint is applied; the event defines when this constraint is checked for application. Such events can be: selection of a menu item, pressing of key shortcut, some editing command as *enter a child* or *a parent* and similar; the action defines the way of the actual application of the constraint. There are three types of constraints, implemented in the system: *regular expression constraints*, *number restriction constraints*, *value restriction constraints*.

Macro Language

In the CLaRK System most of the tools support a mechanism for describing their settings. On the basis of these descriptions (called *queries*) a tool can be applied only by pointing to a certain description record. Each query contains the states of all settings and options which the corresponding tool has. In other words, each query has all the necessary information for applying the tool without any additional information or user interaction.

For user convenience and debugging purposes the queries themselves are represented in XML format. Within the system they can be treated like ordinary XML documents having their names and DTD assignments. For each kind of queries there is a special DTD included in the distribution package of the system.

Once having this kind of queries there is a special tool for combining and applying them in groups (macros). During application the queries are executed successively and the result from an application is an input for the next one. The final result is given by the last query application.

For a better control on the process of applying several queries in one we introduce several conditional operators. These operators can determine the next query for application depending on certain conditions. When a condition for such an operator is satisfied, the execution continues from a location defined in the operator. The mechanism for addressing queries is based on user defined labels. When a condition is not satisfied the operator is ignored and the process continues from the position following the operator. In this way constructions like **IF-THEN-ELSE** and **WHILE-DO** easily can be expressed.

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.

The system supports five types of control operators:

IF (XPath): the condition is an XPath expression which is evaluated on the current working document. If the result is a non-empty node-set, non-empty string, positive number or true boolean value the condition is satisfied;

IF NOT (XPath): the same kind of condition as the previous one but the approving result is negated;

IF CHANGED: the condition is satisfied if the preceding operation has changed the current working document or has produced a non-empty result document (depending on the operation);

IF NOT CHANGED: the condition is satisfied if either the previous operation did not change the working document or did not produce a non-empty result.

GOTO: unconditional changing the execution position.

Each macro defined in the system can have its own query and can be incorporated in another macro. In this way some limited form of subroutine can be implemented.
We will use the CLaRK tools within AsIsKnown Project in two tasks: (1) preprocessing of the trend corpus (articles from fashion magazines), including POS tagging, chunk grammars, semantic annotation; and (2) for searching of patterns for the text mining system.

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.

3 Conclusions and Outlook

In this report we present the elements of AIKOS (AsIsKnown Onto System) which implement the necessary services for integration of AIKOS within the AsIsKnown System.

Some of the functionalities are still general and in the process of integration of AIKOS in the whole system some modifications and additions are possible.

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.

References

[1] Simov K, Peev Z, Kouylekov M, Simov A, Dimitrov M, Kiryakov A. 2001. CLaRK - an XML-based System for Corpora Development. In: Proc. of the Corpus Linguistics 2001 Conference. Lancaster, England. pp: 558-560.

[2] Simov K, Kouylekov M, Simov A 2002 *Cascaded Regular Grammars over XML Documents*. In: Proc. of the 2nd Workshop on NLP and XML (NLPXML-2002), Taipei, Taiwan. September 1, 2002.

[3] Kiril Simov, Alexander Simov, Milen Kouylekov. 2003. *Constraints for Corpora Development and Validation*. In: Proc. of the Corpus Linguistics 2003 Conference, pages: 698-705.

[4] Kiril Simov, Alexander Simov, Hristo Ganev, Krasimira Ivanova, Ilko Grigorov. 2004. *The CLaRK System: XML-based Corpora Development System for Rapid Prototyping*. In: Proceedings of LREC 2004, Lisbon, Portugal. pages 235-238.

Information Society Technologies

AsIsKnown is a project funded within the Information Society Technologies (IST) Priority of the Sixth Framework Programme (FP6) of the European Commission.