OKKAM: Towards a Solution to the "Identity Crisis" on the Semantic Web

Paolo Bouquet, Heiko Stoermer, Michele Mancioppi and Daniel Giacomuzzi
University of Trento
Dept. of Information and Communication Tech.
Trento, Italy
Email: {bouquet, stoermer, manchioppi, giacomuzzi}@dit.unitn.it

***Abstract*— One of the pillars of the Semantic Web enterprise is the idea that if people use standard names for resources (URIs), then the integration of information from different distributed sources will happen smoothly and efficiently simply by using URI identity as a key for merging RDF graphs into a single (virtual) graph. The question this paper will try to address is: how do we find and reuse an identifier for an entity on the (Semantic) Web? In this paper propose a system called OKKAM that is currently under development; we discuss requirements, architecture, usage scenarios and services we have developed so far to tackle this "Identity Crisis" on the Semantic Web.**

I. Introduction

In the W3C recommendation *Uniform Resource Identifier (URI): Generic Syntax* [1] , a resource is defined as "anything that has identity"[1]. This means that not only web accessible pages and documents are resources, but also people, cities and conferences; even the concept of "car" and the property of "being the owner" of a car are resources, which can be referred to and described as any other resource (e.g. in an ontology).

Despite the generality of the Semantic Web approach, here we want to suggest that – in practice – there is an essential difference between managing and reusing identifiers of resources which correspond to "things" (in a very broad sense, ranging from electronic documents to bound books, from people to cars, from conferences to unicorns) – we will call them *entities* –, and identifiers which correspond to abstract objects (like predicates, relations, assertions) – which we will call *logical resources*. Our thesis is the following: *while any attempt of "forcing" the use of the same URIs for* logical resources *is in principle likely to fail (as every application context has its own peculiarities, and people tend to have different views even about the same domain*[2]*), the same does not hold – or holds at a level which is philosophically interesting but of little practical relevance – for* entities. On other words, the claim is that there are compelling *theoretical reasons* why the Semantic Web (and any other semantically driven information system) should not force people to use shared URIs for logical resources, but only (or mostly) *practical reasons* why people do not use shared URIs for entities.

By analogy, our claim can be illustrated by considering the different difficulty of building white page and yellow page services. The former basically requires an efficient mechanism for listing entities, retrieving them, and distinguishing entities one from another; the latter always presupposes some taxonomy, which is typically either too general (and therefore does not help in discriminating services), or too specific (and therefore heavy to master for users), or too complex (not usable).

It should be clear that the problem of unique identifiers for resources (in its two flavors: logical resources and entities) is crucial for achieving semantic interoperability and efficient knowledge integration. However, it is also evident that 99% of the research effort is on the problem of (i) designing shared ontologies, or (ii) designing methods for aligning and integrating heterogeneous ontologies (with special focus on the T-Box part of the ontology). Perhaps because of its "practical" flavor, we must recognize that only a very limited effort has been devoted to address the issue of identity management for entities. For example, ontology editors, such as Protégé, support the "bad practice" of creating new URIs for any new instance created in an ontology. In our opinion, this problem is not only of general interest for the Semantic Web enterprise, but is one of the most critical gaps in an ideal pipeline from data to semantic representation: if we do not have a reliable (and incremental) method for supporting the reuse of URIs for the new entities that are annotated in new documents (or any other data source), we risk to produce an archipelago of "semantic islands" where conceptual knowledge may (or may not) be integrated (it depends on how we choose the names of classes and properties, and on the availability of cross-ontology mappings), but ground knowledge is completely disconnected. And since the most valuable knowledge is typically about individuals, we take this to be an issue that should be attacked.

In this paper, we introduce the main requirements and a

[1]'A resource can be anything that has identity. Familiar examples include an electronic document, an image, a source of information with a consistent purpose (e.g., "today's weather report for Los Angeles"), a service (e.g., an HTTP-to-SMS gateway), and a collection of other resources. A resource is not necessarily accessible via the Internet; e.g., human beings, corporations, and bound books in a library can also be resources. Likewise, abstract concepts can be resources, such as the operators and operands of a mathematical equation, the types of a relationship (e.g., "parent" or "employee"), or numeric values (e.g., zero, one, and infinity)' [1].

[2]This is what in [2], which was co-authored by one of the authors of this paper, was called the *distributed knowledge* argument.

prototype implementation of OKKAM[3], a service for supporting transparent integration of knowledge about entities through simple identity management support. OKKAM can be described at two different levels:

- the basic services – which belong to a module called OKKAMCORE – provide APIs to create and store URIs for entities, to add/modify/remove informal descriptions of each entity, to index the resources in which knowledge about an entity is provided (e.g. ontologies, web pages);
- on top of OKKAMCORE, OKKAM offers a collection of advanced services, including searching for already existing entities (using different search criteria), extracting information about entities, ranking results, supporting the reuse of URIs for entities in ontology editing, and so on.

The structure of the paper is the following: in Sect. II, we introduce our motivations and the resulting goals for our project in more detail. After that, in Sect. III the OKKAM system architecture and design approaches are illustrated. Sect. IV describes first basic services that have been implemented on top of OKKAMCORE, whereas Sect. V illustrates two usage scenarios we have addressed with the system. We conclude with a discussion of issues and an outlook on the further development of OKKAM in Sect. VI.

II. Goals and Motivations

As soon as one starts thinking about the idea of an entity repository, the temptation of building what Craig Knoblock[4] called an EntityBase in one of his recent talks, is very strong. In short, an EntityBase can be thought of as an entity-centric knowledge base, where knowledge is organized around entities instead of schemas (e.g. relational schemas or even ontologies). In such an approach, any entity type would be characterized by a collection of attributes (for example, for entities of type book, some attributes can be "author", "title", "date_of_publication" "publisher"), whose semantics is known in advance and explicitly specified.

We called this a temptation, as it is extremely appealing (we would always know what we know about an entity), but also very dangerous, as it presupposes a commitment on the meaning of an attribute which cannot be guaranteed in most practical situation by a repository which aims at being open, extensible, global. Therefore, an important requirement for our service is that it is light and fast, which can't be confused with yet another attempt in the direction of CYC [3] or SUMO [4], as systems of this type offered useful approaches in certain areas, but have obviously not contributed to a solution of the identity problem in the Semantic Web. What we are aiming to provide is a *naming service* for entities and *directory* containing entity profiles, not a knowledge base.

An Entity Profile stores *untyped* data about entities which will support the human user or an application using the OKKAM API to process descriptions about entities, and in effect enable them to assess whether the entity they want to store knowledge about in their own local KB already has a URI in OKKAM, or whether they have to create a new one. We store untyped data for the reason that typing an entity's attributes would require us classify the entity, which would be in contrast with the abovementioned goal. We do not discriminate types of entities, because we explicitly want to be able to provide naming and descriptions for *any* entity.

Of course at first sight one could think about what types of entities would be described in OKKAM, such as persons, artifacts, locations, companies etc.; this could make it appear sensible to provide a basic set of typed attributes for these entities. But we envision the system also to provide support for less obvious applications such as Named Entity Recognition from the field of Natural Language processing, which we will talk about later, in Section V. In these applications entities might represent a location or a piece of text in a document, a document itself or a collection of documents, and we end up with an unlimited set of potential types of entries, which makes it impossible to provide a common set of typed attributes for. Therefore it is our opinion that only untyped descriptive metadata can provide for the envisioned level of generality.

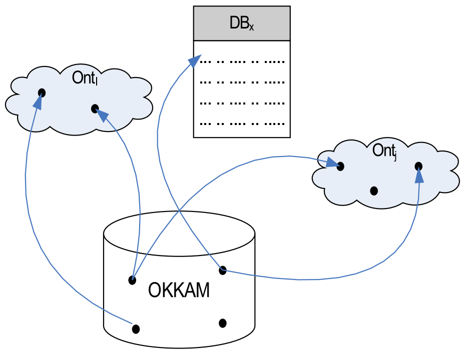


Fig. 1. Schematic overview of OKKAM, plus external K/I sources

In addition to these untyped data, OKKAM provides for the management of what we call *ontology references* in the Entity Profile, i.e. a set of URIs to external sources that are known to store information or knowledge about these entities, as illustrated in Fig. 1. One of the reasons to go in this direction was the motivation to make OKKAM provide a possible solution to integration issues in the Semantic Web. While a great amount of work has been performed on schema-level information integration[5], the aspect of *entity-level information- and knowledge integration* still offers many opportunities for providing interesting approaches. One possible application

[3]The system is named after Occam, a medieval philosopher whose main principle – known as the "Occam's razor" – was: *entities should not be multiplied beyond necessity* (in Latin: *entia non sunt multiplicanda praeter necessitatem*).

[4]Craig Knoblock's homepage: http://www.isi.edu/ knoblock/. Unfortunately, at this point no citeable publications about this topic are available.

[5]It is hardly impossible to cite all related work in this field. Specific to the area of the Semantic Web, the reader is referred e.g. to the publication list on `http://www.ontologymatching.org` for a host of publications, or the Ontology Alignment Evaluation Initative (`http://oaei.ontologymatching.org/`) which performs an alignment contest. For an overview more related to the database world, we refer e.g. to [6].

we envision to support with OKKAM is an extension-based equivalence check for classes in an alignment or integration process. Currently, in a schema-level integration process without extension check, classes can be *estimated* to be equivalent, but without an extension check the result of this estimation cannot be proved. Additionally, without a service that provides strong decision support about whether two individuals with the same name are actually identical or not (which is the current situation in the Semantic Web), an extension check will hardly deliver very reliable results. With the help of Entity Profiles in OKKAM we hope to improve this situation, because if we look at a case where two assumedly equivalent classes show that the sets of OKKAM-registered individuals associated to them are identical, we have very strong reason to support this equivalence assumption.

The last component of the Entity Profile is a set of assertions of identity between entities. We provide these for the case where two entities with different URIs in OKKAM are later discovered to describe the exact same object, and are thus identical. One possible criticism at this point is certainly the question how we can know and be certain about identity of entities. The answer is that we cannot. OKKAM will suffer from the same garbage-in-garbage-out property as any other information system. But with OKKAM at least we can provide a means for the Semantic Web to store and represent such information, and we hope that by consistent use of OKKAM in Semantic Web applications we can strongly improve the current situation by enabling agents to gain a certain level of confidence that they are actually “talking about” the same objects.

III. OKKAMCORE: CHARACTERIZING ENTITIES

A. *Data Model and API*

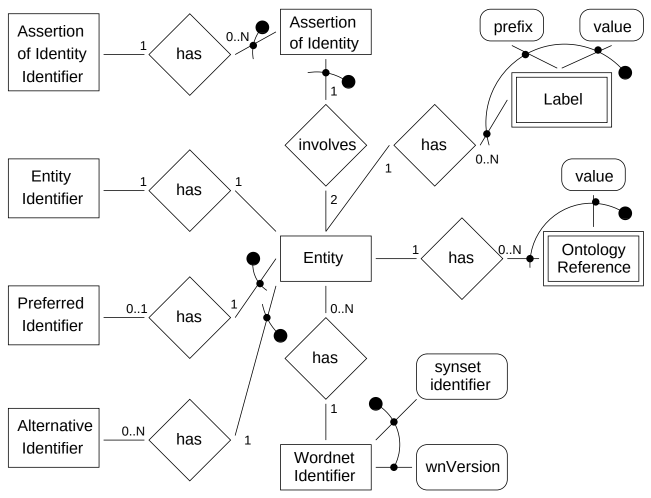


Fig. 2. OKKAMCORE’s Entity Relationship Diagram

The OKKAMCORE application manages data describing entities and assertions of identity between entities. The data structures we are using to model this are shown in the Entity Relationship Diagram presented in Figure 2. An *Entity* is in biunivocal relation with an *Entity Identifier*, which is created by the system and represents the URI of the entity in OKKAM with which users can identify entities within their application or KB. Each Entity may have a *Preferred Identifier*, provided by the user who creates the entity, e.g. to mirror an identifier used in their information system; the relation among Entities and Preferred Identifiers is labeled as a key because Entities can not share the Preferred Identifiers. Each Entity may any number of *Alternative Identifiers*; similarly to the Preferred Identifiers, every Alternative Identifier can belong to only one Entity at a time. To keep the diagram simple, it does not address the fact that there can not be overlap among the different types of identifiers. All the identifiers are names for the Entities on which they are set: then all the identifiers set on a given Entity are synonyms. Since different Entities have different names, each identifier can appear on at most one Entity, no matter if it acts as an Entity Identifier, a Preferred Identifier or an Alternative Identifier.

Entities can have any number of *Labels* set on them. Each Label has a *prefix* and a *value*. Label’s prefixes may be left empty. Different Labels with the same prefix and value can only belong to different Entities; as illustrated in Figure 2, the triple consisting of the Label’s prefix, the Label’s value and the Entity on which the Label is set forms a key. Each *Ontology Reference* has a value; any number of Ontology References can be set on an Entity. Similarly to the Labels, Ontology References with equivalent value can only belong to different entities.

The *Assertions of Identity* are uniquely identified by their *Assertion of Identity Identifiers*. Each Assertion of Identity involves exactly two Entities. Different Assertions of Identity can not involve the same two Entities.

OKKAMCORE provides its users with functionalities to manage and retrieve entities and assertions of identity. From a programmatical point of view, two APIs are provided:

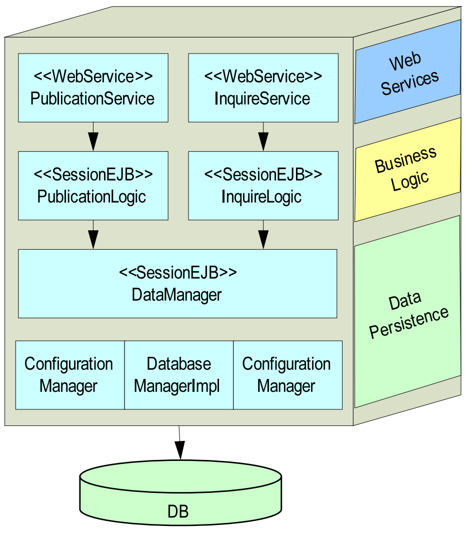
- Publication API: enables publishing, modifying and removing of entities and assertions of identity;
- Inquire API: allows retrieval of entities matching a given set of criteria.

Both APIs offer straightforward functionalities that one would expect in such a system; for the sake of brevity we will not describe the full API in this article.

B. *Architecture*

The currently available implementation of OKKAMCORE is built on top of the J2EE 1.4 platform. The OKKAMCORE application is an Enterprise Application exposing the Publication and Inquire APIs as Web Services. Its architecture, as presented in Figure 3, follows the classical three-tier model that subdivides the Presentation Logic (the Web Services), the Business Logic (carried out by an EJB module), and the Data Persistence logic. The Web Services framework we adopted is Apache Axis2[6]. The DataPersistence layer is subdivided into

[6]Apache Axis 2 Home: `http://ws.apache.org/axis2/`



<<WebService>> PublicationService
<<WebService>> InquireService
<<SessionEJB>> PublicationLogic
<<SessionEJB>> InquireLogic
<<SessionEJB>> DataManager
Configuration Manager
Database ManagerImpl
Configuration Manager
Web Services
Business Logic
Data Persistence
DB

Fig. 3. OKKAMCORE Application Architecture

different modules: i) handling the configurations of the application, ii) marshalling the internal representation of the data to external form, such as XML documents and iii) communication with the database. Thus, implementations exploiting different database technologies may be plugged effortlessly. For this reason, during the development of the OKKAMCORE application, we implemented two different Data Persistence modules: one based on a native XML database, and another one built on top of a relational database.

At first we developed the XML Database based backend because it allowed us to have a first prototype of OKKAMCORE running in a very short time. The backend is based on the Open Source database *eXist*[7]. Although the flexibility of the XML native database together with the XQuery expressiveness enabled us to complete the backend relatively quickly, we experienced scalability issues. It turned out that the number of entities that can be managed by this backend ranges in the tens of thousands, which is way below our desired goal. Although the XML database based backend performed well for testing of the rest of the application, and made the OKKAMCORE application promptly available for tests with the other services built on top of it, we decided to abandon this approach in favor of a relational database based backend.

IV. SERVICES

OKKAM can be viewed as a collection of services built on top of OKKAMCORE. In this section we list the main services which – in our opinion – should belong to OKKAM, describe their implementation (if available), or present ideas on how they could be implemented.

A. *Population of* OKKAMCORE

The first important service is the one that supports the population of OKKAMCORE with new entities. In fact, there are many issues that must be addressed and solved before new data is allowed to be stored. In particular, we want to stress the following:

- first of all, we want to add a new entity only if it is not already stored in OKKAMCORE. But this means that we need smart ways for recognizing if a new candidate entity is already stored, and for deriving when a new entity which looks like an entity already stored actually is a new one. These two requirements are crucial: failing to meet the first would lead to a lack of completeness (failing to support inferences which in theory are sound based on the fact that two names refer to the same entity); failing to meet the second would lead to a lack of correctness (false conclusions would be supported, based on the fact that two different entities have been collapsed onto a single identifier);
- imagine we detect that an entity is already stored, and that we find a new occurence of that entity in a document where some information about it is provided. Question: what if the new information conflicts with the old one? And, even before, how do we detect that there is such an inconsistency?
- as it will be clarified in the section on envisaged application scenarios, information may be imported in OKKAMCORE from very different sources, including humans (who may be carefully making data entry), ad-hoc wrappers designed to import entities from rich sources (e.g. lists of entities from Wikipedia), entity recognition tools (which may be extracting entity descriptions from free text). These potential sources may provide very uneven data, including a lot of garbage, which would undermine the role of OKKAM as a general and reliable tool;
- finally, a theoretical issue which needs to be addressed is the following: what does count as an entity? There is little doubt that people, organizations, cars, computer files, electronic devices, are entities. But, for example, is a document an entity? Is it an abstract entity, or it is identified with its physical realizations? If so, is every copy of a document a different entity? Another example is: are logical resources (like concepts, relations, topics) entities? Or the entity is the linguistic expression used to express a concept? But then are two linguistic formulations of the same concept different entities? And furthermore: are fictitious entities entities? Should we allow Pegasus and Spider Man to sneak into OKKAMCORE? And the list can be made much longer[8] . . .

To address these issues, we have developed the following compontents:

- *OkkamListsManager*
  On the WWW there are many lists of entities and are thus

[7]eXist Home: `http://exist.sourceforge.net/`

[8]We notice that a very practical version of these philosophical questions is the following: what should be represented as an instance in an OWL ontology? And what as a class/property? The issue is tricky, and we make only one example: should “Pizza Margherita” be a class or aan instance of an Italian food ontology? If we check e.g. [5], we find that the answer to this type of questions can be quite disappointing.

a potentially important resource for OKKAM. For example Wikipedia provides lists of countries, cities, members of particulars domains (e.g. Presidents of the United States, Computer Scientists, etc) that are exactly the types of entities that we want to store in the system. With the objective to find a standard mechanism for integrating these entities into Okkam we developed a language (an XML Schema) that describes the input that a data source has to follow to communicate with the population process of OKKAM. The main elements of the schema follows the internal structure of Okkam, in fact we have elements like ”Labels”, ”Label-prefix” or ”Label-value” that are easy to map with the tables elements of OkkamCore. This language is used by different wrappers that we developed and that try to convert the structure of a source list into the OKKAM input standard. For lists from the Web (Wikipedia, Yahoo, Google, etc.) the main purpose of the wrappers is the data cleansing process from HTML tags. After this step the entity collections is normalized with the objective to delete duplicates. Entities with the same annotation label are recognized by the system and the OKKAM administrative user can check if there exist conflicts from members of the list that are the same entity (from a logical point of view). During insertion, for ach entity the system searches OKKAM if there is already an entity with the same label/s. If yes, this entity is “frozen” and included in a set of entities that should be checked by the administrator before addition to the system, otherwise it is added immediately.

- *OkkamDBManager*
  Another important information source for OKKAM can be generic databases, as far as we have access to them. Examples might include direct database access to information systems such as extranets, online shops or publishing houses. In this case the transformation from the internal structure of the tables into the OKKAM input language is easier because the main objective of the process is writing queries that build the link between the database structure and the okkam data structure. When the transformation into the input language is completed, the rows that come from the database follow the same process that we already describe with web lists. With database sources the role of the user becomes more important because, with high numbers of entities, duplication and redundancy are an increasing problem.
- *OkkamManualEntry*
  Another solution we provide to insert new entities is the manual case. A Web interface provides easy access to the insert function. The user can add new entities, with labels, ontology references, etc., to the system using a form to specify all the information that he/she want describe the new entity with. As in the previous case, if the system finds a possible conflict with entities that are already in Okkam, it issues a warning message that informs the user of the possible error. This methodology of insertion is the slowest that Okkam provides, but it is the most precise and complete because the user can provide information that the system can not automatically discover, and optimize the input in a feedback loop.
- *Protege Plugin*
  We provide a plugin for the ontology editor Protege which we describe in further detail in Sect. V-A.

B. Searching for URIs

Another critical service is searching for identifier of some entity which is known either by some description (e.g. name for people), or by an identifier which was not issued by OKKAM. This site should be held very simple, like a traditional search engine, and based on an easy mechanism to visualize results. In a standard use case the user types in a keyword associated to an entity and the system searches the repository for instances that match this label. For example if a user searches for entities that have the label ”Heiko Stoermer” the OKKAM Management System will search in the database the instances that have this keyword and will return the main information about these. The main data will be the URI of the entity, the other labels associated to the entity, in our example can be ”H.Stoermer”, ”Stoermer”, Mr. Heiko Stoermer” and the classes of the ontologies where the entity is used. In our example we have different classes as ”Person” or ”PHD Student”. The information about the classes where other person use the entity, its URI, are very important because with this data the user can chose which entity is the correct URI in the OKKAM.

If we have two URI’s that share the same label ”Roma”, but one is attached to class like ”City” or ”Capital” and other refers to ”Person” or ”Customer” or ”Employee”, the user can easyily understand whether he needs the first URI because he wants to speak about the capital of Italy or the second one because he refers to a Person named ”Roma”. The filtering process, with information about classes, can be performed before the search step: the submitted query can be a pair of ”keyword - class”. This means that the system will return only the URI that fulfil both the terms of the interrogation. Web OKKAM This first use case is very simple and understandable because the most difficult process, the filtering task, is delegated to the user responsible for this operation.

The Web site of the OKKAM is not the only application built on the URI database. There are many situations where it is very difficult to believe that users use the web site to search the URI of the resources that they need. For example, if we have a large database with all employees of an organization is impossible that the designers and developers wanting to build semantic application on this data search in the OKKAM web site all the URI’s of the persons stored in their database. This process can be simplified if they can use an automatic service, in this case a web service, that provide an access point to the OKKAM that an application can use. The developers can build an application that extract the data from their database and send them to the web service which will return some results, URI, about the information that already are stored in the OKKAM.

V. Two usage scenarios

A. Runtime support for ontology editing

Another important area for which OKKAM has to provide services and applications are existing Semantic Web tools. In particular, ontology editors are applications where users build a formalization of part of the world by means of classes and instances of these classes, all identified by URI's. One of the most widely used and important editors is Protege, an open source product that can be extend and modified with "plug-ins" added on the core system. For the OKKAM vision it is of high importance to develop a plug-in for this application which provides a connection with the URI database when users create new instances of a class, which we are doing as illustrated in Fig. 5. If a user creates a new instance of a class, instead of assigning an arbitrary, meaningless number as ID the plug-in will search the repository whether an URI already exists that can be assigned to this new instance. The selection process is envisioned similar to the web search use case where a list of URI's that match the label for the new instance are visualized to the user.

Important support for all the selection processes comes from additional tools, as for example WordNet, that provide information about the meaning of the classes used in the ontologies where the new instances are created. With this information the system has more data to try to recognize the correct URI to return to the users or application that query OKKAM.

B. Supporting Knowledge Extraction and Representation

One of the scenarios we are currently implementing with the help of OKKAM is to support Knowledge Extraction (KE) processes and the resulting Knowledge Representation (KR) in a Semantic Web project[9] that aims at building a large-scale Knowledge Base (KB) from information stored in distributed document bases. The architecture comprises a pipeline of processes that covers all steps from KE to the building of the KB (the so-called *Semantic Resource Network*) for end-user services, as illustrated in Fig. 5

Within the pipeline there are several points of application imaginable, two of which we have currently implemented and are further described here:

- *Information Extraction: Named Entity Recognition and Coreference*
  Whenever our NLP process recognizes a named entity in a piece of text, it interacts with OKKAM to analyze whether this named entity already has a unique URI. If yes, the NLP process stores locally[10] the fact that a uniquely identified entity has been discovered with additional information such as its location in the document, etc. If the entity does not have a URI yet, an Entity Profile is created in OKKAM and the resulting URI is used accordingly. For subsequent discoveries of the same named entitiy, the same URI will be used to indicate that the two discovered entities are in fact the same, just in different locations of the document. This approach is equally applicable to discovered coreferences[11]. If the NLP process updates the Entity Profiles in OKKAM correctly, we gain direct access to search situations of the type "show me all documents that talk about this entity", as the respective links would be stored as Ontology References which we can evaluate and reason about with a higher-level service.
- *Refinement: Identity Discovery*
  In the refinement phase, as depicted in Fig. 5, we can address shortcomings of the NLP processes in terms of discovery of identity. The VIKEF pipeline has dedicated a whole processing step to this issue, as – at the named entitiy extraction level – it is not always possible to detect identity between entities. Obvious examples in this case are missing correspondences between orthographic variations hinted at already in Sect. IV-B, e.g. the fact that within one document there is a certain probability that the strings "Stoermer", "H. Stoermer" and "Heiko Stoermer" denote the exact same individual. With support of the OKKAM system, we have implemented several heuristics to address this issue, the simplest performing a substring query to OKKAM and using a string similarity measure on the results to choose candidates for establishing an assertion of identity between them, and thus to cluster annotations. A higher level process is free to either choose one single URI for all the annotated entities or to retain the original URIs, as it is always possible to perform clustering via analysis of identity assertions in OKKAM.

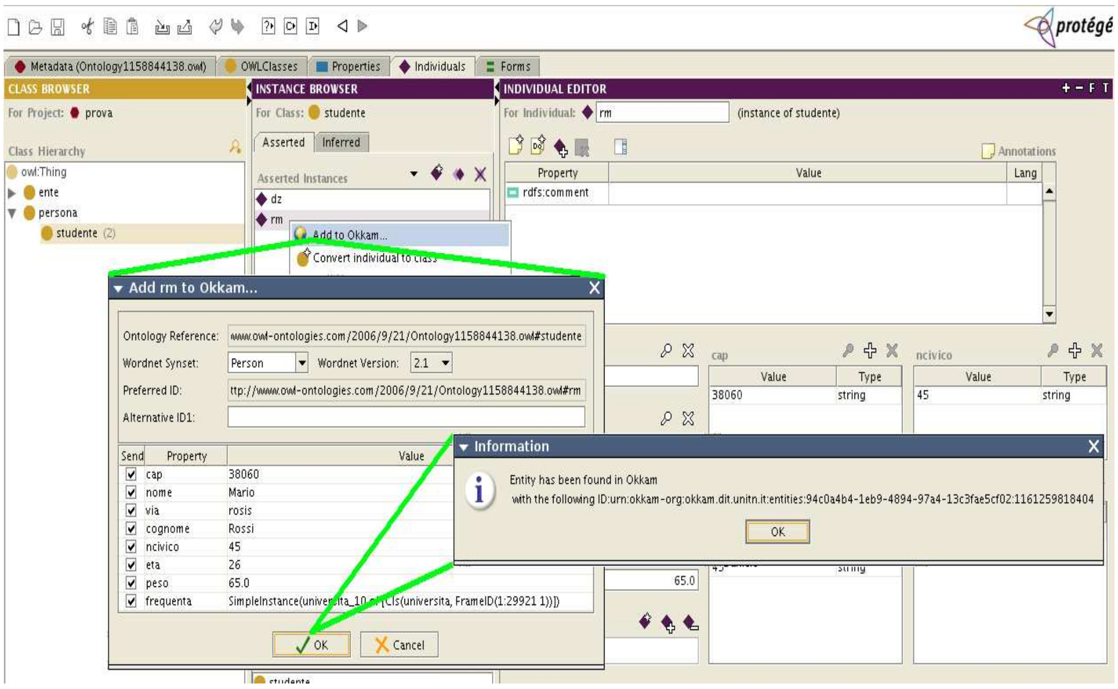
[9]see `http://www.vikef.net` for further information about the VIKEF project.

[10]In fact, the annotations created in this phase are stored in an XML file, which is later refined and then used as a base for the generation of RDF annotations that will be fed into a large knowledge base.

VI. Discussion and Conclusion

OKKAM is the typical example of an application which is not based on some radically new scientific result, but aims at filling a gap by using existing technologies in a new way. In our opinion, without OKKAM (or a similar service), most Semantic Web promises will never be kept, as it provides a sort of bottom level for integration which cannot be achieved *ex post* when the ball stops. However, the fact that the basic technologies are already available should not lead us to underestimate the critical factors which may affect the success and adoption of OKKAM. In addition to aspects already discussed throughout the paper, we identify acceptance issues in the form that not every party involved in the Semantic Web may be willing to use a centrally managed service that is outside of their control. Privacy issues include all the well-known aspects of data security, access management, privacy etc. that almost all public information systems share. Last, but not least there are of course questions of offered features and

[11]A coreference is a linguistic pattern typically involving pronouns when talking about an object that has previously been named. Example: "Peter is a good runner. He does 10k in 45 minutes." The personal pronoun *he* establishes the coreference in this case.



Add rm to Okkam...

Ontology Reference: www.owl-ontologies.com/2006/9/21/Ontology1158844138.owl#studente
Wordnet Synset: Person
Wordnet Version: 2.1
Preferred ID: ttp://www.owl-ontologies.com/2006/9/21/Ontology1158844138.owl#rm
Alternative ID1:

Send	Property	Value
✓	cap	38060
✓	nome	Mario
✓	via	rosis
✓	cognome	Rossi
✓	ncivico	45
✓	eta	26
✓	peso	65.0
✓	frequenta	SimpleInstance(universita_10 of [Cls(universita, FrameID(1:29921 1))])

OK Cancel

Value	Type
38060	string

Value	Type
45	string

Information

Entity has been found in Okkam
with the following ID:urn:okkam-org:okkam.dit.unitn.it:entities:94c0a4b4-1eb9-4894-97a4-13c3fae5cf02:1161259818404

OK

Fig. 4. A Protege plugin for generating individuals registered in OKKAM.

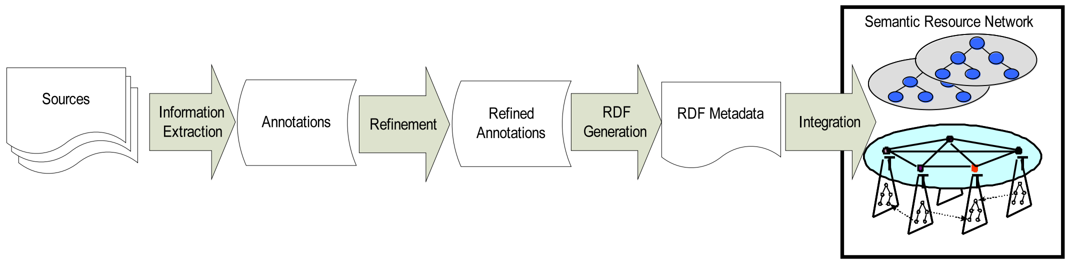


Fig. 5. Knowledge pipeline to be supported by OKKAM

functionality, such as a really efficient and intelligend search and ranking mechanism for Entity Profiles in OKKAM, as well as performance and scalability issues which are again common to most information systems. Our planned next steps are to address exactly these issues in the form of further research and by developing additional services on top of OKKAMCORE.

We conclude with the statement that currently, when creating ontologies, people actually perform two different tasks: they specify a conceptualization, and then "populate" such a conceptualization with instances by assigning instances to some class and specifying the values for properties (if any). It is a trivial observation that the same domain (set of entities) may be used to populate different ontologies (e.g. we may have two different conceptualizations of Italian wines&food), and that any two ontologies (e.g. an ontology about semantic web researchers and another about people living in Italy) may have overlapping domains. Creating a conceptual schema and then populating it with instances address two different issues: the first is an *epistemological* issue (it has to do with knowledge about the world), the second is an *ontological* issue (it has to do with existence).

From a design perspective, what we propose is to keep these two tasks separated: on the one hand, we need a universal and non ambiguous way to refer to the entities about which an agent may have some knowledge; on the other hand, we need a way to specify knowledge about these entities. We believe that the help of OKKAM this goal can be achieved more cleanly for the Semantic Web, as to existing methods of specifying knowledge in the form of ontologies and knowledge bases we add an identity and reference architecture with a central character that enables systems and agents to ensure that they "talk" and store knowledge about the same entities, if these objects share the same identifier.

VII. ACKNOWLEDGMENTS

This research was partially funded by the European Commission under the 6^{th} Framework Programme IST Integrated Project VIKEF - Virtual Information and Knowledge Environment Framework (Contract no. 507173, Priority 2.3.1.7 Semantic-based Knowledge Systems; more information at

`http://www.vikef.net`)

References

[1] T. Berners-Lee, R. Fielding, and L. Masinter. *RFC 3986: Uniform Resource Identifier (URI): Generic Syntax*. IETF (Internet Engineering Task Force), 2005. http://www.ietf.org/rfc/rfc3986.txt.

[2] Matteo Bonifacio, Paolo Bouquet, and Paolo Traverso. Enabling distributed knowledge management: Managerial and technological implications. *Informatik - Zeitschrift der schweizerischen Informatikorganisationen*, 1:23–29, 2002.

[3] Douglas B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):32–38, 1995.

[4] Ian Niles and Adam Pease. Towards a standard upper ontology. In *FOIS*, pages 2–9, 2001.

[5] Natalya F. Noy and Deborah L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford University. http://protege.stanford.edu/publications/ontology_development/ontology101.html.

[6] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350, 2001.