Using Semantic Web Technologies to Support Information Processing and Coalition Decision-Making

Dave Braines*††, Paul R. Smart†, Jie Bao‡, Alistair Russell†, Nigel R. Shadbolt†, and James Hendler‡

*Emerging Technology Services, IBM United Kingdom Ltd, Hursley Park, Winchester, SO21 2JN, UK
†School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK
‡Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

††Corresponding author: dave_braines@uk.ibm.com

***Abstract*—This paper outlines three techniques and tools which have recently arisen from Semantic Web research in the International Technology Alliance. Each of these has been previously introduced in conference publications during 2008 but the purpose of this paper is to draw them together, explain their inter-relationships and to explain how they can be used collectively to support one of our fundamental research contexts: Coalition information processing and decision making. This paper provides the reader with a detailed overview of each along with a view of how they relate to one another and how each may better enable the usage of Semantic Web technologies to facilitate information integration and fusion in a distributed network context amongst disparate (coalition) communities with converged, but not necessarily aligned, goals. The first of these techniques is *POAF* (Portable Ontology Aligned Fragments) which addresses issues relating to the portability and usage of ontology alignments. *POAF* uses an ontology fragmentation strategy to achieve portability, and enables subsequent usage through a form of automated ontology modularization. The second technique, *SWEDER* (Sematic Wrapping of Existing Data sources with Embedded Rules), is grounded in the creation of lightweight ontologies to semantically wrap existing data sources, to facilitate rapid semantic integration through representational homogeneity and embedded rules. The third is *NITELIGHT* which is a tool that has been created to better support end users with respect to the creation and editing of semantic queries. *NITELIGHT* uses a visual query language which we have proposed, named vSPARQL and, as the name suggests, it is based on the W3C SPARQL query language specification. *NITELIGHT* supports end users by providing a set of graphical notations that represent semantic query language constructs enabling the user to combine ontology navigation capabilities with graphical query visualisation techniques.**

I. Introduction

Research sponsored by the International Technology Alliance[1] (ITA) into semantic interoperability and alignment techniques has recently yielded a number of promising techniques, three of which are described in this paper. These address different, but important, sets of challenges in the domain of semantic integration, with a view to easing issues of information processing and subsequent decision making within distributed network environments. Much of the material in this paper is drawn from existing publications that have separately introduced each of these three techniques at various conferences [1–5] during 2008. The specific purpose of this paper is to draw all three together, explain their inter-relationships and describe how they can be used to support information processing and decision making in a distributed coalition context.

The specific operational context for our ITA research has been that of a coalition of military, non-government organisation (NGO) and local force alliance members working together in a distributed network environment. Wherever possible our ITA research is based on de-facto Semantic Web standards and existing technology components.

The first of these techniques is *POAF* (Portable Ontology Aligned Fragments) [1] which addresses issues relating to the portability and subsequent usage of ontology alignments. *POAF* uses an ontology fragmentation strategy to achieve portability with subsequent usage supported through a form of automated ontology modularization.

The second technique is *SWEDER* (Semantic Wrapping of Existing Data sources with Embedded Rules) [2] and is grounded in the creation of lightweight ontologies to semantically wrap existing data sources in order to facilitate rapid semantic integration through representational homogeneity. One important form of semantic integration can be achieved through the creation of *context ontologies* which are built upon the *SWEDER* technique and which have the specific purpose of defining the integrations and providing a portable definition of the rules to achieve these integrations in the form of SPARQL[2] construct clauses.

The third technique is manifest in the form of a tool named *NITELIGHT* citeRussell08 and a visual query language named vSPARQL which is based on the W3C SPARQL semantic query language. *NITELIGHT* supports end users by providing a set of graphical notations that represent semantic query

[1]More information available from: http://usukita.org/

[2]http://www.w3.org/TR/rdf-sparql-query/

language constructs enabling the user to combine ontology navigation capabilities with graphical query visualisation techniques.

The main content of this paper is found in sections II, III and IV which give an overview of *POAF*, *SWEDER* and *NITELIGHT* respectively. These sections consolidate materials from the previous publications [1–3], introducing each of the techniques and giving an appropriate overview of the details. Section VI gives details of other work which is related to the *POAF* and *SWEDER* techniques, and the paper is concluded in section VII.

There is no specific worked example in this paper as each of the three techniques described has extensively defined a worked example in each of the previous publications. For example: the *POAF* technique describes a semantic integration and alignment example based on fusing existing terrorist incident ontologies around the shared concept of City [1]; the *SWEDER* technique describes a simplistic information integration example using common data to be found in email, organisation and social network data sources /citeBraines08a; the *NITELIGHT* tool explores the construction of queries in the domain of terrorist organisations /citeSmart08; and both *POAF* and *SWEDER* are described together in a worked example fusing the semantic integration of terrorist incident data with the non-semantic world gazetteer data regard city populations /citeBraines08b. To help define the particular types of integration task to which each of these technique is best suited a "typical usage" subsection is included for each. This describes the benefits and any shortcomings which apply to the technique, the ideal circumstances for usage and any other relevant information.

II. POAF

POAF (Portable Ontology Aligned Fragments) is the first of three techniques described in this paper:

A. Introduction

A key concern in our domain of interest relates to the physical distribution and semantic heterogeneity of relevant information content: physical distribution makes information difficult to search, retrieve and manage, while semantic heterogeneity makes information difficult to integrate and understand.

Semantic integration and interoperability solutions have been studied and applied to a variety of domains [6] but coalition operations impose strict availability and access constraints on knowledge assets. There is a perceived operational time frame associated with each operation and any semantic integration solution that aims to have an impact must respect this. In today's semantic integration research, time constraints and availability of knowledge assets are not a high priority. To bridge this gap, we set out to explore the use of advanced semantic integration solutions, like ontology mapping [7], which have gained a lot of attention and momentum in recent years[3].

[3]http://www.ontologymatching.org/ for an up-to-date overview of the field

One of the issues with ontology mapping is the relatively sparse and problematic uptake of ontology mapping products: ontology alignments. Despite the advances in describing alignments in rich[4] and standardized ways for programmatic access [8], and storing and sharing alignments [9], the use of alignments in applications remains sparse.

One of the reasons for this phenomenon is the relatively high computational cost for processing ontology alignments. The W3C's[5] omnipresent `owl:sameAs` statement is only a starting point for what turns out to be a high computational cost and time consuming reasoning task to be undertaken by a Description Logic (DL) reasoner. That is because the `owl:sameAs` statement merely indicates the logical equivalence of the two referenced terms, but does not provide any further information about their provenance and semantics. To take full advantage of this logical equivalence we need to consider more than just the reporting of this fact.

B. Operational Ontology Fragments

Our experiences from ontology mapping [10] show that only a small fraction of the referenced ontologies are used in the produced alignments. This observation opens the way for an ontology alignment informed fragmentation task that meets the strict operational constraints needed in a distributed coalition context. In this scenario the ontology alignments become the focal point, both as enablers of semantic integration solutions as well as triggers for fragmenting the original knowledge assets into meaningful and semantically coherent chunks of knowledge. These semantically coherent fragments will mirror the relevant portions of the original ontologies but they are smaller in size and complexity, thus easier to process in a operationally useful time frame.

As an alternative some proposals to overcome the high computational cost and high latency in performing an inference cycle involve fragmentation or modularization of the original ontologies. Others approaches consider enhancing the existing OWL[6] vocabulary with richer constructs that enable more provenance information and semantics to be exposed when logical equivalence is reported. These however are heavyweight approaches that pose certain assumptions on the domain and applications.

We advocate a practical solution to this problem which is ready to use with current technology: to use existing alignments and OWL taxonomic reasoning to identify fragments from the original ontologies that capture the immediate provenance and semantics of the aligned terms. We then propose to extract those fragments using standard W3C rule and query technology that is easy to reuse and replicate in different scenarios. The extracted fragments are bundled together in self-contained and well defined portable OWL fragments without affecting the original ontologies. These fragments can then be accessed and re-used at a lower computational cost than that of accessing and re-using the original ontologies.

[4]e.g. the SKOS vocabulary: http://www.w3.org/TR/skos-reference/

[5]The World Wide Web Consortium - http://www.w3.org/

[6]http://www.w3.org/2004/OWL/

In the next section we describe a novel mechanism for extracting those fragments from ontologies using ontology alignments as the trigger.

C. The POAF Process

POAF aims to increase the usability, tracking of provenance, and portability of ontology alignment products (typically a number of `owl:sameAs` statements) among interested parties. It is a post ontology alignment process. We assume that an ontology alignment or mapping tool has been executed and has delivered a set of ontology alignments using the W3C's standard notation: `owl:sameAs`, `owl:equivalentClass` and `owl:equivalentProperty`. These notations enable us to indicate logical equivalence of two OWL constructs (ranging from individuals to classes and properties, depending on the type of OWL language used). When automated reasoners encounter `owl:sameAs` or similar statements they make use of that information and access the aligned terms in order to complete their reasoning process. However, there are issues with this approach which call for a lighter and more efficient way of sharing aligned terms:

- Availability and access of the aligned ontologies:
  If the ontologies where the `owl:sameAs` referenced terms belong are not accessible (i.e. due to network outage, bandwidth restrictions or interference concerns in an operational context), or not available at the time the reasoner tries to conduct its inference cycle, this could cause a break in the reasoning and consequently bring the inference process to a halt, causing the reasoner to abandon this task (some advanced DL reasoners could resume at another point but the particular inference will not be concluded);
- Unnecessary reasoning steps:
  When a reasoner visits the ontologies where the aligned terms originate it is likely that unnecessary crawling of the OWL ontologies will occur (depending on the type of reasoning task). Even if the task is to simply resolve the name of the aligned concept, visiting the originating ontologies will add unnecessary time to the processing task;
- Fragmented and distributed knowledge base:
  When a reasoner tries to perform a task where multiple `owl:sameAs` statements are involved and point to a number of different ontologies, the reasoner will have to collate information from different URI addresses. Although today's DL reasoners can cope with this task, it is an unnecessary resource load for the system and could affect its performance;
- Difficult to track provenance information:
  When a number of `owl:sameAs` statements are used to convey logical equivalence for the aligned concepts, it is difficult to track their provenance. This becomes increasingly problematic as more and more ontologies are involved. At the very least it is often not feasible to inspect the origin of the aligned concepts using eyeball checking in a casual fashion. It is likely that an engineer will need to employ a reasoner to do this task, which brings us back to the problems mentioned before: those of unnecessary reasoning steps, overload of time and bandwidth resources, and knowledge base fragmentation.

Based on these observations, and our experiences with designing, developing, deploying and using ontology mapping systems [10, 11], we propose a lighter and more portable way for sharing ontology alignment information. We propose a mechanised way for extracting fragments from the underlying ontologies using ontology alignment information as a trigger. We dub these fragments *POAF* (Portable Ontology Aligned Fragments) to highlight the tight coupling of the ontology alignment and the generated fragment. As the name indicates, we are interested in fragments of ontologies and, in particular, we aim to make those fragments portable. By that we mean:

1) Capture a fragment of each ontology that is directly relevant to the aligned term;
2) Use the aligned term, and OWL semantics, to identify and capture those fragments;
3) Bundle these fragments in OWL compatible snippets that reasoners can use automatically as if they were mini-ontologies;
4) Include appropriate provenance information in these fragments, so that tracking and tracing of the original ontology is feasible.

These steps have been implemented in a process which operationalises the idea of *POAF* (See Figure 1).

The first step of the *POAF* generation process involves the *capture fragment* task: we use OWL semantics and the ontology subsumption relations to identify related fragments. These are fed into the second step, *Use W3C technology to extract fragment*, where the actual extraction occurs. We opted for W3C technology so that we increase our interoperability potential with other tools and technologies. We use SPARQL queries, SWRL[7] rules, and the Jena[8] Semantic Web framework from Hewlett Packard Labs. The next step is the *Bundle fragment* task. The aim of this task is to construct well formed OWL fragments so that DL reasoners and external tools can use them as a substitute for the original ontologies, depending on the task and scope of application. Finally, the last step is to provide as much provenance information as possible in the *Provenance aggregator* task. In this task we collate the original namespaces with *POAF* specific ones to distinguish between declared and inferred statements in the *POAF* files.

The following section provides an overview of the second technique covered by this paper: *SWEDER*.

D. Typical Usage

From the earlier description it is apparent that *POAF* is a useful technique for semi-automatically creating self contained mini-ontologies which represent the specific alignment(s) between two or more other existing ontologies. *POAF* does not perform ontology alignment itself and is only useful in

[7] http://www.w3.org/Submission/SWRL/
[8] http://jena.sourceforge.net/

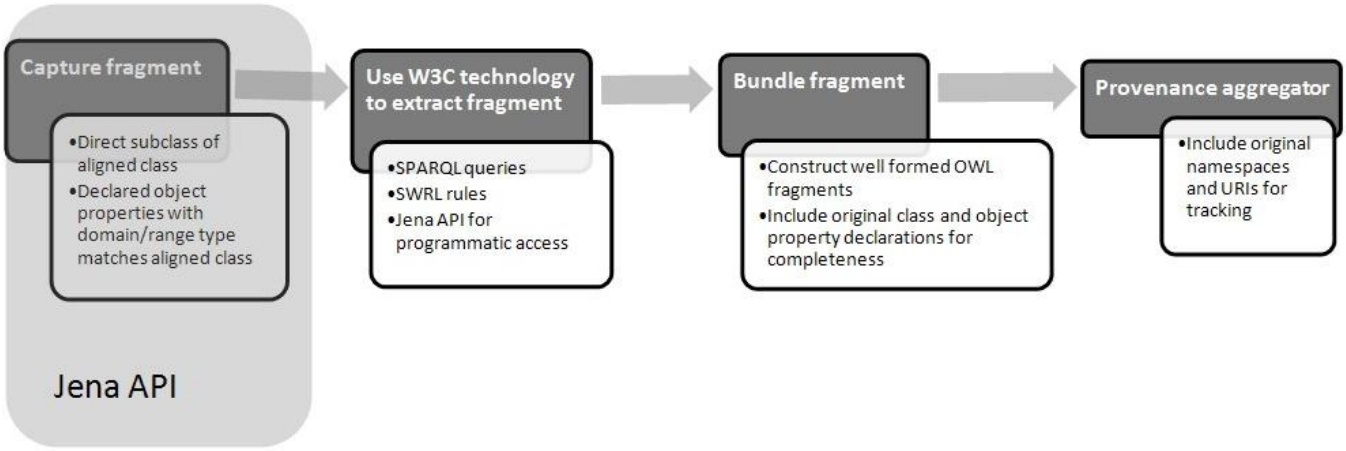

Fig. 1. The POAF process

cases where alignments have already been automatically or manually defined. The stand alone *POAF* ontologies define the core entities that are involved in the alignment and any immediately related information. Further entities within the source ontologies are not contained within the *POAF* ontology as they are deemed to be "out of scope" of the specific alignment context, although references to the source ontologies is retained in the *POAF* ontology to support navigation back to the full details if necessary. This approach enables *POAF* to be of great use is environments where data transmission rates are poor or severely constrained as it means that only the minimum amount of information needs to be passed around the network. It also better enables distributed network processing of ontologies since each *POAF* ontology can be self contained and processed in its own context rather than relying on every consuming application loading the source ontologies which may be very large and located remotely, potentially even in an inaccessible location. So, *POAF* is an ideal technique to be used in distributed network contexts where existing ontologies must be aligned and those alignments used and reasoned over. *POAF* is particularly useful in cases where the source ontologies are extremely large and only a very small fraction of the concepts are of direct relevance to the alignment task (e.g. upper ontologies) but it is also useful in less extreme cases where the source ontologies may be smaller and more domain focused. *POAF* is less useful in cases where no semantic representation of the existing data already exists since it inherently depends on the presence of ontology alignment semantic statements such as `owl:sameAs` but if the non-semantic data sources have been semantically wrapped (e.g. using the *SWEDER* technique) and then aligned using an appropriate alignment tool of technique then *POAF* can be successfully applied. *POAF* does not require any instance data to be defined for the ontologies which are being aligned, although it is assumed that the reasoning which may be performed using the alignments will be based on specific sets of instance data.

III. SWEDER

SWEDER (Semantic Wrapping of Existing Data sources with Embedded Rules) is the second of three techniques described in this paper:

A. Introduction

A plethora of electronic data is available in structured forms today, either in existing Semantic Web encodings such as OWL/RDF or, more likely, in more traditional formats such as XML, CSV, HTML or relational databases. This data is available to the consumer, usually via URIs resolving to network or local addresses, but may also be sourced from directly referenced files and other non-URI referenced resources. The data itself can take any form, but we propose that it can be relatively easily semantically wrapped through a lightweight application of OWL/RDF to represent this existing data in the form of *entities* (classes), *attributes* (data properties) and *relationships* (object properties). The purpose of this lightweight semantic wrapping of these existing data sources is to provide representational homogeneity, thereby providing a common semantic basis for any downstream usage of this data.

Details regarding how this semantic wrapping is achieved are not discussed here since the focus of *SWEDER* is on how to leverage the semantically wrapped data sources. After the semantic wrapping has occurred *context ontologies* can be created to specifically capture the alignments between these semantically wrapped data sources, or potentially with existing ontologies from other sources. Representational homogeneity can now be assumed for all data sources that have been semantically wrapped in this way. In practical terms this semantic wrapping will likely be achieved through manual design and construction of a small ontology followed by a transformation process to convert the existing instance data to the new ontology. A similar existing pattern for this work is that of RDFa[9], microformats[10] (and GRDDL[11]) which are

[9]http://www.w3.org/2006/07/SWD/RDFa/
[10]http://microformats.org/
[11]http://www.w3.org/2004/01/rdxh/spec/

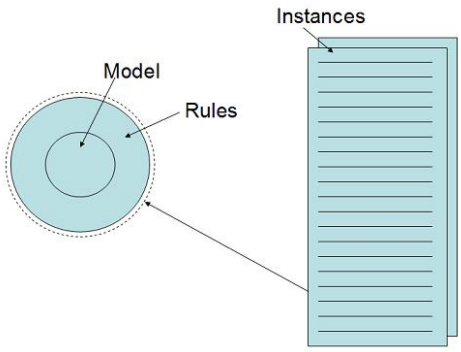

Fig. 2. A two-tier ontology and associated instances

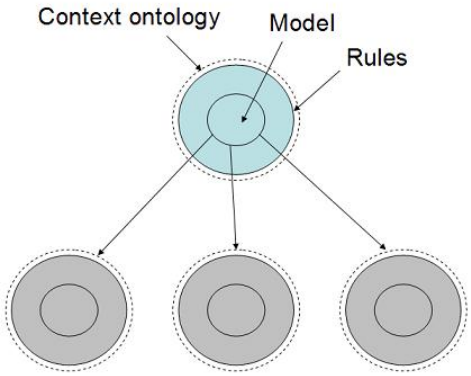

Fig. 3. A typical context ontology

increasingly popular techniques for semantically enriching existing web-based data sources, albeit within existing web markup languages rather than as stand-alone ontologies as we are proposing.

B. Context Ontologies

We adopt a dynamic notion of context which is not common to the formal notions presented in the AI literature (see, for example, the seminal work in [12] on formalizing contexts as first class objects). Our aim is to use context dynamically to capture each purpose for which data is used by a consumer application. It is assumed that the consumer application loads multiple data sources in order to fuse them or otherwise make use of these data sources and the relationships between them. We propose that the consumer application therefore adds a context to these data sources, since this specific combination of data sources has been selected to fulfil a particular need. *SWEDER* enables this context to be captured through the creation of a *context ontology* which specifically integrates the concepts from any semantically wrapped data sources or existing ontologies that it references. This *context ontology* can then be easily used by the consumer application, reading the various semantically wrapped data sources or existing ontologies, processing the instance data and executing embedded rules to derive further information or alignments. The result of this could be published as instance data conforming to the new *context ontology* and then made available for further usage by unknown downstream consumer applications.

Of course the fusion of two data sources by an application to achieve the results above could easily be achieved with existing technology, and does not require semantic representation. As a matter of fact, one could view the current trend of mashups[12] as a successful (usually non-semantic) form of such integrations. The specific benefits of semantically enabling the data sources and capturing the alignment representation and rules in a *context ontology* as defined in our approach lie in the representational homogeneity achieved through this technique, the self-defining and portable nature of the *context ontologies*, and the ease with which consumer applications can use them. Further important capabilities are also enabled through the use of this approach, most notably the support for referencing common definitions via URIs to enable more rapid understanding and information integration.

A key aspect of our proposal is facilitating the creation, representation and execution of information integration rules within these *context ontologies*, and this is something that existing OWL based solutions do not readily support. There are emerging standards in this area, notably SWRL (and potentially RIF[13]), but for various reasons we have chosen a lightweight, pragmatic approach and use SPARQL construct clauses to define and store these rules. This allows any SPARQL enabled endpoint to execute the rules and instantiate the inferred results directly from the construct clause without the need for any additional rule execution engine.

We store each SPARQL construct clause as instance data directly in the *context ontology* to which it applies, thus enabling these rules to be passed to the consumer application as part of the *context ontology*. In further iterations of this work we plan to introduce richer representation formats (such as SWRL, RIF) as these representations could generate SPARQL construct clauses which would be executed as per our current solution. The use of these richer representation languages would expose the semantics of these information integration rules to consumer applications rather than the current solution which simply records the text of the SPARQL construct clause without providing any actual representation of the rule composition.

C. A Conceptual Model

Conceptually and practically we use a two-tier model to represent each of our ontologies, both for the simple semantic wrappings of existing data sources, and for the subsequent *context ontologies* which are created to capture the alignments of other ontologies. This two-tier approach allows for a clear separation between the representation of the model and the capture of any associated rules. An example of this two-tier model is shown in figure 2.

[12]http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)

[13]http://www.w3.org/2005/rules/

The model ontology is at the centre, and it is comprised of entities, attributes and relationships, as described earlier. This is then imported into the outer rules ontology, which simply adds support for rules to be defined against the model. This two-tier approach is only needed when semantically wrapping existing data sources; existing ontologies can be used in their original form. In our current implementation the ability to define and store rules is provided through an import to a generic information integration rules ontology which enables the SPARQL construct based rules to be represented as simple entities, and records the subsequent instances of these rules as text in data-type properties. The separation of these two aspects of our ontologies enables the rules to be captured separately to the model, thus offering us a flexible way in which to improve the rule representation solution in the future without affecting the model ontology.

The final aspect of our solution is the capture of context information for multiple ontologies, which we achieve via the creation of additional lightweight *context ontologies*. These are built according to the same two-tier approach described previously.

The *context ontology* defines (in the inner model ontology) any additional entities, attributes or relations which are relevant to the current context, and also defines (in the outer rules ontology) any instances of rules which are used to populate these additional items. The *context ontology* also imports any required source ontologies (which may be existing ontologies, semantically wrapped data sources or *context ontologies*) and the new *context ontology* therefore captures the representation of this specific new context and embodies it in a semantic format consistent with the constituent ontologies. We visualize this approach in figure 3.

The final step is for a suitable consumer application to load the *context ontology* and any associated instance data for the source ontologies. Since all of the rules are contained within the *context ontology* this consumer application simply invokes a standard process to extract all these rules (which are stored as SPARQL construct clause text), then executes these rules against an appropriate SPARQL endpoint. The results of these rule executions are that new instance data are created within the *context ontology*, and this can then be saved, published or further processed by the consumer application. We depict diagrammatically the interaction with a consumer application in figure 4. Note - the consumer application actually executes all the rules multiple times, until the set of all rule executions results in no further data being inferred.

D. Typical Usage

SWEDER is most useful in cases where large volumes of structured or semi-structured electronic data already exists, but this data has not been semantically represented in the form of an ontology. The assumption that information integration is better achieved through semantic representation is made, resulting in a desire to efficiently convert this existing non-semantic electronic information into an appropriate semantic representation. *SWEDER* also applies to existing semantic data

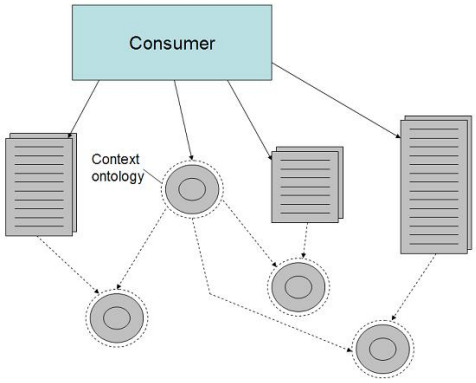

Fig. 4. A typical consumer application

both in terms of ontologies (models) and sets of instance data (most likely in RDF/RDFS), but this is perceived to be a less common occurrence in our specific domain of interest. *SWEDER* does readily support the integration of existing semantic data and newly converted semantic data, thus enabling the potentially common scenario of converting existing non-semantic data to a semantic form and then integrating with existing ontologies and semantic models. *SWEDER* proposes the creation of *context ontologies* to facilitate the extension of existing concepts and the creation of new concepts. These context ontologies also support the creation of rules which are able to be saved in the form of SPARQL construct clauses. This enables these rules to be passed around with the rest of the model, and they are represented in a form specifically designed to be easy to execute so that any SPARQL-capable application can load the model and execute the rules. *SWEDER* is therefore ideal in situations where there is a great deal of existing data, where there is a desire to model richer rules than can be represented in the OWL language, and again in the distributed network environment where reasoning may occur at any point in the network on any set or sub-set of data. It is also a useful technique to support a proliferation of similar but distinct contexts on various combinations of existing data sources which is a common situation in real world communication of understanding. For example two people may have two genuinely different perspectives on how the same two existing data sources related. Each of these perspectives can be modeled as a separate context using the *SWEDER* approach and the consumer can then choose which of the existing contexts (if any) is useful to thin their context. *SWEDER* is not useful in cases where there is no instance data available, or in cases where there are existing ontologies and existing sets of corresponding instance data that has already been successfully aligned.

IV. NITELIGHT

NITELIGHT is the last of the three techniques described in this paper:

A. Introduction

Efficient information retrieval is key to enabling the Semantic Web both in terms of efficient application design and end user experience. The W3C endorsed semantic query language (SPARQL) is a well defined and expressive query language which is now widely supported in semantic web frameworks and tooling. Query developers, and potentially end users, are likely to benefit further from a richer and more interactive environment within which semantic queries can be constructed, tested and executed. *NITELIGHT* requires a set of graphical notations in the form of a visual query language that relates to all or part of the SPARQL specification, and we also describe our proposal for this in the form of vSPARQL.

B. vSPARQL

The development of a graphical tool for SPARQL query formulation necessarily entails the development of a set of graphic notations that support the visual representation of SPARQL query components. Following an analysis of the SPARQL syntax specification [27], we developed a set of graphical notations to support the representation of SPARQL queries. These notations comprise the basis of a SPARQL VQL that we refer to as vSPARQL. In the first half of this section we present some features of this language based on our work to date. The graphical query designer, *NITELIGHT*, was designed to support the user with respect to the formulation of SPARQL queries using vSPARQL constructs. The second half of this section describes the functionality and user interaction features of the *NITELIGHT* graphical editor.

- Core SPARQL Features:
  Because SPARQL queries exploit the triple-based structure of RDF models, graph-based representations comprising a sequence of graphical nodes and links can be used to represent the core of most SPARQL queries. The nodes in this case correspond to the subject and object elements of an RDF triple, while the links correspond to the predicates. vSPARQL uses colour to differentiate between the three types of graphical node (i.e. Bound Variable Node, Unbound Variable Node and Non-Variable Node) that are used by vSPARQL to represent the subject or object elements of a triple (see figure 5). Bound Variables, in this case, represent variables whose value bindings are returned as part of the query resultset, Unbound Variables are variables that are not returned in the resultset (they are used as part of the query execution process) and Non-Variable Nodes are nodes that represent a URI, literal value or blank node. Nodes are associated with a label that indicates the URI, literal value or query variable represented by the node. The predicate part of a triple is visually represented by a graphic link between the subject and object nodes. As with the graphic objects representing the subject or object parts of the triple, the graphical object representing the predicate is associated with a text label that specifies either the URI of the predicate or the query variable (see figure 5).

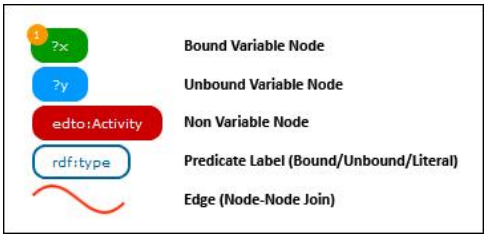

Fig. 5. Core vSPARQL graphical notations

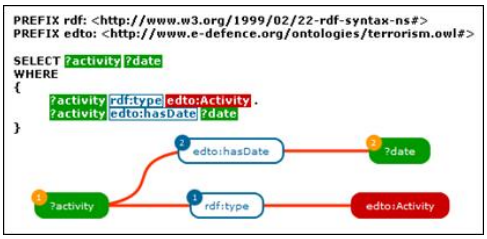

Fig. 6. vSPARQL representation of a SPARQL SELECT query

- Triple Patterns:
  The fundamental component of a SPARQL query is the triple pattern. Collections of triple patterns within a query are matched in sequence against the target RDF model in order to establish variable bindings and return query resultsets. Graphically, a triple pattern can be represented by a subject node connected to an object node by a predicate link. Subject and object nodes within the triple pattern are identified by their connection with the Predicate Label: a graph edge protruding from the right hand side of a node into the left hand side of the Predicate Label is the subject. of the RDF triple; a graph edge protruding from the left hand side of a node to the right hand side of the Predicate Label is the object. of the RDF triple.
- Simple Select Query:
  In vSPARQL a SELECT query comprises graphical representations of the triple patterns that are ultimately matched against the target RDF model (see figure 6). The query variables that are returned as part of the SELECT query are represented by the Bound Variable Nodes (coloured green), while the query variables that are used internally as part of the vSPARQL query are represented by Unbound Variable Nodes (coloured blue, but not shown in figure 6). The order in which Bound Variables are returned in query resultsets can sometimes be important. This ordering information is represented in vSPARQL using a numeric value in an orange circle added to the top left of the (Bound or Unbound) Variable Node. The order in which triple patterns appear within the SPARQL WHERE clause is defined by a similar order indicator on the label associated with predicate link (see figure 6).
- Graph Patterns:
  In SPARQL, there are multiple types of graph patterns

(e.g. basic graph pattern, group graph pattern, etc.). The query presented in figure 6 is an example of a basic graph pattern that comprises one or more triple patterns. Graph patterns influence variable bindings because each variable has local scope with respect to the (basic) graph pattern in which it is contained. This means that the same variable could be bound to different values in different graph patterns. In SPARQL, a group graph pattern is a collection of two or more basic graph patterns. Graphical support for the representation of group graph patterns in vSPARQL is accomplished by grouping triple patterns into separate graphical groups. Two further types of graph pattern are encountered in SPARQL: optional graph patterns and union graph patterns. Optional graph patterns, as their name suggests, are optional; they allow a user to extend the query solution with respect to additional triple patterns that may or may not match against the RDF model. Union graph patterns (or alternative graph patterns) allow a user to specify alternatives for graph pattern matching. In this case, one of several graph patterns may match the target graph; the failure of one graph pattern to match successfully will not necessarily result in the failure of the query, as a whole, to return a solution. Optional graph patterns are represented in vSPARQL by graphically grouping triple patterns and assigning a unique colour (brown) to the group. Union graph patterns are represented using a graphic link between two graph patterns. The specification of a default RDF graph, or the retrieval of a graph as part of a query, is represented in vSPARQL by using a link to a (Bound/Unbound) Variable (graph retrieval/specification) Node or Non-Variable Node (graph specification).

- Solution Sequence Ordering:
  In SPARQL, the ORDER BY clause establishes the order of a solution sequence, i.e. the order in which the elements of the query resultset are returned. A direction indicator (either Ascending or Descending) specifies whether the query resultset should be ordered in an ascending or descending sequence with respect to the relevant ordering variable. In vSPARQL, solution sequencing is realized by the use of an arrow icon within a (Bound/Unbound) Variable Node. The arrow icon uses a numeric value to indicate the order in which variables will be evaluated with respect to the ORDER BY clause; the direction of the arrow specifies the order direction, Ascending (up) or Descending (down).
- Filtering:
  SPARQL filtering is used to restrict the resultsets returned by a query using a variety of expressions, e.g. SPARQL operators, SPARQL functions and XPath casting functions [27]. The visual representation of a filter expression is based on the addition of Filter Field Boxes. beneath (Bound/Unbound) Variable Nodes. Because of the complexity of some SPARQL filters expressions, it is not always practical to display all the terms of the filter expression in the Filter Field Box. Instead, the Filter Field Box displays a short summary of the filter expression which is subsequently expanded in the *NITELIGHT* tool using a tooltip display mechanism.

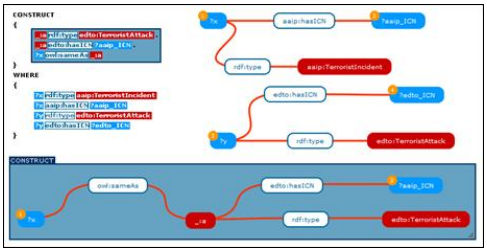

Fig. 7. vSPARQL representation of a SPARQL CONSTRUCT query

- SPARQL CONSTRUCT Queries:
  SPARQL has a number of query forms, namely SELECT, CONSTRUCT, ASK and DESCRIBE [27]. All the examples we have encountered so far are of the SELECT query form variety. CONSTRUCT queries are different from SELECT queries because they define both a set of triple patterns to match against the RDF graph, as well as a template for RDF graph construction. The RDF graph generated as a result of query execution is formed by taking the values of variable bindings associated with the triple patterns (in the WHERE clause) and substituting these into the RDF graph template (see [27] for more details). In vSPARQL, when a CONSTRUCT query is created, the graph pattern that comprises the graph template is highlighted using a colored box (blue in figure 7). This distinguishes the graph template from graph patterns specified as part of the WHERE clause (Figure 7).
- Other SPARQL Features:
  There are some features of the SPARQL specification that do not easily lend themselves to a visual representation. These features are supported in the *NITELIGHT* tool, but they are not part of the vSPARQL specification. They include, ASK and DESCRIBE query forms, as well as DISTINCT, LIMIT and OFFSET solution modifiers.

C. The NITELIGHT Editor

To test and evaluate the features of vSPARQL, we developed a Java-based prototype application, called *NITELIGHT*, using a combination of Jena [26] and Standard Widget Toolkit (SWT) components. *NITELIGHT* (see figure 8) provides 5 distinct components, each of which works together to give the user an intuitive interface for graphical query creation. The centerpiece of the *NITELIGHT* tool is the Query Design Canvas. The functionality of this component is supplemented by an Ontology Browser component, a SPARQL Syntax Viewer, a Query Results Viewer and a Quick Toolbar.

- Query Design Canvas:
  The Query Design Canvas is the centerpiece for user interaction and query construction in the *NITELIGHT* tool. It provides a canvas for the graphical rendering of SPARQL queries using vSPARQL constructs. It also includes a number of user interaction features that allow

users to create and refine semantic queries. Triples are drawn as two polygon nodes joined with a single link. To allow for more complex queries, the polygon nodes can be moved around the canvas freely, and the canvas itself can be zoomed and panned to view the entire query at different levels of visuo-spatial resolution. Both the nodes and links are selectable objects that can be edited using either the Quick Toolbar or a context menu. Both the Quick Toolbar and the context menu allow users to define filtering, ordering and grouping information for the selected object. The support for defining filter expressions is currently limited, consisting of a simple text entry form. Our future development plans aim to provide better support for filter expression definition, perhaps using a wizard-like utility.

- Ontology Browser:
  To facilitate the process of query formulation, and to provide users with a starting point for query specification, the *NITELIGHT* editor includes an Ontology Browser component. The first column of the Ontology Browser is a persistent list of currently loaded ontologies (the Source Ontologies Column). New ontologies can be loaded into the browser, and the selection of one of the loaded ontologies will result in the enumeration of top-level classes (root classes) in the second column of the Ontology Browser. The Ontology Browser consists of a series of columns that display the classes and subclasses of an ontology with more abstract classes situated to the left. The column immediately to the right of the Source Ontologies Column is always populated with the root classes of the currently selected ontology. Selecting a class from this column causes an adjacent column to appear to the right of the root classes column. This new column contains the subclasses of the currently selected root class. The pattern of subclass enumeration is repeated as the user progressively selects classes from the right-most column. The Ontology Browser also provides access to information about the properties associated with each class. In this case, the user can expand a class node in the Ontology Browser to view a list of properties associated with the class. The Ontology Browser enables a user to drag and drop classes and properties onto the Query Design Canvas. A new node can be created by dragging a class item from the Ontology Browser onto the canvas. A new link can be created by dragging a property from the Ontology Browser and attaching it to a node on the canvas.
- SPARQL Syntax Viewer:
  The SPARQL Syntax Viewer component provides a text-based view of the query that is dynamically updated to reflect any changes made using the Query Design Canvas. At the present time, the SPARQL Syntax Viewer is read-only, i.e. the user cannot edit the SPARQL syntax directly; they must implement any changes to the query via the Query Design Canvas. Future work could explore the possibility of bi-directional translation capabilities in which the user would be permitted to modify the graphical representation of a SPARQL query by interacting directly with the SPARQL Syntax Viewer. This would be of particular benefit to users who wanted to visualize existing text-based SPARQL queries for the purposes of query refinement or improved understanding.
- Query Results Viewer:
  The Query Results Viewer allows a user to execute a vSPARQL query against any SPARQL endpoint. In the current version of the tool the results are presented in the form of a simple table; however, one could imagine a variety of alternative output formats that might be more suited to the processing capabilities of human end-users. Examples include map-based visualizations, timelines and natural language serializations of query result sets. Since these output formats are often tied to a particular application context, we do not intend to explore the use of these richer visualizations as part of the current development effort.
- Quick Toolbar:
  The Quick Toolbar provides access to commonly used tools for manipulating the Query Design Canvas and its graphical query contents. Example tools include pan and zoom buttons, grouping functions and node editing utilities.

Fig. 8. The NITELIGHT tool main showing a graphical query and results

D. Typical Usage

NITELIGHT and vSPARQL are primarily aimed at providing a better and more usable interface to the construction of semantic queries through the use of visual design elements. As mentioned previously, this work is currently aimed at users who have a level of familiarity with the SPARQL query language, but perhaps require a more interactive and visual environment in which to operate. In the context of the other two techniques mentioned in this paper, we see *NITELIGHT* being especially valuable in terms of supporting the user in the construction of the SPARQL CONSTRUCT "rules" that can be embedded into *SWEDER* ontologies. This generally isn't just a case of a user sitting down to write the correct SPARQL CONSTRUCT rule, but instead is an iterative exercise to

query the existing models and instance data, refining the user's understanding of the data and models available, ultimately culminating in the creation of the SPARQL CONSTRUCT rule. This highly iterative and fundamentally investigative process is well supported in an environment such as that provided by *NITELIGHT*, although we have not yet empirically tested how well *NITELIGHT* performs at this task.

V. Usability and User Evaluation

At this stage we have not performed any user evaluation studies, and in the case of *POAF* and *SWEDER* we are continuing to refine and extend these techniques, and will consider providing some level of tooling or framework to help foster their adoption. Until we reach that stage we do not intend to carry out any formal usability testing of these techniques. In the case of *NITELIGHT* we have not yet undertaken any user evaluation studies, but we do aim to do so in the near future. Specific focus areas for evaluation include the general usability of the tool, the ability of the tool to support users with regard to query formulation and comparative analyses of the tool with other graphical (e.g. [23]) and non-graphical (e.g. [21]) query formulation interfaces. Of particular interest are proposed comparisons between *NITELIGHT*, SEWASIE [23], SPARQLViz [22], and iSPARQL [20]. Clearly, there are a number of dependent variables that might be assessed in the context of user evaluation studies. These include:

1) Syntactic Validity: the number of syntactic errors made during query formulation.
2) Query Accuracy: the extent to which the query returns the right information.
3) Query Comprehensibility: the level of comprehension attained by a user about a specific query.
4) User Satisfaction: subjective ratings of the users satisfaction with the tool.
5) Query Formulation Efficiency: the amount of time taken to formulate queries.

The initial evaluation of *NITELIGHT* will be based on our target user community (experienced SPARQL users). After this initial set of evaluations we may progress to test the perception of *NITELIGHT* as a tool to support the investigation and modeling of alignments in the form of SPARQL CONSTRUCT clauses, specifically in support of the *SWEDER* technique.

VI. Related Work

The papers which introduced *POAF* [1, 4], *SWEDER* [2, 4] and *NITELIGHT* [3, 5] describe in detail the various related work, but here we provide a summarized list for simplicity:

- Using ontology alignments to inform the syntax for expressing ontology modules [13]. To enable the inclusion of related content in the module, also used to enhance the syntax. This is different from the use of ontology alignment within *POAF* which is to identify the starting points for the fragmentation algorithm.
- Taxonomic reasoning is similarly deployed to *POAF* in [14]. The proposed modularization algorithm takes advantages of the subsumption checking of Description Logics (DLs) to identify self-contained pieces in the given ontology.
- Improving the expressivity of modular languages [15] through relaxing the subsumption dependencies between ontology constructs. Proposing new modular semantics or extensions to improve the functionality of `owl:import` whereas *POAF* uses only existing semantic representation constructs.
- A different notion of context that contribute information relevant to natural language understanding [16]. Each context is used to serve a different purpose, similar to *SWEDER*.
- Context has been used in [17] to aid in ontology elicitation whereby certain features of context dictate the primitive ontological constructs that will comprise an ontology.
- Information integration is the focus of [18] but unlike *SWEDER* the authors propose to use a special kind of context knowledge, namely assumption knowledge, which refers to a set of implicit rules about assumptions and biases that govern the source data. This is similar to our notion of rules that integrate information from semantically wrapped data but we apply them at a later stage.
- A naming convention scheme [19] which is based on a loose ontology that represents the notions of kind and super-kind. The aim is to ease data usability by providing a naming scheme that allows for classification of source data. In our *SWEDER* work we use properties and super properties found in the *context ontologies* to aid information integration and grouping.
- Graphical query formulation tools for the Semantic Web, such as: OntoVQL [24], SEWASIE [23], SPARQLViz [22], and iSPARQL [20].
- The visual query builder associated with the iSPARQL framework [20] has similar functionality to *NITELIGHT*. The iSPARQL Visual Query Builder supports the user in all SPARQL query result forms (i.e. SELECT, CONSTRUCT, etc.). It also supports the creation of optional graph patterns as well as UNION combinations of graph patterns in a manner similar to that described for vSPARQL. Despite these similarities, a number of differences exist between the iSPARQL Visual Query Builder and *NITELIGHT* tool [3].

VII. Conclusion

In this paper we have described three techniques to aid information processing in a distributed coalition network environment: *POAF*, *SWEDER* and *NITELIGHT*.

POAF (Portable Ontology Aligned Fragments) is a lightweight mechanism for sharing and distributing enhanced ontology alignment information. Making use of the generated alignments between two ontologies and exposing their immediately related terms in a well defined and concise OWL structure. *POAF* taps into the area of ontology fragmentation, but is not directly related to a fragmentation strategy. Rather,

it highlights the importance of distributing more information when reporting logical equivalence between two aligned terms. This information could enhance uptake and re-use of ontology alignments. *POAF* may also contribute to adoption of Semantic Web technologies in scenarios where operational time constraints and resource load are important considerations. A *POAF* based solution aims to reduce the time needed to process ontologically aligned structures as it encapsulates the aligned concepts and their immediately related concepts in small, portable, and easily manageable fragments.

POAF can also be used as a quality assessment tool for ontology alignment. Since the generated *POAF* structure exposes the related terms of the aligned terms (subclass and properties) this information can be used to semantically sanitize and validate the proposed alignment. This process could be manual or semi-automatic depending on the assessment task.

SWEDER (Semantic Wrapping of Existing Data sources with Embedded Rules) is a pragmatic approach to semantically enable existing sources of data, and then utilise multiple semantically enabled sources of such data through the creation of *context ontologies* to capture the specific rules and any new entities, relationships or attributes arising from this new context. *SWEDER* allows the storage of rules directly within the ontologies in such a way that they can be easily extracted and executed by a common capability within any consumer application, specifically through the usage of SPARQL construct clauses.

NITELIGHT is a graphical editing environment for the construction of semantic queries based on the SPARQL language specification. It is primarily intended for use by those with previous experience of SPARQL (although it could also potentially serve as a support tool for novice users who aim to acquire SPARQL expertise). *NITELIGHT* specifically supports an existing text-based query language; namely SPARQL. In contrast to the recommendations of some commentators [25] we do not propose to develop a simplified query language for end-users; rather we aim to support end-users with respect to the creation of complex queries using supportive user interfaces and user interaction mechanisms. Our tool is one of growing number that are being developed to support information retrieval in the context of the Semantic Web.

ACKNOWLEDGMENT

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] Y. Kalfoglou, P. Smart, D. Braines, N. Shadbolt, *POAF: Portable Ontology Aligned Fragments*. In International Workshop on Ontologies: Reasoning and Modularity (WORM'08) hosted by the 5th European Semantic Web Conference (ESWC 2008). Tenerife, Spain (2008).

[2] D. Braines, Y. Kalfoglou, P. Smart, N. Shadbolt, J. Bao, *A data-intensive lightweight semantic wrapper approach to aid information integration*. In 4th International Workshop on Contexts and Ontologies (C&O-08) hosted by the 18th European Conference on Artificial Intelligence (ECAI 2008). Patras, Greece (2008).

[3] A. Russell, P. R. Smart, D. Braines, N. Shadbolt, *NITELIGHT: A Graphical Tool for Semantic Query Construction*. In Semantic Web User Interaction Workshop (SWUI-08) hosted by the Computer-Human Interaction Conference (CHI 2008). Florence, Italy (2008).

[4] D. Braines, Y. Kalfoglou, P. Smart, J. Bao, N. Shadbolt, J. Hendler, *Semantic Web techniques to support interoperability is distributed network environments*. In 2nd Annual Conference of International Technology Alliance (ACITA-2008). London, UK (2008).

[5] P. R. Smart, A. Russell, D. Braines, Y. Kalfoglou, J. Bao, and N. R. Shadbolt. *A Visual Approach to Semantic Query Design Using a Web-Based Graphical Query Designer*. In 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW'08). Acitrezza, Catania, Italy, October 2008.

[6] Y. Kalfoglou, M. Schorlemmer, M. Uschold, A. Sheth, and S. Staab. *Semantic interoperability and integration*. Seminar 04391 - executive summary. Schloss Dagstuhl - International Conference and Research Centre, Sept. 2004.

[7] Y. Kalfoglou and M. Schorlemmer. *Ontology mapping: the state of the art*. In the Knowledge Engineering Review, 18(1):1-31, 2003.

[8] J. Euzenat. *An API for ontology alignment*. In Proceedings of In the 3rd International Semantic Web Confernece (ISWC'04), LNCS 3298. Hiroshima, Japan, pages 698-712, Nov. 2004.

[9] R. Palma, P. Haase, and A. Gomez-Perez. *OYSTER: sharing and re-using ontologies in a peer-to-peer community*. In Proceedings of the 15th International World Wide Web Conference (WWW'06). Edinburgh, UK, May 2006.

[10] Y. Kalfoglou and B. Hu. *CMS: CROSI Mapping System - results of the 2005 ontology alignment contest*. In Proceedings of the K-Cap'05 Integrating Ontologies workshop. Banff, Canada, pages 77-84, Oct. 2005.

[11] Y. Kalfoglou and M. Schorlemmer. *IF-Map: an ontology mapping method based on Information Flow theory*. Journal on Data Semantics, 1:98-127, Oct. 2003. LNCS2800, Springer, ISBN: 3-540-20407-5.

[12] J. McCarthy, *Notes on formalizing contexts*. In Proceedings of the 5th National Conference on Artificial Intelligence. Los Altos, CA, USA, pp. 555560, (1986).

[13] J. Euzenat, A. Zimmermann, and F. Freitas. *Alignment-based modules for encapsulating ontologies*. In Proceedings of the K-Cap'07 International Workshop on Modular Ontologies (WoMo'07). Whistler, BC, Canada, Oct 2007.

[14] M. d'Aquin, M. Sabou, and E. Motta. *Modularization: a key for the dynamic selection of relevant knowledge components*. In Proceedings of the ISWC'06 International Workshop on Modular Ontologies (WoMo'06). Atlanta, GA, USA, Nov 2006.

[15] J. Bao and V. Honavar. *Divide and conquer semantic web with modular ontologies - a brief review of modular ontology language formalisms*. In Proceedings of the ISWC'06 International Workshop on Modular Ontologies (WoMo'06). Atlanta, GA, USA, Nov 2006.

[16] R. Porzel, H-P. Zorn, B. Loos, and R. Malaka. *Towards a separation of pragmatic knowledge and contextual information*. In Proceedings of the 2nd International Workshop on Contexts and Ontologies (C&O 2006). Riva del Garda, Italy, (August 2006).

[17] P. DeLeenheer, and A. deMoor. *Context-driven disambiguation in ontology elicitation*. In Proceedings of the AAAI05 Workshop on Contexts and Ontologies (AAAI05/W1). USA, (July 2005).

[18] H. Zeng and R. Fikes. *Extracting assumptions from incomplete data*. In Proceedings of the CONTEXT05 Context Representation and Reasoning (CRR05). Paris, France, (July 2005).

[19] N. Cohen, P. Castro, and A. Misra. *Descriptive naming of context data providers*. In Proceedings of the CONTEXT05 Context Representation and Reasoning (CRR05). Paris, France, (July 2005).

[20] http://demo.openlinksw.com/isparql/ *OpenLink iSPARQL*.

[21] A. Bernstein, E. Kaufmann, A. Gohring, and C. Kiefer. *Querying ontologies: A controlled english interface for end-users*. In Proceedings of the 4th International Semantic Web Conference (ISWC05). (2005), 112126.

[22] J. Borsje, and H. Embregts. *Graphical Query Composition and Natural Language Processing in an RDF Visualization Interface*. In Erasmus School of Economics and Business Economics, vol. Bachelor. Rotterdam: Erasmus University, 2006.

[23] T. Catarci, P. Dongilli, T. D. Mascio, E. Franconi, G. Santucci, and S. Tessaris. *An ontology based visual tool for query formulation support*. In 16th European Conference on Artificial Intelligence (2004).

[24] A. Fadhil, and V. Haarslev. *OntoVQL: A Graphical Query Language for OWL Ontologies*. In International Workshop on Description Logics (DL-2007). Brixen-Bressanone, Italy, 2007.

[25] H. H. Hoang, and A. M. Tjoa. *The virtual query language for information retrieval in the semanticLIFE framework*. In International Workshop on Web Information Systems Modeling. Trondheim, Norway 2006.

[26] B. McBride. *Jena: A Semantic Web toolkit*. Internet Computing, IEEE 6, 6 (2002), 55-59.

[27] http://www.w3.org/2001/sw/DataAccess/rq23/ *SPARQL Query Language for RDF*.