Describing semantic web applications through relations between data nodes

Enrico Daga[1], Mathieu d'Aquin[1], Aldo Gangemi[2] and Enrico Motta[1]
(1) Knowledge Media Institute, The Open University
{enrico.daga,mathieu.daquin,enrico.motta}@open.ac.uk
http://kmi.open.ac.uk/
(2) Institute of Cognitive Sciences and Technologies, Italian National Research Council
aldo.gangemi@cnr.it
http://istc.cnr.it/

Report, May 21, 2015

Abstract

Semantic Web Applications can only be understood if the complex data flows they implement are clearly described. However, application developers have very little support at the moment for documenting such data flows and their rationale, in an appropriately formal and conceptual manner. In this paper, we propose to apply a knowledge engineering approach to the formal description of Semantic Web Applications. Following an ontology building methodology based on the analysis of several existing Semantic Web Applications, we devise an abstract, foundational model of data flow descriptions which focuses on the graph of relationships between "data objects" within the application, rather than the tasks and processes being implemented. The result is Datanode, a conceptual framework designed to describe systems by expressing relations between data nodes (objects), implemented as an ontology of the possible relationships between such data nodes. We evaluate Datanode by applying it to the description of eight recent Semantic Web Applications.

Keywords: Semantic Web Applications, Linked Data, Ontology, Knowledge Engineering, Data Integration, Data Documentation

1 Introduction

Semantic Web Applications can be considered as systems operating in the World Wide Web that follow principles or incorporate technologies of the W3C Semantic Web such as RDF and other standards [1]. We restrict this pragmatic definition by considering as Semantic Web Applications systems that, in addition: (1) make use of information from sources they do not control; (2) manipulate it to produce new knowledge; (3) process the resulting data for presentation in a new form.

This definition embraces a set of systems whose complexity makes them hard to be apprehended by final users, not to mention software agents. The output of Semantic Web Applications is the result of a process which is hidden in the code. However, providers of applications have no method for declaring the rationale behind the information they offer to the user. How are the resulting data justified? How the presented data connect to the sources exploited by the application? Which were the assumptions that have been considered to support the implementation? Answering these questions, along with providing formal support to interpreting the output of a Semantic Web application, could help users understand and trust it. There should be a way to provide a formal context to the data that Semantic Web Applications have now started to publish and republish in multiple forms on the WWW. We argue that we can document this diversity coherently. This documentation should be generic enough to be fruitfully exchanged, and expressive enough to be useful. To obtain this, it is necessary to abstract from the specific technologies and implementations of concrete systems.

In this paper we propose to apply a knowledge engineering approach to the formalisation of descriptions of Semantic Web Applications. We observed that these systems have a strong focus on "data objects" that may emerge as input or output at different steps of the process and/or layers of an application's architecture. We started from this common ground. We decided not to represent them in terms of tasks and processes but to describe them as a graph connecting "data nodes". The result is Datanode, a conceptual framework designed to describe systems by expressing relations between data objects, which is implemented as an ontology.

We report on the design and implementation of Datanode and on its evaluation by applying it to the description of eight Semantic Web Applications. The evaluation demonstrates that these applications can be understood as networks of data nodes.

2 Related work

The problem of trustworthiness of information on the web of data can be seen as a provenance issue. In [2] the authors develop a vocabulary to annotate data with provenance information, including metadata about the process, involved software and provider. The goal is to support the selection of linked data sources based on trustworthiness. The Linked Data Value Chain [3] is a conceptualisation targeted to the design of business models that make use of linked data. This model can be used to analyse the roles of providers and data in a linked data applications. A component based approach to the design of Semantic Web Applications have been discussed in [4]. Our research problem raise a complementary issue: how to support the *understanding* of data that is produced by Semantic Web Applications?

In [5] an approach based on knowledge engineering has been applied to classify Semantic Web Applications with respect to reasoning patterns. In this work the objective is to abstract the tasks to design inference processes. Similarly, Semantic Web Applications have been classified in [1] by analysing the functionalities offered to the users with the aim of extracting generic requirements to support and improve the development methods. These efforts were targeted to understanding the structure of a system while we focus on the characterisation of the data manipulated in order to support the interpretation and documentation of its output.

The description of data flows as workflows is part of traditional ETL[1] and research on data integration [7] in database systems. Rather than the way data are managed, which is what the relational model is about, we care about the way they are manipulated. Workflow description is also part of the research on Semantic Web Services [8]) and, more recently, on Linked Stream Data processing [9]. Workflow-centric Research Objects [6] have been designed with the objective of make persistent (and reproducible) research experiments in the scientific discourse. However, workflows are made of agents, tasks, processes and methods. We are interested in analysing the relations beween data objects instead.

Our task is comparable to the documentation of software architectures [10]. The literature in this field emphasises the multiplicity of structures existing in information systems. We choose the point of view of data and designed an ontology to describe data flows. Our design methodology has been iterative, incremental and test-driven, and is strongly inspired by Extreme Design in ontology engineering [11].

[1] `http://en.wikipedia.org/wiki/Extract,_transform,_load`

3 Knowledge Engineering Methodology

Understanding the output of systems requires a certain degree of confidence in the domain of the application as well as with the methods and tasks the application implements. The main question here is: how to devise a common, generic conceptualisation to describe heterogeneous and diverse data flows? Our methodology to build such a conceptualisation therefore starts from a survey of existing Semantic Web Applications. It is summarised as follow:

a) **Selection**. Select applications described in recent contributions to Semantic Web conferences by using the definition of Section 1.
b) **Acquisition**. Isolate the descriptions of each application from the related paper.
c) **Use case abstraction**. Generalise the description by translating it to a generic synopsis.
d) **Design**. Annotate the synopsis with names of classes and properties that appear in the text; generate a draft ontology and then refine it through discussions.
e) **Testing**. Model the synopsis using the ontology. Testing was done in parallel with the design phase, until all cases were fully covered.

In the following sections we report on the process that led to the Datanode ontology. As illustration, we will use the evolution of a snippet from DBRec [14]. We will leave out the testing phase, that is covered in Section 5, where we discuss the evaluation of the ontology through descriptions of concrete systems.

3.1 Selection & acquisition

We selected six applications described in recent contributions to Semantic Web conferences that fit well our definition of Semantic Web Application: Aemoo [12], DBRec [14], DiscOU [15], Spud [13], Yokohama Art Spot [16], EventMedia [17]. All these are applications that do not start from a dedicated dataset, but transform some existing sources for some final task. We added Rexplore [18] and IBM Watson [19] at a later stage, to strenghten our evaluation (see Section 5). We started by isolating the portions of text describing the system. When the system was including several different features and descriptions spread over many points in the document, we isolated all of the snippets and merged them in a unique picture at a later stage. This example is extracted from the discussion of a data preparation process in DBRec:

"On the other hand, we identified lots of redundancy and inconsistencies in our DBpedia subset. Especially, many links between resources are defined redundantly as http://dbpedia.org/ontology/xxx and at the same time as http://dbpedia.org/pro-perty/xxx. We then removed duplicates, leading to 1,675,711 triples, i.e. only 55.7% of the original dataset."

3.2 Use case abstraction

The aim of this activity was to try and extract generic use cases from specific ones. We went through each description and translated it into a list of atomic sentences, each describing a step of the process (or a specific feature). We obtained a synopsis from each text snippet. During this process we tried to abstract from the specific examples, replacing peculiar aspects with their generalisation, when possible. We obtained a set of synopses, covering several aspects of Semantic Web Applications, similar to the following:

1. having data with redundancies and inconsistencies
2. remove redundancies and remove inconsistencies
3. remove duplicates (same as redundancies?)
4. data is now 55.7 percent of before

As a result of this process we obtained a list of text snippets describing system features, each coupled with a synopsis. The whole set of synopses constituted our modelling requirements.

3.3 Ontology design

We traversed the collection of synopses and treated each sentence as standalone. We annotated each sentence with a set of keywords. We then looked at the keywords and modified them in order to make them represent either a verb or a type of things. In the above case, we wrote it capitalised:

0) access, Data, Redundancy, Inconsistency
1) remove, Redundancy, Inconsistency
2) remove, Duplicate
3) Data, Now, Before

We traversed all synopses several times, trying to harmonise the concepts (e.g., merging synonyms) while maintaining the specificities emerging from each case.

We then collected all 159 keywords from our annotations, analysed them trying to reduce redundancy (for example normalising names to be singular). The result was a list of concepts and verbs. We automatically generated a draft ontology considering all concepts to be istances of `owl:Class` and all verbs to be relations between (yet) unspecified things. This draft ontology included 132 classes and 168 properties, covering several different aspects of a Semantic Web Application. Here is a sample list: Operation, Collect, Optimize, Capability, newer-version-of, not-fits, before, Entity, Dataset, about, Summarize, component-of, Stand-in, concern-of, derive-from. The next step of the process required to abstract and simplify this automatic, disorganised and heterogeneous ontology. This was based on two main observations: 1- many of the types represented data sources of objects in various forms; and 2- many of the processes could be represented through the relationships between their input data and their output data. The foundation of the Datanode ontology is therefore devised as a simple abstraction including a unique class, Datanode, which represents "data" in a broad sense (data sources of any kind or format), and formalised relationships between these datanodes.

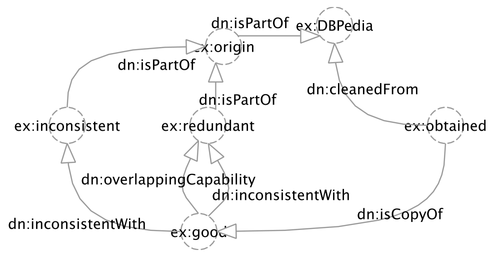

Figure 1: Example test case, from DBRec [14]

With this foundation established, we then refactored all the elements obtained in the previous steps so that they are either abstracted into the Datanode class, or formalised as properties of the Datanode class, as described in the next section.

4 The Datanode ontology

The Datanode ontology terms are under the `http://purl.com/datanode/ns/` namespace[2]. The ontology defines a unique type - Datanode - and 114 relations, starting from a single top property: relatedWith, having Datanode as domain and range. We found data to have several different facets in the sys-

[2]Documentation is online at `http://purl.org/datanode/docs/`. It includes a tree diagram of the relations and usage examples.

tems' descriptions, examples include: sets of data, data sources, identifiers, individuals - but even methods (like in Aemoo [12]). The Datanode class abstracts from the notion of dataset and it is independent from the notions of containment, individual or identifier. A *datanode* is any data object. The class groups datasets, individuals, schema elements such as classes and properties, as well as identifiers under the same umbrella. The nature of this concept is voluntarly underspecified. Datanode relations can be grouped in ten "branches" covering several aspects of a semantic web data flow:
meta. This branch covers the relations between something and its metadata. "Meta" is used to designate a relation with information that applies to the datanode as a whole, possibly including it. The top relation is "*metadata*", which has for inverse "*about*". This kind of relations specialises as *describes/describedBy*, *hasAnnotation/isAnnotationOf* and *hasStatistic/isStatisticOf* (we will follow this convention to pair relations that are inverse of each other).
reference. The top relation is *references/referencedBy*. A datanode includes a mention to something which is another datanode. This branch covers linking (as intended in the context of linked data) - *links/linkedBy*; as well as dependencies between data objects (*hasDependency/isDependencyOf*) and between data and a schema definition, for example an ontology (*usesSchema/schemaUsedBy*).
derivation. This relation indicates that a datanode is the origin of another, in the sense that the second have been produced using the first as information source. It is possible that a source *hasInterpretation* (*isInterpretationOf*) some other data, as a result of a mining process - *hasExtraction/isExtractionOf*; or a logical process - *hasInference/isInferenceOf*. A relation *processedFrom/processedInto* is identified when a derivation is also the transformation of a data node into another one, being it *cleanedFrom/cleanedInto*, *optmizedFrom/optimizedInto*, *refactored*, *remodelled* or the result of a computation - *hasComputation/isComputationOf*. It is possible to derive a new data node by making a copy - *hasCopy/isCopyOf*, when making snapshots or caching strategies (*hasSnapshot/isSnapshotOf*, *hasCache/isCacheOf*). We consider also the case when a derivation is done by cloning a part of a datanode with specific features - *hasSelection/isSelectionOf*.
adjacency. adjacentTo is used to represent proximity between two datanodes without assuming any special relation at domain level. Proximity may result from being parts of the same dataset - *disjointPartWith*. Another case of proximity is between an object and its attachment - *hasAttached/attachedTo*. The *hasAnnotation/isAnnotationOf* relation, mentioned above, specialises also *hasAttached/attachedTo*.
partition. A datanode may be described as container of other datanodes:

hasPart/*isPartOf*. A first distinction is done via *hasSection*/*isSectionOf* and *hasPortion*/*isPortionOf*. We call "section" the partition of a datanode by the means of a set of attributes (with no information about its population), while "portion" is a partition done by keeping a part of the population (with no information about the attributes). In the extreme case, *hasSection* can be further specialised into *hasIdentifiers*/*identifiersOf*. A portion can also be a sample - *hasSample*/*isSampleOf*. Deriving by selection is also a way of isolating a part of the source, thus *hasSelection*/*isSelectionOf* appears also on this branch, with the sub-property *hasExample*/*isExampleOf*.
vocabulary. The range of *hasVocabulary* is a datanode which enumerates a set of terms that are all used by the subject data node as identifiers (inverse is *isVocabularyOf*), to name structural elements like *hasDescriptors*/*descriptorsOf*. These can be attributes, relations, or to link a datanode to its types - *hasTypes*/*typesOf* or datatypes with *hasDatatypes*/*datatypesOf*. This is different from the case of a relation with a schema (under "references"): a schema is defined independently from the actual data.
version. This property implies a temporal relation between two data nodes that are meant to be the same at a different point in time. We find under this category relations such as *newerVersionOf*/*olderVersionOf* and *nextVersionOf*/*previousVersionOf*. versionOf is symmetric and does not specify a direction. It is not transitive. While it can be argued that the identity of something tracked over time should not change, thus implying transitivity, we want to support the case when a datanode has more than one single following version (branching).
update. A data node may be related to another because it contributes to improve its validity. *hasUpdate* (inverse *isUpdateOf*) may happen in the form of a new version that is meant to substitute the original (*hasUpdatedVersion*/*isUpdatedVersionOf*, which is also under the *hasVersion* umbrella). While the updating (as activity) can be seen as a way of versioning, the two concepts are different when modelled as relations between datanodes. An update may not replace its target datanode. It is the case when the update is meant to modify the data node: *hasChange*/*isChangeOf* and its sub-properties *hasAddition*/*isAdditionOf* and *hasDeletion*/*isDeletionOf*.
consistency. This aspect is covered by two properties: *consistentWith* and *inconsistentWith*. We intend (in)consistency in a wide sense: two datanodes are (in)consistent because they are (not) compatible in some fundamental respect. For example, (*in*)*consistentWith* may be used to indicate a data node that should (not) be used together with another. In some settings, it may be useful to specify that some datanodes are consistent with others, eventually to state that a datanode is *consistentWith* itself. It is also possible that a datanode is *inconsistentWith* itself, meaning that it is wrong, buggy or

somehow corrupted.
capability. Capability is intended as *"the power or ability to generate some outcome"*[3]. Similarly to consistency, capability is covered with two separate branches starting from *overlappingCapabilityWith* and *differentCapabilityFrom* respectively. Two data nodes may have similar potential. This may refer to any kind of feature, be it structural (e.g., they share schema elements), physical (e.g., they are both in XML) or related to the domain (they both talk about Music Artists) - to name but a few examples. This relation is intentionally left abstract since there might be many different ways to express capabilities. Extensions for specific use cases can of course be made as specialisations of this relation. Datanode therefore only contains a few general cases: *overlappingVocabularyWith* and *overlappingPopulationWith*, both leading to *redundantWith*, *sameCapabilityAs* and duplicate - all describing a similar phenonmenon with different intentions. Under this scope we also positioned *optimizedFrom/optimizedInto* - to state the empowerment of an existing capability; and *cleanedFrom/cleanedInto* - to represent the result of a process aimed to make emerge a potential capability whilst maintaining another fundamental one. In contrast, two datanodes may have different potential. For example, two data nodes use different vocabularies (*differentVocabularyFrom*, or *disjointSectionWith*, when parts of the same datanode) or have different population (*differentPopulationFrom*, or *disjointPortionWith*). When both vocabularies and populations are different it is possible that two datanodes have disjoint capabilities - *disjointCapabilityWith*.

5 Describing systems: evaluation in practice

In the introduction we made the assumption that a knowledge engineering approach could produce a method to represent the diversity of semantic web applications coherently. In this section we describe how the Datanote ontology can be used to describe eight different systems. Six of these applications are the ones used in the survey part of the methodology. For the evaluation phase we added two more applications, Rexplore [18] and IBM Watson [19]. This evaluation corresponds to testing the requirements that have been extracted from the survey on Semantic Web Applications. We consider this an evaluation of both the abstract conceptualization - foundation of the ontology, and of the ontology itself.

To setup an evaluation use case we followed a precise process starting from the textual descriptions of the systems:

1. isolate portions of text describing the system (similarly of what hap-

[3]Definition from `http://en.wiktionary.org/wiki/capability`.

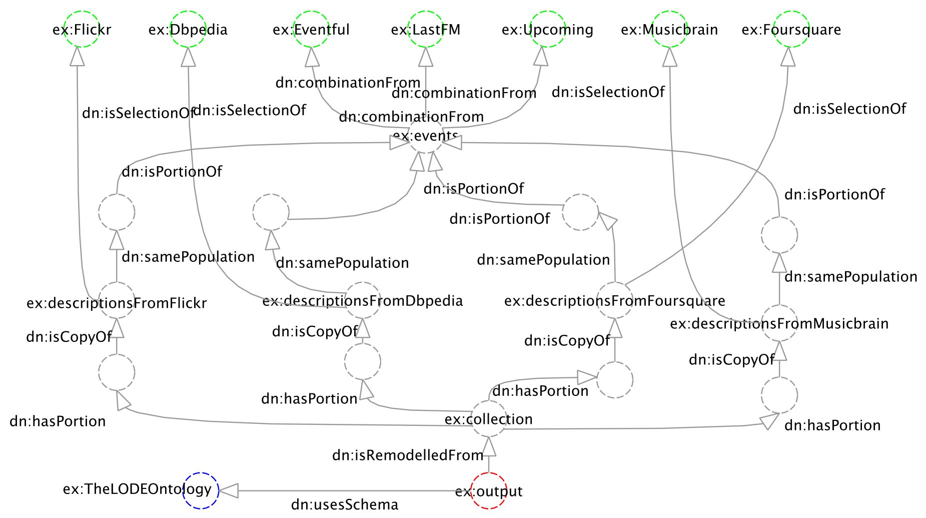

Figure 2: Description of the reuse of sources in the EventMedia mash-up [17]. The input sources are the top nodes. The node at the bottom depicts the output data, which is a remodelling of the data collected from various sources according to a specific schema. Used branches: reference, derivation, partition, capability.

pened in the initial phase of our methodology, see Section 3);

2. identify data source(s) and data output and model them as datanodes;
3. connect the two using the properties defined in the ontology to express the data flow of the system, by adding any necessary datanode in between.

The use cases are grouped following the definition of Semantic Web Applications mentioned in Section 1.

SWAs make use of information from sources they do not control. **EventMedia** [17] is a a web-based system that exploits real-time connections to enrich content describing events and associate it with media objects[4]. The emphasis here is on the linkage with multiple sources for collecting event data and multimedia objects. Figure 2 displays how data in EventMedia is kept from three public event directories and is enriched with data from sources like the BBC or Foursquare [17]. The information is then presented using the LODE ontology. Similarly, **Yokohama Art Spot** relies on linked data to present the cultural offers of artistic institutions spread in the area of Yokohama [16] - see Figure 3 .

[4]See `http://eventmedia.eurecom.fr/`.

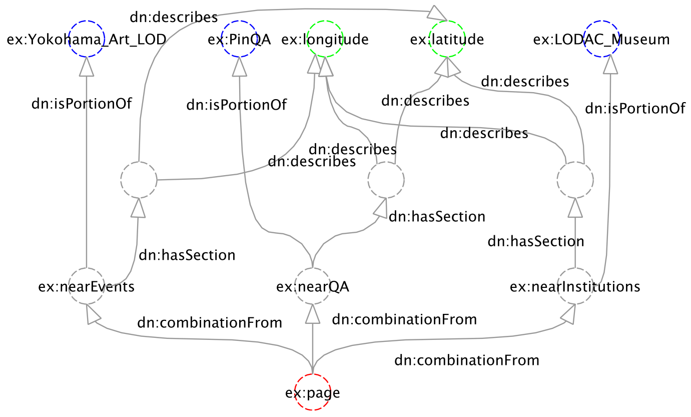

Figure 3: Starting from the user's location, Yokohama Art Spot [16] aggregates information from artistic events and museums nearby, packing it in a single web page. Used branches: derivation, partition

.

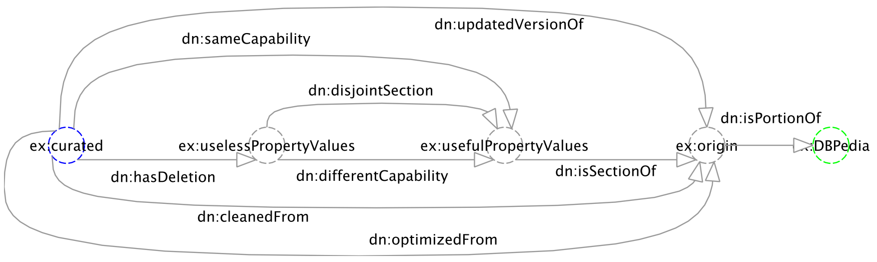

Figure 4: Syntetic description of the DBRec data preparation process. Used branches: derivaton, partition, update, consistency and capability.

SWAs manipulate information to produce new knowledge **DBRec** is a music recommendation system built using DBPedia [14]. The paper, among other aspects, describes the preparation of the data done to maximise the efficiency of the Linked Data Semantic Distance algorithm. The Datanode graph depicted in Figure 4 shows how a process that goes from the DBPedia dataset to an optimized and curated dataset can be formalised. **SPUD** exploits semantic technologies *to obtain business results in an environment with hundreds of heterogenous real datasets coming from different data sources and spanning multiple domains* [13]. The system is a web application with several functionalities. One of them is the "*incremental semantic lifting of data. [...] We capture provenance (using the latest W3C working draft) by making views over datasets immutable*" [13]. It is quite easy to show this engineering solution purely at the knowledge level using datanodes (see Figure 5). In the same application, some data are published in a slightly changed fashion

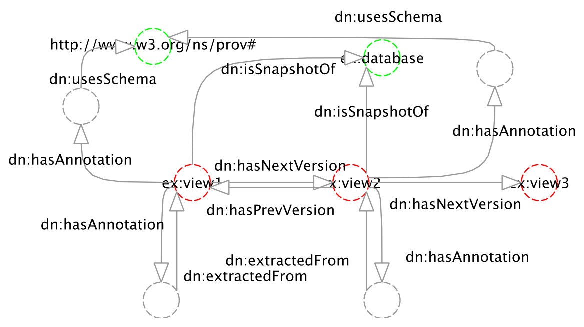

Figure 5: In Spud [13] the database is versioned. Each view is a snapshot of the data, annotated with provenance information. Semantic linfting is executed and stored in the views as data enrichment (they are the nodes at the bottom, using the pattern hasAnnotation-isExtractedFrom). Uses: reference, adjacency, version, derivation

in order to protect sensible information. Figure 6 represents this solution. The published data can be described as composed by two sections, one containing all the non private information from the database, the second being an anonymisation of the private data (the new knowledge, in our definition). **DiscOU** is presented as a *discovery engine for Open Educational*

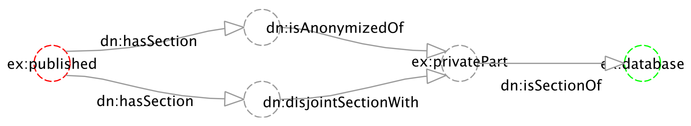

Figure 6: In Spud [13] information which is publicely shared contains the part of the database with no privacy concerns as well as some data which is processed to hide sensible information. Uses: partition, capability, derivation.

Resources Using Semantic Indexing and Relationship Summaries [15]. The indexing component stores DBPedia entities extracted from the text of a Semantic Web resource. Giving a resource as input the engine performs a similarity search returning the best similar resources. This process is described in Figure 7.

SWAs process the resulted data for presentation in a new form. This last part of the evaluation is dedicated to show how the visible results of a system's behaviour can be sustained by a formal description with Datanode. **Aemoo** is presented as a web application for exploratory search over the semantic web [12]. The input requested is the name of a semantic web entity (as string) and the output is a summarisation of the relevant knowledge

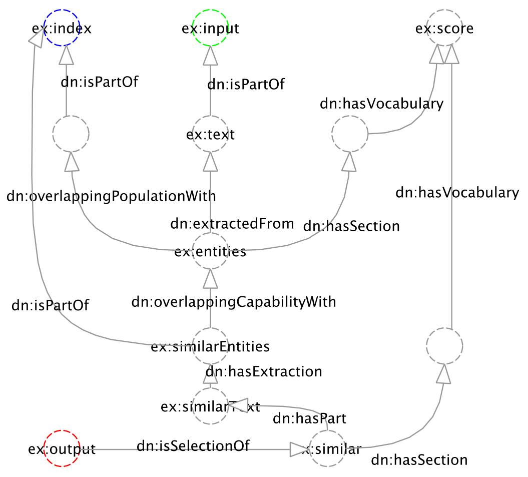

Figure 7: The similarity search process in DiscOU. This example shows the usage of derivation, partition and capability.

about the entity extracted from dbpedia and other linked data sources and arranged according a set of Knowledge Patterns [12] mapped to entity types. We can see in Figure 8 a description of the system according to Datanode. The output node `dn:resourceSummary` is `dn:isSummarizationOf` the input, `dn:remodelledFrom` a node which in turn `dn:isPortionOf` DBPedia. The output `dn:usesSchema` the knowledge pattern `dn:isDescriptionOf` the resource's primary type, a dn:isSelectionOf the types of the resource, from the set of dbpedia types. More precisely, the knowledge pattern is one of a set of summarization methods which are a `dn:combinationFrom` knowledge patterns and DBPedia types. It is clear that we have abstracted from the implementation of Aemoo, while we have now a better understanding of its output. Finally, the output is presented as `dn:isStandInOf` the data about the input resource.

Rexplore "*integrates statistical analysis, semantic technologies, and visual analytics to provide effective support for exploring and making sense of scholarly data*" [18]. Figure 9 shows a diagram of the visualised information about Topic's trends. Publication trends is the number of publications associated with a semantically enriched topic on a timeline; similarly, author trends is the number of associated authors. Migration trends is the number of estimated migrations between two topics computed by analysing the shifting in authors' interest.

IBM Watson is the well known computer that became famous after winning the *Jeopardy!* contest. Figure 10 shows the data flow of the system as it is described in the Wikipedia page [19].

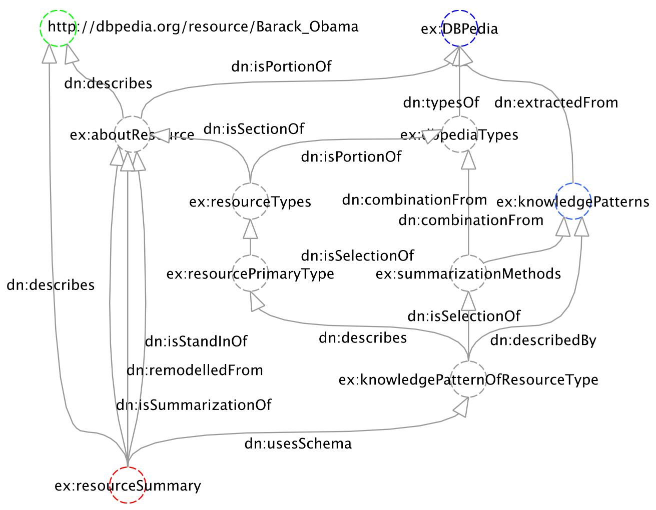

Figure 8: Description of AEMOO [12]. The input is a resource described in DBPedia. The system outputs a resource summary. Used branches: meta, reference, derivation, partition, vocabulary.

We have been able to express with Datanode all the data flows of the Semantic Web Applications collected. The following table shows the coverage of the Datanode ontology branches on each application considering not only the descriptions reported here but all the use cases evaluated:

	EventMedia	Yokohama Art Spot	DBRec	Spud	DiscOU	Aemoo	Rexplore	IBM Watson
meta						x	x	
reference	x			x		x		
derivation	x	x	x	x	x	x	x	x
adjacency				x	x			
partition	x	x	x	x	x	x	x	x
vocabulary	x				x	x		
version				x				
update			x					
consistency			x					
capability	x		x	x	x			x

6 Conclusions

In this paper we devised a conceptualisation of data flows within Semantic Web Applications which is based on describing relationships between datanodes. This abstract conceptualisation, implemented in an ontology was designed based on the analysis of several Semantic Web Applications of various origins and was evaluated by showing that it can be used to adequately capture data flows implemented by these applications and others.

We know from the literature [10] that understanding a software system is

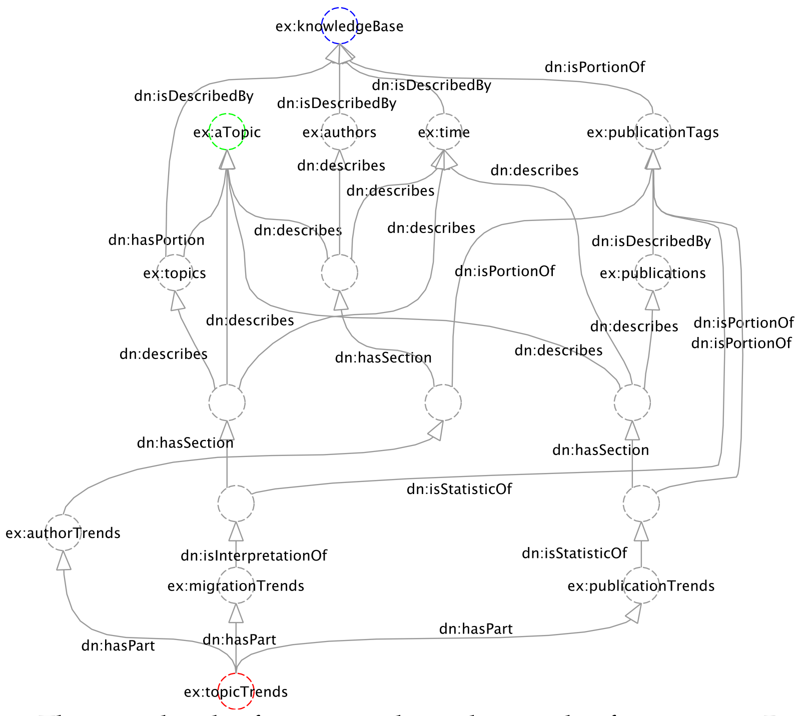

Figure 9: The visualised information about the trends of `:aTopic` in Rexplore. The web page includes publication trends, author trends and migration trends. Uses: meta, derivation, partition.

not achievable without choosing a certain facet as focus of the analysis. We choose the "data" to be our "ideological perspective" because we observed Semantic Web Applications and saw that manipulating data is the facet they had in common. As stated several times in the paper, the Datanode ontology and the class Datanode are voluntarily abstract. As a result, often the same system could have been described in different ways, all perfectly valid. This distinctive aspect of each representation - be the result of intentional choices - is important because it formalises and makes explicit the intention that system's authors put in their descriptions, and which is often hard to document differently than in natural language.

We focused here on improving the understanding of the output of Semantic Web Applications. It can be argued that the same approach could be tested by describing a wider range of applications, for instance in the context of sensor networks or data intensive systems.

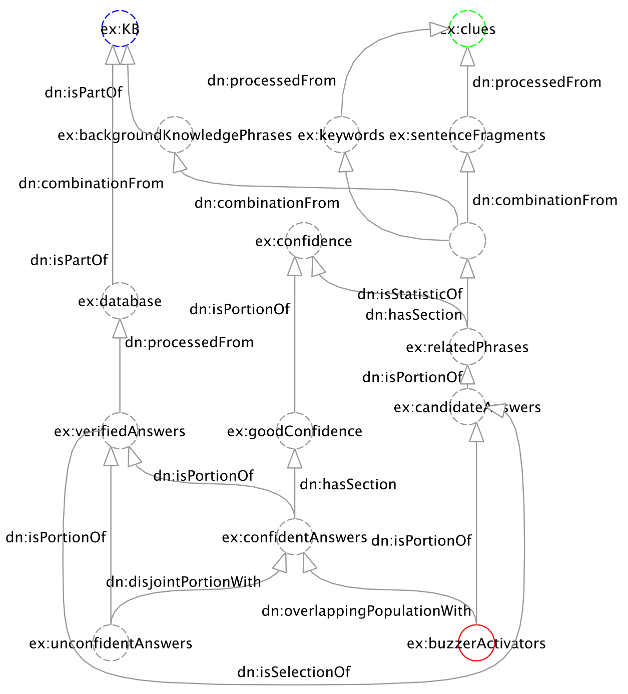

Figure 10: Watson at *Jeopardy!*, from the “clues” to the set of answers “buzzer activators”. This has been designed from the description on Wikipedia [19]. Uses: derivation, partition and capability.

References

1. Oren, E.: Algorithms and Components for Application Development on the Semantic Web. PhD thesis, Nat. Univ. of Ireland, Galway (2008)

2. Hartig, O., Zhao, J.: Publishing and Consuming Provenance Metadata on the Web of Linked Data. In: Provenance and Annotation of Data and Processes. Third International Provenance and Annotation Workshop (IPAW), Troy, NY, USA, June 15-16. LNCS Volume 6378, pp 78-90, Springer (2010)

3. Latif, A., Hoefler, P., Stocker, A., Ussaeed, A., Wagner, C.: The Linked Data Value Chain: A Lightweight Model for Business Engineers. In: Proceedings of International Conference on Semantic Systems, pp. 568-576, Graz, Austria, 2-4, Sep. 2009. (2009)

4. García-Castro, R., Gómez-Pérez, A., Muñoz-García, Ó., Nixon, L. J. B.: Towards a Component-Based Framework for Developing Semantic Web Applications. In: The Semantic Web, LNCS Volume 5367, 2008, pp 197-211, Springer (2008)

5. van Harmelen, F., ten Teije, A., Wache, H.: Knowledge Engineering rediscovered: Towards Reasoning Patterns for the Semantic Web. In: Proceedings of The Fifth International Conference on Knowledge Capture, page 81–88. ACM (2009)
6. Belhajjame, K., Corcho, O., Garijo, D., Zhao, J., Missier, P., Newman, D.R., Palma, R., Bechhofer, S. and others: Workflow-Centric Research Objects: A First Class Citizen in the Scholarly Discourse. In: Proceedings of the ESWC2012 Workshop on the Future of Scholarly Communication in the Semantic Web (SePublica2012), Heraklion, Greece (2012)
7. Lenzerini, M.: Data integration: A theoretical perspective. In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp 233-246, ACM (2002)
8. Pedrinaci, C., Domingue, J., Sheth, A. P.: Semantic Web Services. In: Handbook of semantic web technologies, pp 977–1035 Springer (2011)
9. LePhuoc, D., Parreira, J. X., Hauswirth, M.: Linked stream data processing. In: Reasoning Web. Semantic Technologies for Advanced Query Answering. Proceedings of the 8th International Summer School 2012, Vienna, Austria, September 3-8, 2012. LNCS Volume 7487, 2012, pp 245-289, Springer (2012)
10. Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., Stafford, J.: Documenting software architectures: views and beyond. 2nd Ed., Pearson Education (2010)
11. Presutti, V., Blomqvist, E., Daga, E., Gangemi, A.: Pattern-Based Ontology Design. In: Ontology Engineering in a Networked World, pp 35-64, Springer (2012)
12. Musetti, A., Nuzzolese, A. G., Draicchio, F.,Presutti, V., Blomqvist, E., Gangemi, A. and Ciancarini, P.: Aemoo: Exploratory search based on knowledge patterns over the semantic web. In: Semantic Web Challenge, 10th International Semantic Web Conference, Springer (2011)
13. Kotoulas, S., Lopez, V., Lloyd, R., Sbodio, M. L., Lecue, F., Stephenson, M. and others: SPUD: Semantic Processing of Urban Data. In: Semantic Web Challenge, 11th International Semantic Web Conference, Springer (2012)
14. Passant, A.: Dbrec - music recommendations using DBpedia. In: The Semantic Web - 9th International Semantic Web Conference, Springer (2010)
15. d'Aquin, M., Allocca, C., Collins, T.: DiscOU: A Flexible Discovery Engine for Open Educational Resources Using Semantic Indexing and

Relationship Summaries. In: 11th International Semantic Web Conference. Posters & Demos. (2012)

16. Matsumura, F., Kobayashi, I., Kato, F., Kamura, T., Ohmukai, I., Takeda, H.: Producing and Consuming Linked Open Data on Art with a Local Community. In: Third International Workshop on Consuming Linked Data (COLD2012) in conjunction with the 11th International Semantic Web Conference. Boston, MA, USA, November 12, 2012. `http://ceur-ws.org/Vol-905/` (2012)

17. Khrouf, H., Troncy, R.: EventMedia: A LOD dataset of events illustrated with media. In: Submitted to Semantic Web Journal, Special Issue on Linked Dataset descriptions, 2012 `http://www.semantic-web-journal.net/content/eventmedia-lod-dataset-events-illustrated-media`

18. Osborne, F., Motta, E., Mulholland, P.: Exploring Scholarly Data with Rexplore. In: The Semantic Web – ISWC 2013. 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I Series: LNCS in Computer Science, Vol. 8218 pp 460-477. Springer (2013)

19. Wikipedia page for Watson_(computer): `http://en.wikipedia.org/wiki/Watson_(computer)`