Title: CODA: Computer-aided Ontology Development Architecture

Authors: Manuel Fiorelli, Maria Teresa Pazienza, Armando Stellato, Andrea Turbati

Abstract: This paper introduces CODA: Computer-aided Ontology Development Architecture. CODA is both an architecture and an associated framework supporting the transformation of unstructured and semi-structured content into RDF (Resource Description Framework) datasets. The purpose of CODA is to support the entire process embracing data extraction and transformation, identity resolution up to feeding semantic repositories with knowledge extracted from unstructured content. The motivation behind CODA lies in the large effort and design issues required for developing knowledge acquisition systems using content analytics frameworks such as UIMA (Unstructured Information Management Architecture) and GATE (General Architecture for Text Engineering). Therefore, CODA extends UIMA with facilities and a powerful language for projection and transformation of UIMA annotated content into RDF. The proposed platform is oriented towards a wide range of beneficiaries, from semantic applications developers to final users that can easily plug CODA components into compliant desktop tools. We describe and discuss the features of CODA through the article, and we conclude by reporting on the adoption of the CODA framework in the context of a usage scenario, related to knowledge acquisition in the agricultural domain.

Introduction

Nowadays, we are exposed to a huge volume of information from a plethora of channels, encoded in different formats and made available or accessible in more or less opaque ways. Efficient information management and, before that, information gathering, are becoming fundamental for exploiting this considerable flow of data. The Linked Data [1] paradigm allows to publish information in the form of interlinked datasets, while promoting reuse and ongoing integration across different providers towards unexpected applications. URIs (Uniform Resource Identifier) identify resources, and allow the retrieval of their RDF [2] descriptions. These descriptions consist in triples, which represent relationships between resources. Progressive download of RDF data associated with unknown resources enables a form of incremental data discovery. SPARQL (SPARQL Protocol and RDF Query Language) [3] supports ad-hoc querying of RDF datasets through pattern-matching. LOD (Linked Open Data) refers to Linked Data that are released under an open license [4].

However, much of the content on the web still does not conform to these uniform publication standards. Initiatives, such as the LOD Triplification Challenge [5] [6] ("triplification" indicates the process of transforming data into RDF triples), held from 2008 until present time in different editions, promoted the exposure of existing structured (relational) representations as Linked Data [7]. At the same time, there is also a considerable demand for systems and frameworks capable of managing unstructured content. Unfortunately, processing, transforming and manipulating heterogeneous information require coordinated capabilities of several dedicated tools. The success of semantic search engines such as Eqentia [8] or Evri [9], Information Extraction services such as OpenCalais [10] and Zemanta [11], and architectures and frameworks supporting text analytics and content extraction such as GATE [12] and UIMA [13], demonstrates the large interest for unstructured information management.

The search engines and the extraction services mentioned above provide facilities for producing knowledge modeled according to open standards. However, users may not extend them, e.g. by providing new content extractors and annotators. On the other hand, platforms such as GATE and UIMA support the development of content analytics, while do not provide any specific help for data transformation and publication, delegating these tasks to developers. This is surely a gap which needs to be filled with dedicated solutions, as modeling principles for the Web of Data [14] have today reached a high maturity, and non-trivial aspects associated with knowledge elicitation – such as identity of resources [15] and its resolution [16] – have been widely discussed in literature.

In this article we introduce a novel systematic approach to knowledge acquisition, which aims to overcome the above limitations. It is completely based on standard technologies (such as UIMA for Unstructured Information Management) and on models originated from the *semantic web* [17] community, while offering an open architecture and platform for driving systems in the entire process of semantic content production, ranging from content analytics, information extraction and RDF triplification. While reusing UIMA for orchestrating the analysis of the unstructured content, our architecture supports the transformation of the analysis results into semantic content, which is compatible with the semantic web ecosystem.

Background, Motivation And Objectives

Knowledge acquisition encompasses several tasks, which are often addressed in different research areas, leading to independent approaches and technologies. Until now, architectural definitions and interaction modalities have been defined in detail, reaching industry-standard level in some cases, for processes such as *ontology development* and *text analysis*. Most recent ontology development tools follow the path laid by Protégé [18], while TIPSTER [19] is the reference architecture for text processing, which has been implemented by GATE and influenced the recently approved OASIS (Organization for the Advancement of Structured Information Standards) standard UIMA [20].

However, development of knowledge acquisition systems today largely requires non-trivial integration effort and the development of ad hoc solutions for tasks which could be better defined and channeled into an organic approach. Until now, ontology development driven by knowledge from external resources lacks a comprehensive study and the synthesis of an architecture. Results in the field of *ontology learning* [21] fulfilled this goal to different extents: OntoLearn [22] provides a methodology, algorithms and a system for performing different ontology learning tasks, OntoLT [23] is a Protégé plugin for adding new ontology resources extracted from text, while only Text2Onto [24] embodies a first attempt to realize an open architecture for management of (semi)automatic ontology evolution processes.

A few attempts at empowering information extraction architectures with data projection mechanisms are represented by the RDF UIMA CAS (Common Analysis Structure) Consumer [25], Clerezza-UIMA integration [26], and Apache Stanbol [27]. The first two actually were targeting the same objective: integrating UIMA and RDF Clerezza technologies, and just started from inside their respective communities. While the first two approaches support the RDF serialization of UIMA analysis metadata, the third one provides a modular software stack and reusable set of components for semantic content management. However, the scope of all three approaches is limited to performing a mere vocabulary-agnostic RDF serialization of the CAS, not including any machinery for projecting information towards specific RDF vocabularies, which reflect different ontological and modelling decisions, and possibly adopted by existing data. More recently, Stanbol has addressed this aspect, through a specific component called Refactor Engine [28]. The Refactor Engine performs RDF graphs transformations to specific target vocabularies or ontologies by executing rules expressed in a dedicated syntax, which is then grounded over SPARQL CONSTRUCT.

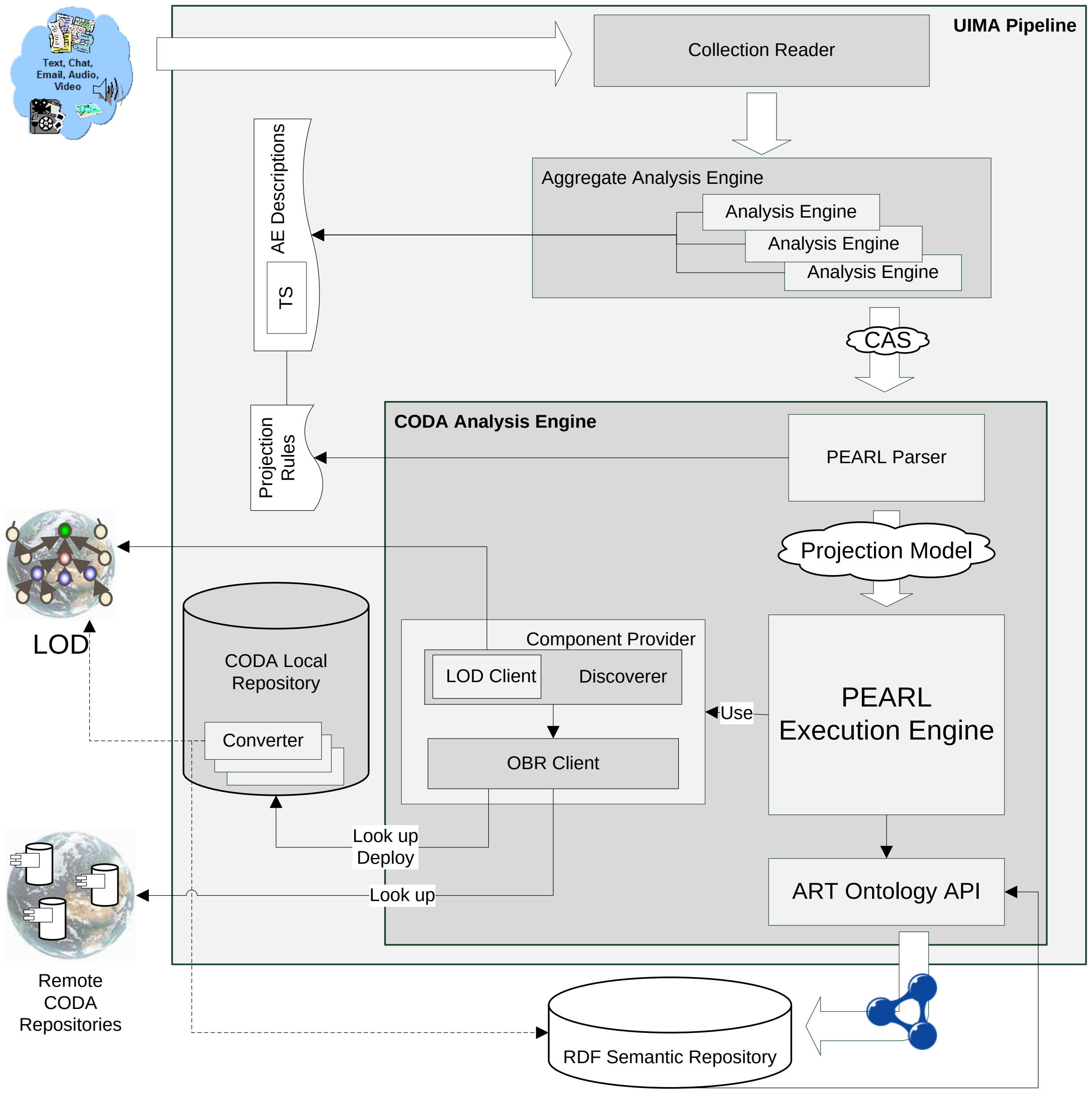

Figure 1. CODA Architecture. (AE: Analysis Engine, API: Application Programming Interface, CAS: Common Analysis Structure, CODA: Computer-aided Ontology Development Architecture, LOD: Linked Open Data, OBR: OSGi Bundle Repository, PEARL: ProjEction of Annotations Rule Language, RDF: Resource Description Framework, TS: Type System)

What is lacking in all current approaches is an overall perspective on the task and a proposal for an overall architecture driving and supporting the entire flow of information, from data acquisition from external resources to their exploitation for RDF triplification. In this context, we propose CODA [29] (Computer-aided Ontology Development Architecture), an architecture which aims at completing the unstructured content analysis capabilities of frameworks such as UIMA and GATE, with a coordinated environment supporting the processing, transformation and projection of produced metadata into RDF Semantic Repositories.

Architecture

CODA extends UIMA with specific capabilities for populating RDF datasets and enriching ontologies with information mined from unstructured content. This section is organized as follows. We begin with background information about the UIMA architecture, aimed to explain its interfacing with CODA. Then, we introduce the components shown in Figure 1 and detail their role in the CODA architecture.

UIMA: Unstructured Information Management Architecture

UIMA offers an architecture (OASIS Standard in 2009) and an associated framework (released by IBM as open source in 2004, then donated in 2006 to the Apache Software Foundation [30]), supporting development and orchestration of analysis components – called Analysis Engines (AE) – for the extraction of information from unstructured content. UIMA is not limited, though naturally bound, to *text analytics*, and can be potentially adopted for processing any kind of media.

In UIMA, a CAS (Common Analysis Structure) [31] is the primary data structure which Analysis Engines use to represent and share analysis results. A CAS contains both the information to be analyzed (called Subject of Analysis, or *Sofa*) and the extracted metadata, represented as typed feature structures [32] constrained by a model called *Type System* (TS) in the UIMA terminology. A specific kind of metadata (*annotations*) contains references to the region of the Sofa from which it was extracted. A total order is defined over annotations, usually reflecting the physical position of the referenced regions within the Sofa. A metadata index is maintained inside the CAS for iterating over annotations in a specified order.

Usually, a UIMA workflow consists in a simple pipeline whose last element is an Analysis Engine that consumes the extracted metadata for some purpose (e.g. to augment a structured source of information). Being focused on a general architecture, UIMA does not provide any implementation targeting a specific source, such as relational databases or RDF knowledge bases.

In Figure 1, we depict the architecture of a typical application based on CODA. Our primary contribution with respect to the standard UIMA pipeline is the introduction of a concrete Analysis Engine (*CODA Analysis Engine*) targeting the evolution of RDF datasets (*Target RDF Dataset* in the figure) where RDF datasets consist of both knowledge bases (data) and ontologies (schemes). The Analysis Engine delegates domain specific tasks to external components which can be automatically discovered on the Web and deployed into a local repository (*CODA Local Repository*).

CODA Analysis Engine

CODA provides dedicated components and their orchestration for projecting UIMA analysis results into the Target RDF Dataset, while exploiting UIMA for the analysis of unstructured information. The main task of CODA is to project UIMA metadata contained in a CAS into user defined RDF graph patterns based on a vocabulary of choice. In performing this task, CODA integrates the fully fledged API (Application Programming Interface) provided by UIMA for accessing the subject of analysis with its range of specific functionalities for transforming the metadata. Actually, any UIMA pipeline can be extended with the Analysis Engine wrapping CODA, so that at the end of the processing the metadata extracted so far can be projected to RDF. Additionally, existing UIMA-powered applications may deal with CODA in a uniform manner, since deployment, management and execution of UIMA components follow known conventions.

The projection of UIMA metadata to RDF triples is a complex process including the following activities:

- Select relevant information from within the CAS;
- Turn UIMA metadata into RDF nodes;
- Build an RDF graph from a user defined graph pattern and the prepared RDF nodes.

For supporting the user in the specification of this process, CODA provides a rule-based pattern matching and transformation language, called PEARL (see next section for more details about its syntax and semantics).

The *PEARL Execution Engine* is the component implementing the PEARL operational semantics. This engine orchestrates the triplification process, executing PEARL *projection rules* against each CAS produced by the UIMA workflow. These rules specify in a declarative way how metadata conforming to a given type system (*TS*) are related to RDF triple patterns using a given vocabulary. PEARL syntax covers UIMA metadata selection and RDF graph construction patterns. The construction of RDF nodes out of UIMA metadata is a highly varied task, possibly depending on domain and application specific procedures for composing URIs and literals, or on external services for identity resolution (such as OKKAM [16]). In order to preserve its simplicity, the PEARL syntax provides specific extension points for delegating this task to external components, called *converters*.

A dedicated rule set is sufficient to handle most usage scenarios, without any coding activity in a general-purpose programming language, such as Java. Very specific tasks can be accomplished through converters implemented in Java using the API provided by CODA. Still, most of the transformation may be specified in PEARL using transformation rules that, in turn, delegate some of the work to the converters. Actually, converters are indirectly referred by means of a URI (see section on *Language Overview*) which identifies the desired behavior (a *contract*) instead of a concrete implementation. In a certain sense a contract identifies a set of functionally equivalent converters, possibly differing for non-functional properties, such as resource consumption, performance with respect to the task, or even licensing terms. The exact behavior represented by a contract is expressed in terms of specifications in natural language (as usual in contracts for service or methods), provided through standard metadata properties (e.g. *dcterms: description*) and it is actually the URI of the contract which provides the sole semantic anchor. For instance, the contract <*http://art.uniroma2.it/coda/contracts/default*> is described as “the procedure invoked by default for transforming a UIMA value into a valid RDF term”. In fact, the input/output behavior of contracts is difficult to express in a formal language. Therefore, we are unable to exploit approaches for matching semantic web services [33], based on annotations with respect to ontologies and formal preconditions and postconditions. However, approaches for discovering (semantic) web services through natural language descriptions might be extended to CODA contracts, as well. Influenced by the web services architecture [34], the distinction between contracts and converters helps the reuse of projection specifications, since an application can bind the contracts to the converters best fitting its own requirements.

The PEARL execution logic is kept separate from the contract resolution process as contract references are resolved into suitable converters by the *Component Provider*. Converters are bundled complying with the OSGi (formerly known as the Open Services Gateway initiative) specification [35] and stored in repositories organized according to the OBR (*OSGi Bundle Repository*) architecture [36]. OBR repositories maintain metadata about the hosted bundles including their name, version, provided capabilities and requirements. A CODA converter is advertised on OBR repositories as a *service* capability, which holds the contract URI, and the Java interface for interrogating the converter. OBR requirements are instead populated with the non-functional properties of the converters.

The Component Provider follows a two-step procedure for resolving contracts into suitable converters. At first it uses the *OBR Client* to access a known set of OBR repositories (starting from the *CODA Local Repository*) looking for a candidate whose metadata match the contract. If no candidate is found, the Component Provider relies on its *Discoverer* module to explore the Web looking for additional repositories. The Discoverer exploits the fact that PEARL specifications are self-describing [37] and, moreover, grounded in the Web of Data, since the required contracts are mentioned through dereferenceable URIs [38]. In compliance with the Linked Data principles, the Discoverer obtains through those URIs an RDF description of the contract, including a list of authoritative repositories of known implementations. This architecture enables autonomous configuration of CODA systems, disburdening the user from manual settings prior to the execution of a PEARL projection. This is especially valuable when reusing PEARL documents written by third parties, as in the open and distributed scenario described in [39].

The execution of a PEARL specification against a given CAS produces a set of triples which are eventually committed to an RDF triple store. Also, both the core PEARL syntax and the converters may refer to the underlying Target RDF Dataset, for closing a feedback loop around the Target RDF Dataset. The interaction with a triple store is mediated by the OWL ART API [40], an open RDF middleware providing implementations for different triple store technologies.

PEARL: The Projection Language

PEARL (ProjEction of Annotations Rule Language) is a pattern-matching and transformation language for the triplification of UIMA metadata. Being specifically tailored to this purpose, the language allows concise yet powerful specifications of the transformation. An initial specification of PEARL was presented in [41]. This article has a wider scope, covering CODA as a whole and thus including the PEARL language. While we refer the reader to the past publication for a thorough reading about PEARL and for more examples of its usage, we provide here an overall description and discuss features which have been introduced since then. Another novel element is a detailed comparison of PEARL with alternative approaches, in order to show the advantages of our solution. Finally, we consolidate the soundness of our proposal, by relating it to similar approaches adopted for solving analogous problems.

Language Overview

A PEARL Document mainly consists of rules driving the UIMA Metadata Projection process. A rule is introduced by the keyword *rule* (optionally preceded by the keyword *lazy*), followed by the matched UIMA type and, optionally, by an identifier and a set of dependencies on other rules. A non-lazy rule (a rule not introduced by the keyword “lazy”) is invoked for each annotation from the index that matches the UIMA type specified in its declaration. Due to their triggering condition, such rules are also called *matched rules*. A *lazy rule* is executed only if it is invoked explicitly by a matched rule through a binding in its *bindings* section. Lazy rules are syntactically constrained, as they can only define a *nodes* section for the generation of RDF terms out of the extracted information. Therefore, matched rules may be decomposed though bindings to lazy rules, thus fostering the modularization of a PEARL rule-set.

The *nodes* section (see Figure 2 for an example of a PEARL rule) uses a *feature path* (a sequence of features inside a feature structure) to select relevant UIMA metadata from the matched annotation which will be converted to RDF nodes. These generated nodes are assigned to *placeholders*. For instance, in

```
prefix xsd:    <http://www.w3.org/2001/XMLSchema#>
prefix foaf:   <http://xmlns.com/foaf/0.1/>
prefix bibo:   <http://purl.org/ontology/bibo/>
prefix cdbk:   <http://art.uniroma2.it/book/coda#>

rule it.uniroma2.Book id:bookRule  {
   nodes = {
      bookISBN      uri(cdbk:isbn)        isbn
      bookAuthor    uri                   author
      bookTitle     literal               title
      authorName    literal^^xsd:string   author
   }

   graph = {
      $bookISBN                a                 bibio:Book     .
      $bookISBN                bibo:title        $bookTitle     .
      OPTIONAL {   ?bookAuthor  foaf:name        $authorName .       }
      OPTIONAL {   $bookISBN    bibo:author      ?bookAuthor   .     }
   }

   where = {
      ?bookAuthor              foaf:name         $authorName .
   }
}
```

Figure 2. A PEARL rule example.

the rule in Figure 2, the UIMA feature it.uniroma2.Book:author is projected both as an RDF named resource (URI) and as a string typed literal. In the first case a URI (bookAuthor) is created after the feature content, and will be used as the resource identifying the extracted author, while in the second case a literal value (authorName) is generated to populate a property of the author (as specified in the graph section of the rule). Default conversion heuristics are applied, and are inferred on the basis of the specified node type. For instance, if the type of the target is URI, the feature value is first "cleaned", to remove characters which are incompatible with the URI standard, and then used as a local name and concatenated to the namespace of the target ontology to create an URI. Optionally, dedicated *converters* (see end of the section above) may be declared to operate the transformation. In the example, a URI is generated from the extracted ISBN (International Standard Book Number), following the ISBN-A specification [42], by invoking a converter compliant to the cdbk:isbn contract, and is assigned to the bookISBN placeholder. An optional section, similar to *nodes*, and called *bindings*, allows to declare complex placeholders as objects referencing placeholders declared in lazy rules (an example on the use of bindings and lazy rules is provided later in this article).

The *graph* section defines a graph pattern [43] that is used as a template for the instantiation of RDF triples. The graph pattern is dynamically instantiated with RDF nodes generated in the *nodes* and *where* sections. Inside a graph pattern, placeholders originated in the *nodes* section are identified through the prefixed symbol "$", while *variables* grounded in the *where* section, are prefixed with the symbol "?". When a placeholder and a variable have the same name, they could be related through the fallback mechanism described in section "Operation Semantics".

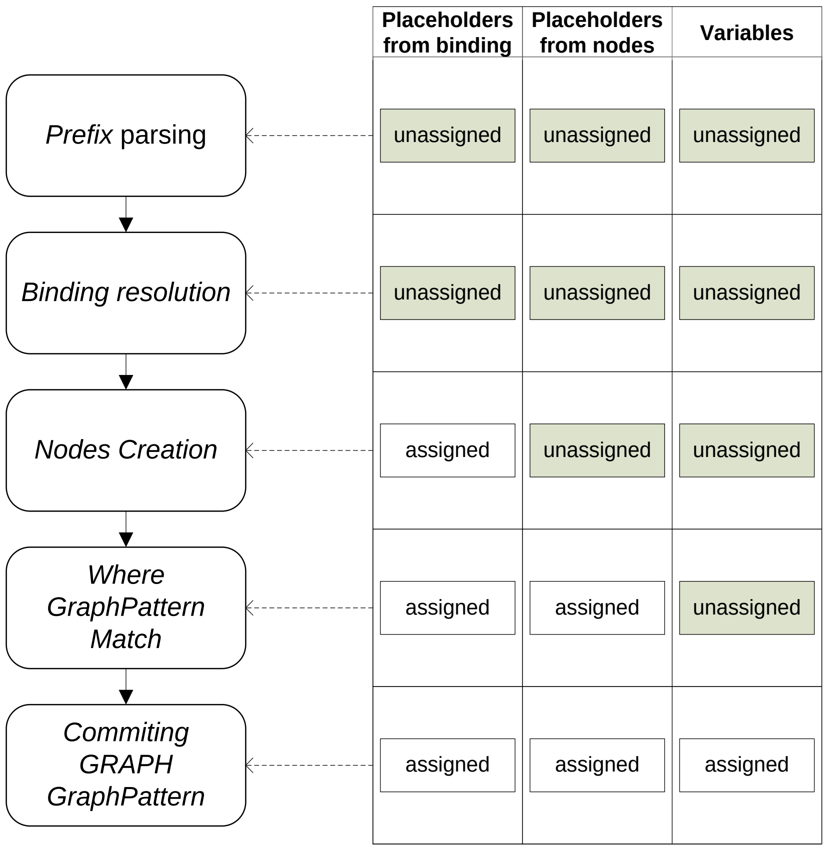

Figure 3. PEARL execution flow.

The *where* section contains a graph pattern, which is matched against pre-existing information in the Target RDF Dataset, for binding variables to matching RDF nodes by means of unification. The purpose of this section is to link newly extracted data with information which is already present in the target ontology. In the example, the graph pattern in the *where* section is composed of a single triple characterized by a variable in the subject, the property foaf:name as the predicate, and the placeholder authorName as the object. The placeholder has been initialized with a string typed literal in the nodes section, and is thus used in combination with the ground predicate to retrieve a substitution for the variable author from the Target RDF Dataset.

The key distinction between placeholders and variables lies in how they are initialized: the former with RDF nodes generated from UIMA annotations (*nodes* section), the latter with existing resources retrieved from the target semantic repository through pattern matching (*where* section).

Operational Semantics

The *Pearl Execution Engine* iterates over the UIMA annotations according to the order provided by the default annotation index. For each annotation, all matched rules that apply to the annotation type are executed to produce RDF triples. These triples will be committed after a validation phase. The execution flow of a matched rule is shown in Figure 3: each row of the table represents the assignment status of placeholders and variables at the start of each phase. The execution starts with a pre-processing routine, by parsing and replacing all prefixes in the document with their assigned namespaces. The *bindings* section, if present, contains a collection of bindings, each one instructing the engine to execute a lazy rule against the current UIMA annotation. Such a rule generates RDF nodes and assigns them to placeholders, which can be referenced within the caller rule through the binding symbolic name. Later, in section “A Use Case: Evolving AGROVOC”, we provide an example combining lazy rules and bindings.

Execution then passes to the *nodes* section by initially ensuring that all referenced *converters* are deployed in the CODA Local Repository (or, in negative case, by retrieving them from the Web, as described in section on *CODA Analysis Engine*). After all the machinery is ready for the transformation of UIMA metadata, RDF nodes are generated and assigned to *placeholders*.

Ground placeholders are then forwarded to the next step that is the resolution of the *where* section, in which *variables* are assigned values through unification with the Target RDF Dataset. The application of both graph patterns from the *where* and *graph* sections may *fail*. In the first case, the *where* graph is matched against the Target RDF Dataset in much the same way of a SPARQL *where* condition. In PEARL, a failed match is always considered as having a result-set consisting of a single tuple with all variables left unbound, thus the rule is not considered to be *failed*, its execution continues, but the variables declared in the *where* are not bound.

The application of the graph pattern in the *graph* section is different: the graph does not need to be matched against the Target RDF Dataset, but instead to be stored into it. In this case, satisfying the graph equals to satisfying the set of all *write operation*s. A *write operation* of a graph pattern GP into a graph G succeeds if all the three elements (subject, predicate, object) of the triples in GP are bound (instantiated). The OPTIONAL modifier here has slightly different semantics from the one in SPARQL (and thus in the PEARL *where* section): writing of the whole graph pattern does not fail if the subgraph inside an OPTIONAL clause fails to be written (at least one of its triples is not completely instantiated), and the triples of this subgraph are simply left out from the output. If the application of a graph pattern in the *graph* section fails, no triple is produced for that rule application.

In the example in Figure 2, the placeholder $bookAuthor is never used in the rule, though it is being created inside the *nodes* section. This brings in another feature of the variable assignment process: whenever a variable in the *graph* pattern is not bound, before declaring that write to be *failed*, the processor tries to match it with a placeholder of identical name. In the given example, the variable ?bookAuthor is used two times: the first time as the object of the property bibo:author and the second time as the subject of the property foaf:name. The variable is bound to a match in the target ontology through its presence in the graph pattern of the *where* section, which looks for already existing authors with the same name. If the triple in the *where* graph is a match, then the variable is bound and initialized with the URI already assigned to the author with that name. If the match fails, the variable is not bound, but the value of the placeholder with the same name – that is the one initialized by converting the it.uniroma2.Book:author feature – will be used in place. The idea is that, if the author already exists in the target ontology, then its URI is retrieved through the *where* pattern, otherwise a URI is generated for it. In other words, the execution of the *where* section is a solution for the identity resolution problem. Furthermore, the same objective can be achieved by using a converter, especially when sophisticated application or domain specific procedures are needed. The selection among different constructs for the same task should be driven by the user capabilities and needs.

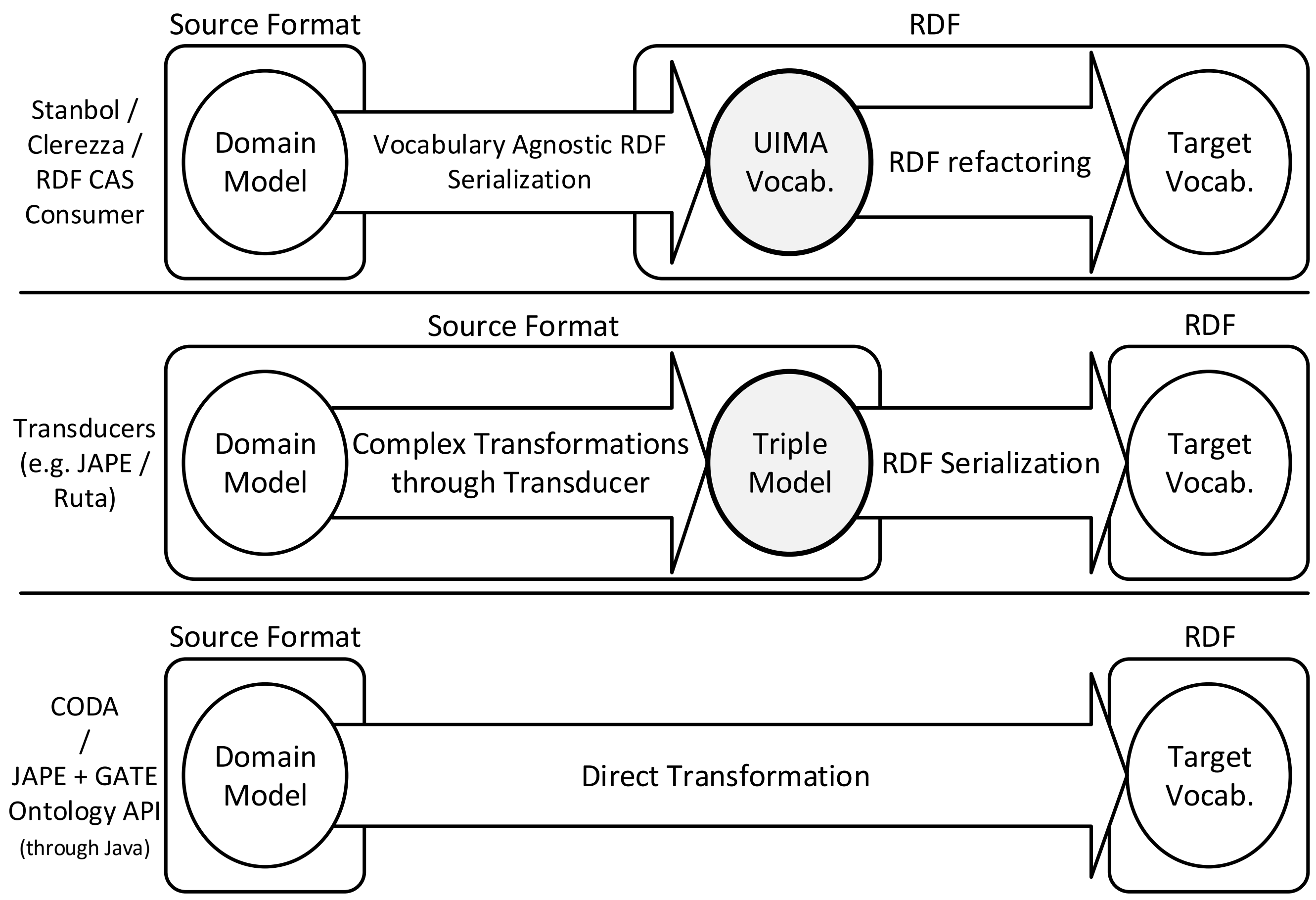

Figure 4. Comparison of integration approaches.

The execution of lazy rules is limited to the initialization of placeholders, since such rules may not define other sections. Moreover, they follow a different invocation strategy, based on *bindings* declared inside other matched rules.

PEARL offers many other language features (e.g. rule dependencies, UIMA list management, conditional assignments etc...), widely documented in [41], which we do not report here due to space limitation.

A Comparison of PEARL with Other Existing Approaches

While discussing background and motivation, we have mentioned existing systems and tools aimed at triplification of data extracted from unstructured content. We analyze here the positioning of PEARL with respect to them. Figure 4 presents a categorization of the main approaches. The first row represents those systems that initially perform a vocabulary agnostic triplification of extracted data, then reduce the generation of vocabulary specific triples to an RDF refactoring problem. The second row presents a specular approach, in which extracted data are transformed by means of rule languages into an intermediate model, which can be straightforwardly projected as RDF triples. The approach in the third row introduces a single step conversion, which necessarily requires a dedicated language, combining capabilities for querying and manipulating both the source format and RDF.

RDF UIMA CAS Consumer, Clerezza-UIMA integration, and Apache Stanbol all fall in the first category. The advantage in adopting such an approach lies exactly in the early migration to RDF, which allows to reuse existing languages and technologies (such as SPARQL-CONSTRUCT and SWRL) for projecting source data towards the target vocabulary. On the other side, these languages are not optimized for the specific task, thus implementing the required transformations is not immediate. A lossless serialization of a CAS into RDF clearly depends on some vocabulary for representing [44, 45, 46] the references to (portions of) the subject of analysis.

The second approach can be actually implemented through transducing tools existing for most architectures, such as JAPE (Java Annotation Patterns Engine) [47] for GATE or Ruta (formerly, TextMarker [48]) for UIMA. These tools allows easy processing and conversion of data in the source format (e.g. a Type System in UIMA), which can then be fed to an intermediate model (in the same format) for which a straightforward projection to RDF is defined once for all. The drawback of this approach lies in the inaccessibility of the Target RDF Dataset. In GATE, they overcame this limitation by introducing an ontology access layer that can be used in the right hand side part of JAPE rules. The combination of JAPE and Ontology Access Layer is an example of the third approach. However, this solution lacks the elegance and conciseness of a dedicated language, as the ontology layer is invoked by means of Java code (as JAPE allows for Java code to be used in the right hand side part of a rule).

PEARL falls too in the third category, though with respect to the above case, provides a dedicated declarative language which combines *feature path* expressions for querying UIMA data, with SPARQL-like expressions for accessing the ontology. The possibility to use SPARQL not only for generating RDF triples, but also for querying the dataset to be populated, allows the refinement of the output according to feedback gathered from already existing – and assessed – triples (see end of previous section, or [41] for reading about further feedback-based possibilities).

Approaches combining different languages to support transformation between their respective domains are not new in literature. In the context of Semantic Web Services, where there is a need for transforming RDF content into XML-based (eXtensible Markup Language) [49] languages for Web Services, the XSPARQL [50] proposal combines SPARQL RDF querying capabilities with the XML manipulation of XQUERY [51]. In the context of data lifting for the Semantic Web, DM (Direct Mapping) [52] allows to view relational databases as RDF datasets, whose structure and vocabulary are derived systematically from the input relational schemas. A companion specification, R2RML (RDB to RDF Mapping Language) [53], allows to personalize the mapping, while leveraging SQL (Structured Query Language) for constructing views over the input databases. These two specifications roughly correspond to the approaches represented in Figure 4, in the first and third row, respectively. Facelets [54] provides a view-declaration language for enterprise web applications by exploiting the extensibility of XML to embed its own tags inside standard XHTML web pages, which can thus be easily shared between web designers and developers. By considering how the above manifestations of this task have been addressed in different areas, we can gather further evidences supporting the design of PEARL.

Meta-model type systems [55] and to a lesser extent the `Referent` type in the OASIS specification support the use of ontological resources in the analysis metadata. Providing that the focus of these metadata is in deep-semantics, then it should be possible to derive systematically their RDF representation. Nonetheless, RDF refactoring could be still necessary, unless the analysis engine are aware of, or even coupled to, the target dataset and vocabularies.

A Use Case: Evolving AGROVOC

CODA has been successfully applied to some real life scenarios, regarding different tasks, such as ontology population, ontology learning and semantic indexing. We describe here one of those scenarios, discussing the benefits provided by CODA in enriching the AGROVOC [56] thesaurus with new content extracted from text. AGROVOC [57] is a thesaurus developed and maintained by The Food and Agriculture Organization (FAO, [58]) of the United Nations (UN). It contains more than 32000 concepts in up to 25 languages covering topics related to food, nutrition, agriculture, fisheries, forestry, environment and other related domains. In 2010, the vocabulary has been migrated to SKOS (Simple Knowledge Organization System) [59] and published as Linked Data. In compliance with the fourth Linked Data rule, reliable mappings [60] to other popular datasets have been produced by means of semi-automatic processes.

AGROVOC contains, among others, a number of concepts regarding plants, insects and, to a minor extent, pesticides. However, recent measurements performed by AGROVOC maintainers have highlighted a low coverage of semantic relations among them, therefore motivating a massive introduction of new relation instances.

In the context of a collaboration between our research group and OEKC (Office of Knowledge Exchange and Capacity Development) group of UN FAO, we have developed a platform for the semi-automatic augmentation of AGROVOC, based on the CODA framework. This platform, called *AgroIE* [61], has been designed by considering the following conditions:

- the target relations are known in advance (an insect *is a pest of* a plant, a substance *is a pesticide of* an insect, a substance *is a herbicide of* a plant);
- there is no ready to use labelled corpus for the task;
- the unlabelled documents come from sources (e.g. Wikipedia) which tend to use a formal and repetitive language;
- the system must provide results of medium-high quality, otherwise the work of human validators is too cumbersome to be accomplished.

Given these conditions, we chose to use a small set of high precision extraction patterns, manually identified through experimentation over a hundred documents from the available sources.
AgroIE uses a UIMA pipeline whose components are in charge of linguistic analysis and lexico-syntactic pattern matching. Our lexico-syntactic patterns rely on numerous linguistic and semantic features, including tokens, lemmas, constituents and named entities. For their computation, we use the analysis engines from DKPro [62] wrapping Stanford Core NLP (Natural Language Processing) [63]. The relevant entities (insects, plants and pesticides) are recognized using an internally developed NER (Named Entity Recognizer), based on AGROVOC and NAL (National Agricultural Library) [64].

Figure 5 offers an excerpt of the PEARL document managing the projection of UIMA metadata into RDF triples for the AGROVOC dataset. For the sake of clarity a more human readable name, `agrovoc:insects`, is used instead of the AGROVOC true identifier, which is `agrovoc:c_11792`. The same applies to the name `agrovoc:plants` which is used instead of `agrovoc:c_5993`. The graph section lists the triples to be produced for creating new SKOS concepts describing insects and plants, their lexicalization in the SKOS-XL extension of SKOS, and the definition of the relationship, based on the vocabulary [65] adopted by AGROVOC. In the example, two *lazy* rules do not contain any graph pattern for triple projection and provide instead the mere generation of RDF nodes from the metadata of their matched UIMA types. These rules adopt two dedicated converters which act as *identity resolution* modules: they

```
prefix skos:        <http://www.w3.org/2004/02/skos/core#>
prefix skosxl:      <http://www.w3.org/2008/05/skos-xl#>
prefix agrovoc:     <http://aims.fao.org/aos/agrovoc/>
prefix agronto:     <http://aims.fao.org/aos/agrontology#>
prefix agroIE:      <http://art.uniroma2.it/agroie/coda#>

rule agroie.type.IsPestOf  id:IsPestOfRule  {
    bindings = {
        plant           plantList           id:PlantRule
        insect          insectList          id:InsectRule
    }
    graph = {
        $insect.insectId        a                       skos:Concept            .
        $insect.insectId        skos:broader            agrovoc:insects         .
        $insect.insectId        skosxl:label            _:ixlbl                 .
        _:ixlbl                 skosxl:literalForm      $insect.insectName      .
        $plant.plantId          a                       skos:Concept            .
        $plant.plantId          skos:broader            agrovoc:plants          .
        $plant.plantId          skosxl:label            _:pxlbl                 .
        _:pxlbl                 skosxl:literalForm      $plant.plantName        .
        $insect.insectId        agronto:pestOf          $plant.plantId          .
    }
}

lazy rule agroie.type.Insect id:InsectRule {
    nodes = {
        insectId        uri(agroIE:insectConverter)    insectName
        insectName      literal@en                     insectName
    }
}

lazy rule agroie.type.Plant id:PlantRule {
    nodes = {
        plantId         uri(agroIE:plantConverter)     plantName
        plantName       literal@en                     plantName
    }
}
```

Figure 5. PEARL document example.

look up the name of entities in both AGROVOC and NAL (preferring identifiers from the former dataset, if available), and generate a new URI for AGROVOC in case a new entity (insect or plant) is found. This example shows how the main aspects of the transformation, which are intrinsically declarative, may be easily represented as a series of PEARL rules, instead of being hard-wired in the business logic of the UIMA Analyzers, which are more opaque, more difficult to be modified, and thus harder to maintain. Also, the classification of functionalities into clearly separated, web-dereferenceable and auto-deployable components boosts the portability of the system.

This experience showed how CODA facilitates the adoption of a human-scalable, distributed development workflow: different team members may focus on the aspects they are more proficient with, interacting with other members on the basis of a high-level clear plan. In our setting, two MsC students have been developing – in the context of their thesis work – the UIMA Information Extraction engine, a PhD student with expertise in CODA has written the PEARL document once the UIMA type system had been defined, while a fourth developer familiar with linked data has written the two *converters and identity resolvers*. The clear separation between content extraction and triplification issues and the adoption of common interfaces to share enabled parallel development and thus increased productivity.

Conclusion And Future Work

CODA establishes a process and architecture for the triplification of unstructured content by integrating state-of-the-art solutions for unstructured information management (UIMA) and for knowledge representation (the family of RDF languages). As shown in the previous section, the proposed architecture and its associated framework support the agile development of systems for Knowledge Acquisition through a clear separation of concerns among concurrently working teams, whose synchronization is limited to the definition of shared interfaces.

Currently, CODA focuses on processing information extracted by UIMA analyzers for producing RDF content, therefore its activation naturally takes place at the end of a UIMA Processing Pipeline. This approach totally decouples CODA from UIMA processors, and surely encourages reuse of independently built AEs. However, it is advisable that CODA provides specific support to the development of “Ontology-Aware” analyzers, which are somehow influenced by the already assessed knowledge in the target repository. This approach should raise the relevance of the analysis results, and simplify their successive triplification.

A further interesting step towards the envisioned *Computer-aided* Ontology Development – which we are currently investigating – lies in the definition of modalities for effective and productive interaction with human users for semi-supervised evolution of ontologies and knowledge bases. Human intelligence could improve the flexibility of the overall process, and enhance the quality of the produced semantic content. Human involvement would be viable only if dedicated support tools reduced the cognitive load and the effort associated with manual activities.

Acknowledgments

This research has been partially supported by the EU funded project SemaGrow (Grant agreement no: 318497).

References

[1] C. Bizer, T. Heath and T. Berners-Lee, "Linked Data - The Story So Far," *International Journal on Semantic Web and Information Systems (IJSWIS), Special Issue on Linked Data*, vol. 5, no. 3, pp. 1-22, 2009.

[2] F. Manola and E. Miller, "RDF Primer," World Wide Web Consortium (W3C), 10 February 2004. [Online]. Available: http://www.w3.org/TR/rdf-primer/. [Accessed 21 November 2013].

[3] W3C SPARQL Working Group, "SPARQL 1.1 Overview," World Wide Web Consortium (W3C), 21 March 2013. [Online]. Available: http://www.w3.org/TR/sparql11-overview/. [Accessed 25 November 2013].

[4] Open Knowledge Foundation (OKFN), "Open Definition website," [Online]. Available: http://opendefinition.org/. [Accessed 25 November 2013].

[5] "LOD Triplification Challenge," [Online]. Available: http://triplify.org/Challenge/.

[6] "I-Challenge in I-Semantics 2013," [Online]. Available: http://i-semantics.tugraz.at/i-challenge.

[7] S. Auer, S. Dietzold, J. Lehmann, S. Hellmann and D. Aumueller, "Triplify: light-weight linked data publication from relational databases," in *Proceedings of the 18th international conference on World wide web*, New York, NY, USA, 2009.

[8] "Equentia official website," [Online]. Available: http://www.eqentia.com/.

[9] "Evri official website," [Online]. Available: http://www.evri.com/.

[10] "OpenCalais official website," [Online]. Available: http://www.opencalais.com/.

[11] "Zemanta official website," [Online]. Available: http://www.zemanta.com/.

[12] H. Cunningham, "GATE, a General Architecture for Text Engineering," *Computers and the Humanities,* vol. 36, pp. 223-254, 2002.

[13] D. Ferrucci and A. Lally, "Uima: an architectural approach to unstructured information processing in the corporate research environment," *Nat. Lang. Eng.,* vol. 10, no. 3-4, pp. 327-348, 2004.

[14] "Definition of Web of Data from W3C Linked Data Glossary," [Online]. Available: http://www.w3.org/TR/2013/NOTE-ld-glossary-20130627/#web-of-data.

[15] A. Gangemi and V. Presutti, "Identity of Resources and Entities on the Web," *Int. J. Semantic Web Inf. Syst.,* vol. 4, no. 2, pp. 49-72, 2008.

[16] P. Bouquet, H. Stoermer and B. Bazzanella, "An Entity Naming System for the Semantic Web," in *In Proceedings of the 5th European Semantic Web Conference (ESWC 2008),* 2008.

[17] T. Berners-Lee, J. A. Hendler and O. Lassila, "The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities," *Scientific American,* vol. 279, no. 5, pp. 34-43, 2001.

[18] J. Gennari, M. Musen, R. Fergerson, W. Grosso, M. Crubézy, H. Eriksson, N. Noy and S. Tu, "The evolution of Protégé-2000: An environment for knowledge-based systems development,," *International Journal of Human-Computer Studies,* vol. 58, no. 1, p. 89–123, 2003.

[19] D. Harman, "The DARPA TIPSTER project," *SIGIR Forum,* vol. 26, no. 2, pp. 26-28, 1992.

[20] "Unstructured Information Management Architecture (UIMA) Version 1.0," OASIS Standard, 2 March 2009. [Online]. Available: http://docs.oasis-open.org/uima/v1.0/uima-v1.0.html.

[21] P. Cimiano, Ontology Learning and Population from Text Algorithms, Evaluation and Applications, vol. XXVIII, Springer, 2006, p. 347.

[22] P. Velardi, R. Navigli, A. Cucchiarelli and F. Neri, "Evaluation of ontolearn, a methodology for automatic population of domain ontologie," in *Ontology Learning from Text: Methods, Applications and Evaluation*, IOS Press, 2005.

[23] P. Buitelaar, D. Olejnik and M. Sintek, "A Protégé Plug-In for Ontology Extraction from Text Based on Linguistic Analysis," in *Proceedings of the 1st European Semantic Web Symposium (ESWS)*, Heraklion, Greece, 2004.

[24] P. Cimiano and J. Völker, "Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery," in *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems,* Alicante, 2005.

[25] "Apache UIMA RDF CAS Consumer documentation," [Online]. Available: http://uima.apache.org/downloads/sandbox/RDF_CC/RDFCASConsumerUserGuide.html.

[26] "Clerezza integration with Apache UIMA," [Online]. Available: http://incubator.apache.org/clerezza/clerezza-uima/.

[27] "Apache Stanbol official website," [Online]. Available: http://stanbol.apache.org/.

[28] "Official Refactor webpage in the Apache Stanbol website," [Online]. Available: http://stanbol.apache.org/docs/trunk/components/rules/refactor.html.

[29] "Official CODA website," [Online]. Available: http://art.uniroma2.it/coda/.

[30] "Official UIMA website," [Online]. Available: http://uima.apache.org.

[31] T. Götz and O. Suhre, "Design and implementation of the UIMA common analysis system," *IBM System Journal,* vol. 43, no. 3, pp. 476-489, July 2004.

[32] B. Carpenter, The Logic of Typed Feature Structures. Cambridge Tracts in Theoretical Computer Science, (hardback) ed., vol. 32, Cambridge University Press, 1992.

[33] T. R. Payne and O. Lassila, "Semantic web services," *IEEE Intelligent Systems,* vol. 19, no. 1, pp. 14-15, 2004.

[34] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris and D. Orchard, "Web Services Architecture," World Wide Web Consortium (W3C), 11 February 2004. [Online]. Available: http://www.w3.org/TR/ws-arch/. [Accessed 2225 November 2013].

[35] OSGi Alliance, OSGi Service Platform, Release 3, IOS Press, Inc., 2003.

[36] OSGi, "OSGi Bundle Repository Specification," 2005. [Online]. Available: http://www2.osgi.org/Download/File?url=/download/rfc-0112_BundleRepository.pdf.

[37] W3C, "The Self-Describing Web," 7 February 2009. [Online]. Available: http://www.w3.org/2001/tag/doc/selfDescribingDocuments.html. [Accessed December 2012].

[38] "Definition of dereferenceable URIs from the W3C Linked Data Glossary," [Online]. Available: http://www.w3.org/TR/2013/NOTE-ld-glossary-20130627/#dereferenceable-uris.

[39] A. Diosteanu, A. Turbati and A. Stellato, "SODA: A Service Oriented Data Acquisition Framework," in *Semi-Automatic Ontology Development: Processes and Resources*, M. T. Pazienza and A. Stellato, Eds., IGI Global, 2012, pp. 48-77.

[40] "Official OWL ART API website," [Online]. Available: http://art.uniroma2.it/owlart/.

[41] M. T. Pazienza, A. Stellato and A. Turbati, "PEARL: ProjEction of Annotations Rule Language, a Language for Projecting (UIMA) Annotations over RDF Knowledge Bases," in *International Conference on Language Resources and Evaluation (LREC 2012)*, Instanbul, Turkey, 2012.

[42] "Official ISBN-A factsheet in the DOI website," [Online]. Available: http://www.doi.org/factsheets/ISBN-A.html.

[43] "Graph Pattern in the SPARQL W3C webpage," [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/#GraphPattern.

[44] S. Cassidy, "An RDF Realisation of LAF in the DADA Annotation Server," in *Fifth Joint ISO-ACL/SIGSEM Workshop on Interoperable Semantic Annotation*, Hong Kong, 2010.

[45] P. Ciccarese, M. Ocana, L. Garcia Castro, S. Das and T. Clark, "An open annotation ontology for science on web 3.0," *Journal of Biomedical Semantics,* vol. 2, pp. 1-24, 2011.

[46] R. Sanderson, P. Ciccarese and H. Van de Sompel, "Designing the W3C Open Annotation Data Model," in *Proceedings of the 5th Annual ACM Web Science Conference,* Paris, France, 2013.

[47] "Jape in the GATE official website," [Online]. Available: http://gate.ac.uk/sale/tao/splitch8.html#x12-2080008.

[48] P. Kluegl, M. Atzmueller and F. Puppe, "Textmarker: A tool for rule-based information extraction," in *Biennial GSCL Conference,* 2009.

[49] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler and F. Yergeau, "Extensible Markup

Language (XML) 1.0 (Fifth Edition)," 26 November 2008. [Online]. Available: http://www.w3.org/TR/REC-xml/. [Accessed 14 November 2012].

[50] S. Bischof, S. Decker, T. Krennwallner, N. Lopes and A. Polleres, "Mapping between RDF and XML with XSPARQL," *Journal on Data Semantics,* vol. 1, no. 3, pp. 147-185, 2012.

[51] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie and S. Jérôme, "XQuery 1.0: An XML Query Language (Second Edition)," 14 December 2010. [Online]. Available: http://www.w3.org/TR/xquery/. [Accessed 14 September 2013].

[52] M. Arenas, A. Bertails, E. Prud'hommeaux and J. Sequeda, "A Direct Mapping of Relational Data to RDF," World Wide Web Consortium (W3C), 27 September 2012. [Online]. Available: http://www.w3.org/TR/rdb-direct-mapping/. [Accessed 25 November 2013].

[53] S. Das, S. Sundara and R. Cyganiak, "R2RML: RDB to RDF Mapping Language," World Wide Web Consortium (W3C), 27 September 2012. [Online]. Available: http://www.w3.org/TR/r2rml/. [Accessed 25 November 2013].

[54] "Official Java Facelets website," [Online]. Available: http://facelets.java.net/.

[55] K. Verspoor, W. Baumgartner Jr, C. Roeder and L. Hunter, "Abstracting the types away from a UIMA type system," in *2nd UIMA Workshop at German Society for Computational Linguistics and Language Technology (GSCL),* Tagung, Germany, 2009.

[56] "Official AGROVOC website," [Online]. Available: http://aims.fao.org/standards/agrovoc/about.

[57] C. Caracciolo, A. Stellato, A. Morshed, G. Johannsen, S. Rajbhandari, Y. Jaques and J. Keizer, "The AGROVOC Linked Dataset," *Semantic Web Journal,* vol. 4, no. 3, p. 341–348, 2013.

[58] "Official FAO website," [Online]. Available: http://www.fao.org.

[59] C. Caracciolo, A. Stellato, S. Rajbahndari, A. Morshed, G. Johannsen, J. Keizer and Y. Jacques, "Thesaurus Maintenance, Alignment and Publication as Linked Data," *International Journal of Metadata, Semantics and Ontologies (IJMSO),* vol. 7, no. 1, pp. 65-75, Tuesday, 14 August 2012.

[60] A. Morshed, C. Caracciolo, G. Johannsen and J. Keizer, "Thesaurus Alignment for Linked Data Publishing," in *International Conference on Dublin Core and Metadata Applications*, 2011.

[61] M. T. Pazienza, A. Stellato, A. G. Tudorache, A. Turbati and F. Vagnoni, "An Architecture for Data and Knowledge Acquisition for the Semantic Web: The AGROVOC Use Case," in *On the Move to Meaningful Internet Systems: OTM 2012 Workshops*, vol. 7567, P. Herrero, H. Panetto, R. Meersman and T. Dillon, Eds., Springer Berlin Heidelberg, 2012, pp. 426-433.

[62] I. Gurevych, M. Mühlhäuser, C. Müller, J. Steimle, M. Weimer and T. Zesch, "Darmstadt knowledge processing repository based on UIMA," in *First Workshop on Unstructured Information Management Architecture at Biannual Conference of the GSCL*, 2007.

[63] "Official Stanford CoreNLP website," [Online]. Available: http://nlp.stanford.edu/software/corenlp.shtml.

[64] "Official National Agricultural Library's Agricultural Thesaurus website," [Online]. Available: http://agclass.nal.usda.gov/.

[65] "Official FAO's Agrontology website," [Online]. Available: http://aims.fao.org/aos/agrontology.

Biographical sketches

Manuel Fiorelli *University of Rome "Tor Vergata", Via del Politecnico 1, Roma 00133, Italy (fiorelli@info.uniroma2.it).* Mr. Fiorelli is a Ph.D. student in "Computer Science and Automation" at the University of Rome, Tor Vergata. He received B.S. and M.S. degrees in computer engineering from the University of Rome, Tor Vergata in 2008 and 2011, respectively. Then he joined the Artificial Intelligence Research Group at Tor Vergata University, where he researches on methodologies and architectures for knowledge acquisition and management.

Maria Teresa Pazienza *University of Roma Tor Vergata, Via del Politecnico 1, 00133 Roma (pazienza@info.uniroma2.it).* Prof. Pazienza is Full Professor in the Enterprise Engineering Department at the University of Rome, Tor Vergata, and Head of the Artificial Intelligence Research Group at University of Rome, Tor Vergata (http://art.uniroma2.it/). She coordinated for several years the NLP working group of the Italian Association for Artificial Intelligence, is on the editorial board of a few international journals and received for two times the IBM Faculty awards for her research on natural language processing and knowledge engineering. She is author or coauthor of more than 130 scientific papers. Prof. Pazienza is involved in several cooperation activities between the University of Tor Vergata and national/international scientific institutions for jointly developing researches and technologies. She is involved also in technology transfer activities for Italian companies. Currently she is involved in initiatives on Big Data.

Armando Stellato *University of Rome "Tor Vergata", Via del Politecnico 1, Roma 00133, Italy (stellato@info.uniroma2.it).* Dr. Stellato is Faculty Researcher in the Enterprise Engineering Department at the University of Rome, Tor Vergata. He received M.S. and Ph.D. degrees from the University of Rome Tor Vergata in 2002 and 2006, respectively. He researches and teaches in the fields of Knowledge Representation and Knowledge Based Systems. He is author of more than 60 publications and has been member of the program committees of over 30 international scientific conferences and workshops in the Semantic Web area. Dr. Stellato is involved in a framework research agreement between University of Rome, Tor Vergata and the Food and Agriculture Organization (FAO) of the United Nations as Semantic Architect, working on all aspects related to maintenance and publication of FAO RDF vocabularies such as AGROVOC, Biotech and Journal Authority Data (JAD). He is also leading development of VocBench, a web-based platform for collaborative management of RDF thesauri. In 2006, together with Prof. Pazienza, he received a UIMA Innovation Award for his research results on Semantic Annotation.

Andrea Turbati *University of Rome "Tor Vergata", Via del Politecnico 1, Roma 00133, Italy (turbati@info.uniroma2.it).* Dr. Turbati is Research Fellow in the Enterprise Engineering Department at the University of Rome, Tor Vergata. He received M.S. and Ph.D. degrees from the University of Rome, Tor Vergata in 2008 and 2012, respectively. Currently his interests cover Knowledge Based Systems and Systems for Semi-automatic Knowledge Acquisition. He is now one of the main contributors to VocBench: a web-based platform for collaborative management of RDF thesauri. CODA is an ongoing project which originated from studies that Dr. Turbati conducted during his PhD.