Opening Up Magpie via Semantic Services

Martin Dzbor, Enrico Motta, and John Domingue

Knowledge Media Institute, The Open University, Milton Keynes, UK
{M.Dzbor, E.Motta, J.B.Domingue } @open.ac.uk

Abstract. Magpie is a suite of tools supporting a 'zero-cost' approach to semantic web browsing: it avoids the need for manual annotation by automatically associating an ontology-based semantic layer to web resources. An important aspect of Magpie, which differentiates it from superficially similar hypermedia systems, is that the association between items on a web page and semantic concepts is not merely a mechanism for dynamic linking, but it is the enabling condition for locating services and making them available to a user. These services can be manually activated by a user (pull services), or opportunistically triggered when the appropriate web entities are encountered during a browsing session (push services). In this paper we analyze Magpie from the perspective of building semantic web applications and we note that earlier implementations did not fulfill the criterion of "open as to services", which is a key aspect of the emerging semantic web. For this reason, in the past twelve months we have carried out a radical redesign of Magpie, resulting in a novel architecture, which is open both with respect to ontologies and semantic web services. This new architecture goes beyond the idea of merely providing support for semantic web browsing and can be seen as a software framework for designing and implementing semantic web applications.

1 INTRODUCTION

Magpie [5-7] is a suite of tools supporting a 'zero-cost' approach to semantic web browsing. It avoids the need for manual annotation by automatically associating an ontology-based semantic layer to web resources. There are many ways to characterize Magpie. One view, emphasized in earlier papers, is to consider Magpie as a tool supporting the *interpretation of web pages*. Specifically, one can see the automatic recognition of entities in web pages and the linking of these entities to semantic concepts as a way to bring an interpretative context to bear, which can help users in making sense of the information presented in a web page. For instance, we are using Magpie in a learning context to help students of a course in climate science in understanding the vast mass of information about climate change that can be found on the web. In such a context, using Magpie can be seen as adopting the viewpoint of an expert in the field, and use this as an aid for navigating the web.

Another way to look at Magpie is as a *semantic web browser*. If we take this view, then Magpie can be seen as providing an efficient way to integrate semantic and 'standard' (i.e., non-semantic) web browsing, through the automatic association of semantics to web pages and the provision of user-interface support. This allows the

user to navigate the web using both semantic and hypertext links, and helps him/her to invoke the services appropriate for a given class of ontological entities.

A third viewpoint we can use to characterize Magpie is as a *framework for developing semantic web applications*. According to this view, the Magpie suite of tools can be seen as a 'shell' for building semantic web applications, which provides generic mechanisms for bringing together ontologies, web resources and (semantic) web services. For instance, the climate science example mentioned above can be viewed as a semantic web application, characterized by a number of ontology-based services, which are made available to students opportunistically, when the 'right web page' is encountered. The key feature of Magpie here is that it allows developers to focus on the semantic functionalities, i.e., specifying and populating the ontology and defining the services, with no need to identify, let alone annotate web resources.

Although the idea of Magpie as a framework for building semantic web applications has informed our research since the very beginning, the implementations described in earlier papers fall somewhat short of realizing this vision. In particular, the original Magpie architecture was open with regard to ontologies, but not with respect to services, which had to be statically coupled with the ontology. In other words, they had to be designed by a Magpie developer, much like in a 'closed system' scenario. This approach goes against the vision of the web as an open architecture and more importantly goes against the vision of the semantic web as an open web of interoperable applications [1], which can be opportunistically located and composed, either manually (web services) or automatically (semantic web services).

For these reasons, in the past twelve months we have carried out a radical redesign of Magpie, resulting in a novel architecture, which is open both with respect to ontologies and with respect to functionalities, the latter delivered through semantic web services. This new architecture goes beyond the idea of providing support for semantic web browsing and can be seen as software framework for designing and implementing semantic web applications. Among other things, the new Magpie opens up new communication modalities allowing bi-directional exchange of information among services and users. This is crucial for going beyond the traditional 'click&go' modality of existing hypermedia systems, such as COHSE, and realizing the 'semantic web of applications' vision described above.

In this paper we describe this new architecture for Magpie and we illustrate its functionalities using the example of the Open University's course on climate science, which we mentioned earlier. The paper is structured as follows. In section 2 we illustrate the Magpie functionalities from an end user's perspective. Section 3 describes the new architecture in detail. In section 4 we elaborate the concept of "open publishing", further stressing its importance in the context of the semantic web. Finally, we conclude the paper by reviewing related work in section 5, and by reiterating its key contributions in section 6.

2 MAGPIE AS A RESOURCE AGGREGATOR IN EDUCATION

At The Open University, students enrolling in a level-one climatology course receive printed and multimedia educational material. In addition, they are expected to use

web resources that are often complex scientific analyses and technical reports of climate scientists, as well as technical news stories related to the subject. Magpie facilitates a course-specific perspective on such texts. It enables students to relate the content of third-party documents to the relevant course concepts, materials, activities, and knowledge they are expected to gain from studying the course.

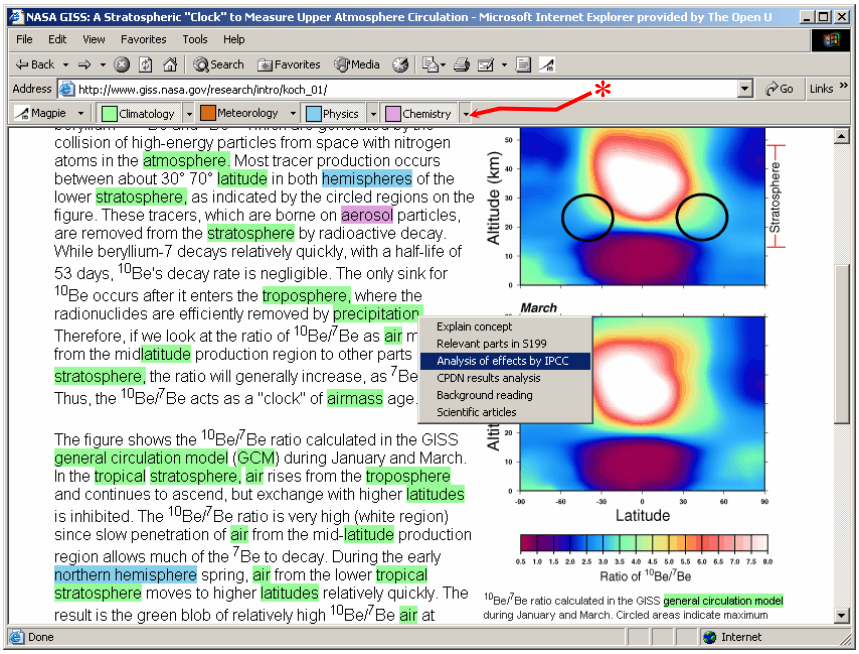

Fig. 1. A climate science related web page with Magpie plug-in highlighting concepts relevant from the perspective of climatology course for a particular student. Menu shown in the center is associated with the concept of '*precipitation*'.

Fig. 1 shows a student's web browser with a web page describing stratospheric circulation. This is a relevant but complex text, so the student interacts with it using the Magpie plug-in. The web page[1] is first annotated with several course-specific ontological concepts by selecting some of the ontology-specific toolbar buttons. In this course the student can annotate concepts in four scientific areas: *Climatology*, *Meteorology*, *Physics*, and *Chemistry*. Annotated and highlighted concepts become 'hotspots' to allow the user to request a menu with relevant functionalities for a relevant item. In Fig. 1, the contextual right-click on the '*precipitation*' reveals a menu of semantic services. The choices depend on the ontological classification of a particular concept in the selected ontology.

Our new, services-oriented framework supports composition of such semantic menus from the services available for a particular ontology. These services can be in principle implemented by different knowledge providers. For instance, service '*Relevant parts in S199*' is an internal index to the course material. On the contrary, the

[1] The original text is a property of NASA Goddard Institute for Space Studies, and the page can be accessed at http://www.giss.nasa.gov/research/intro/koch_01/.

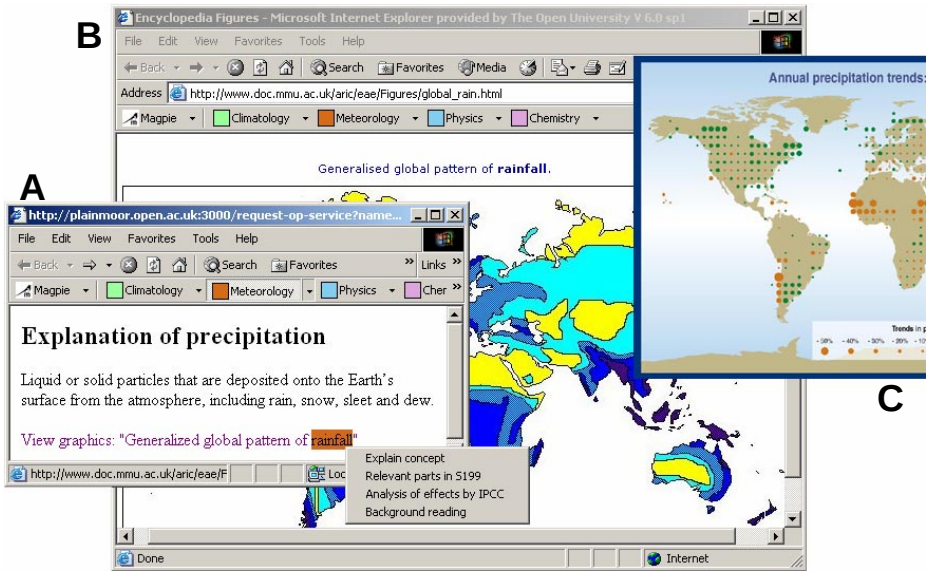

Fig. 2. Results of the '*Explain concept*' semantic query invoked for the '*precipitation*' concept by the semantic menu action depicted in Fig. 1. Window A shows a brief explanation drawing on course glossary and a link to the associated image originating at a third-party site. The actual image related to the concept based on its semantic proximity is in window B. Window C shows a sample analysis relevant to the same concept by Intergovernmental Panel on Climate Change.

'*Background reading*' service is provided by a different university that uses its own, proprietary encyclopedia to provide contextually related reading on a particular concept/topic. Yet another type of service is '*Explain concept*'. This is an aggregating service using sophisticated ontology-based reasoning to combine chunks of textual and visual knowledge explaining a particular concept. The combination is based on having access to simpler services retrieving semantically annotated knowledge chunks from several sources and appreciating their semantic 'closeness'. Obviously, the degree of sophistication of the services is independent of the Magpie architecture, which considers all services as black boxes.

Thus, the '*Explain concept*' service in Fig. 1 generates a textual explanation from the course glossary, and attempts to attach a related image or scheme if this exists in its repository of annotated materials (e.g. Fig. 2B). The answer as shown in Fig. 2A does not explicitly exist in the course books, and indeed it is an interpretative viewpoint of the selected ontology. It facilitates an expert's view – as if a tutor was associating different materials together. Because the answer to a semantic query may be a web resource in its own right, it can be further browsed or annotated semantically. Here Magpie merges the independent mechanisms for recognizing semantic relevance and browsing the resulting web resources.

Entirely different strategy is employed by the '*Analysis of effects by IPCC*' service. Unlike the services mentioned earlier, this focuses more on computing the answer rather than linking to any relevant document. A sample response to invoking this semantic service for concept '*precipitation*' may look as shown in Fig. 2C.

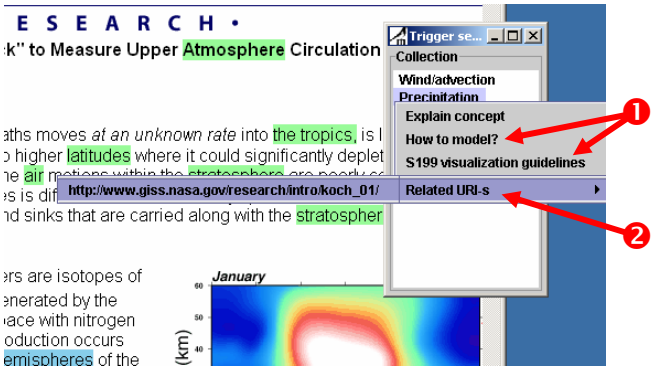

Fig. 3. A simple trigger service aggregating all those concepts from a page, which could be further investigated by the users. This 'further investigation' usually comprises a hands-on modeling exercise with their customized climate model (courtesy of the climate*prediction*.net project) – see pointer ❶. In addition to providing dedicated guidance, the trigger service also serves as a semantic bookmarking tool remembering where particular concepts appeared (see pointer ❷).

The support Magpie provides for trigger services, is based on the 'subscribe&acquire' rather than 'click&go' user-system interaction modality. Our dedicated interfaces, called *collectors*, visualize the results of such services. For example, the collector in Fig. 3 aggregates those concepts appearing in the web page that can be modeled or visualized using the state-of-the-art climate model each student runs on his/her computer. As shown in the figure, the list of collected items differs from the highlighted concepts because the model is constrained to run a sub-set of possible analyses. This alternative way of delivering information to the students might implement a tutor's pedagogical goal of providing additional information to those students who are interested in a deeper understanding of the topic. Magpie collectors also offer a range of other functionalities, such as semantic bookmarking or browsing history management. These are discussed in detail e.g. in [5].

This example presents Magpie as an application development framework for building semantic web applications. In this case, a course-specific ontology supports students in making sense of information about climate science, independently of where this information resides on the web. The application is built by selecting or constructing the appropriate ontology and by defining the appropriate services. This example highlights the desirability of an architecture open with respect to services, so that more functionalities can be made available to students, using standard mechanisms for interoperability on the web. This openness facilitates more focused approach to using scarce resources, and enables better personalization. In the next section we look at the backend infrastructure required to support openness with respect to services.

3 AN OPEN SEMANTIC SERVICES ARCHITECTURE

We now describe the architecture, which allows Magpie users to define, publish and use their own semantic services (shown in Fig. 4). It uses an infrastructure we have developed – IRS-II [13], which supports the publishing and invocation of semantic

services. IRS-II is based on the UPML framework [8], and therefore differentiates between *tasks*, *problem solving methods* (generic reasoners) and *domain models*. By distinguishing between tasks and problem solving methods we separate the activity of specifying and implementing semantic services (problem solving methods) from making them available in a form that can be invoked in terms familiar to a user (tasks).

The Magpie architecture comprises a *Service Provider* and a *Service Recipient*. These are briefly described in the next two sections.

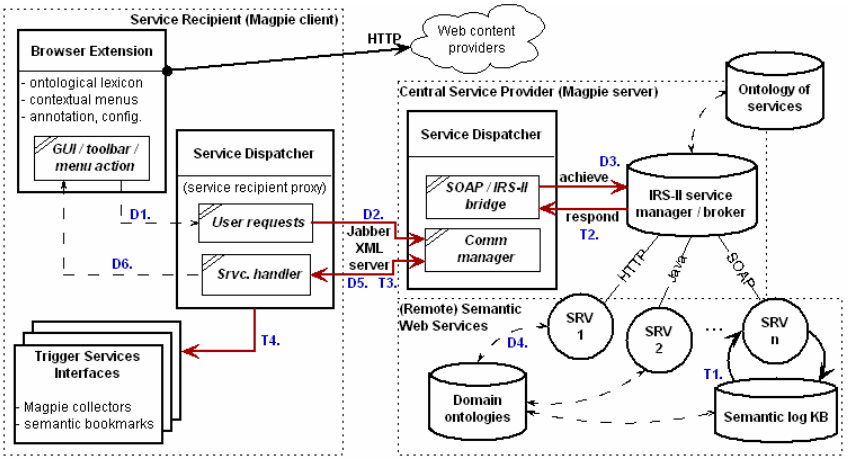

Fig. 4. Schematic architecture of "open services" Magpie framework

3.1 Service Recipient Components

On the service recipient side the framework features: the *Magpie Browser Extension* (of Internet Explorer), the *Magpie Client-Side Service Dispatcher*, and *Trigger Service Interfaces*. The Magpie Browser Extension has already been described in detail in earlier papers [5, 7], and therefore does not need to be discussed again. In a nutshell it provides the basic Magpie functionality, by automatically matching items in a web page to items in the selected ontology and by allowing users to bring up a menu of services contextualized for ontological concepts.

The Magpie Client-Side Service Dispatcher acts as a dedicated service proxy for the user. It manages the communication between the Browser Extension and the service dispatcher embedded in the Magpie server. The Dispatcher delivers both user requests and the responses from providers, using customized XML messages, e.g. to be used by collectors. These are a form of Magpie Trigger Service Interfaces, which are able to visualize the data pushed by the specific trigger services a user subscribes to.

Trigger services are an important innovation in the Magpie infrastructure. Unlike contextual, menu-based services, trigger services are activated based on patterns and relations among concepts recognized in the page and automatically asserted in a semantic log. The subscription system allows the user to filter only 'useful' items to be collected. Since a lot of spam is due to 'pushing' unsolicited content to the users, the principle of trigger services is different. They are not designed for 'blanket coverage'

of all users browsing a particular page. They are selected and activated by the user and only push information to him/her when a specific pattern emerges on the page.

The Client-Side Service Dispatcher handles the interactions between the user, the Magpie-enabled browser and the Magpie service providers. In principle, it is an alternative to the GET/POST requests available for standard hypermedia. Although Magpie supports such requests, a growing number of services are available in formats not suitable for integration with a standard web browser, and for this reason the Magpie architecture supports a more generic approach to service mediation.

In particular, the Magpie dispatcher acts on behalf of the user and can be identified as such. Hence, it is possible to communicate service requests/responses *asynchronously*. This is an important extension of standard hypermedia protocol, which assumes synchronous interactions. This capability is also critical for supporting trigger services or generally, semantically-filtered 'pushed content'. Such a *two-way communication* is not possible in standard HTTP-based hypermedia systems.

The support for asynchronous interaction between the client and the server makes the Magpie architecture extremely flexible. For instance, trigger messages may be redirected to a more appropriate user interface than a standard web browser (e.g. a graphic visualization widget). The possibility of bi-directional information exchange is also useful to support negotiation. For instance, different degrees of response granularity may be available, or ontologies may be stored in different formats. Our dispatchers may make it possible for the user to customize the ontology used for interpreting the web pages; e.g. by selecting "a relevant sub-set" of an extensive domain model.

3.2 Service Provider Components

A number of components on the service provider side manage value-added functionalities such as semantic logging, the association of semantic services with ontological classes, and reasoning for trigger services. Two criteria important for designing this "back-office support" for semantic enrichment of web browsing in Magpie are:

- Open as to the definition of new semantic services, which may use an existing ontology and a require access to the semantic log, and
- Allowing users to customize how the output of a service is rendered.

The rationale for these criteria is similar to that of the envisaged Semantic Web [1]. Rather than authors hard-wiring the relationships into the content of web resources, the users are allowed to (re-)use data and services, and adapt them to different contexts. Our Magpie service manager caters for authors publishing new services, and for users selecting or subscribing to a particular set of services. Since Magpie relies on ontologies for associating services with web content, the authors have to publish semantic descriptions of their services. In other words, given the ontology-centric nature of Magpie, a key requirement here is to integrate Magpie with an architecture for *semantic* web services, rather with standard (i.e., non-semantic) web services.

As already mentioned, we have fulfilled this requirement by integrating Magpie with the IRS-II architecture [13], and the Magpie service manager uses the IRS-II framework to handle the subscriptions of individual users to individual services. The

service manager also communicates with the IRS broker, whose job is to locate the appropriate web service when a request is made to achieve a task.

Conceptually, the integration between Magpie and the IRS is achieved by defining a top-level *Magpie task*, which takes as input an ontological identifier of a concept, various related arguments and a choice of a visual renderer specifying the desirable output format (typically HTML). Specific semantic services inherit from this generic task, and extend it with specific input or output roles (e.g., the semantic service 'Explain concept' described in section 2 takes a concept from a specific category as input and a textual definition or a pair of textual/graphical definitions as its output).

In IRS-II a task can be handled by one or more *problem-solving methods* (PSMs). PSMs are the knowledge-level descriptions of code for reasoning with particular input roles as specified by the task definition. PSMs tackle a particular task, and introduce a system of *pre-conditions* (e.g. an argument supplied mustn't be a 'Physics' or 'Chemistry' concept), and *post-conditions* (e.g. show graphics if available). While PSMs are crucial for reasoning, the end-user only interacts at the task level – thus specifying what needs to be achieved rather than how to achieve it. The IRS-II framework supports different modes of service publishing and invocation, as well as other emerging standards such as WSDL [13]. Regardless of the publication, IRS-II generates a unique "access URI" where the web service can be invoked. These URIs are then used in Magpie to achieve a particular task when a user right-clicks a particular 'hotspot' item in a web page.

4 . DEFINING MAGPIE SEMANTIC SERVICES

The process of manual definition of semantic services for different ontologies by the knowledge engineer is not feasible on a larger scale. We argued in section 3.2 that the publication procedure has to be open. Before describing the technical details of defining semantic services using the web services framework, we re-consider the educational example from section 2 to highlight key benefits for the end-user.

4.1 Benefits of Open Services Architecture

Magpie shows the *on-demand services menu* whenever the mouse hovers over a recognized entity. This menu is context-dependent, but so-far, we have used a one-size-fits-all semantic context defined by the membership of a particular entity to a particular ontological class. The class membership is defined in the ontology, and is essentially the same for all users subscribing to a particular ontology. This acknowledges a certain degree of commonality of purpose among the subscribers, but it is a rather superficial commonality. For example, if a tutor decides to divide the students into 'standard' and 'advanced' learners, Magpie should reflect this pedagogic strategy by offering suitable services to the sub-groups of students based on the tutor's choice.

Another issue concerns the development of such educational resources. Institutions accelerate their course production and update procedures, but with printed materials this may still take some time. In Magpie framework, new semantic services for stu-

dents could be developed and updated continuously. Because of the nature of the framework, new services are accessible to the end users (students) with 'near-zero' delay since their development.

4.2 Defining Magpie Tasks

The generic Magpie framework comes with core ontologies that define a) a basic domain ontology of items such as 'Thing' and 'HTML', b) a task ontology defining the top-level Magpie task, and c) a basic PSM ontology of default Magpie services (e.g. rendering HTML and XML-based output). The magpie-generic-request-task is a top-level, generic description taking three inputs: user's identity (UID), ontological concept identifier (OID), and rendering. Its output has to be a result that complies with the rendering provided as an input argument, and it can be in form of RDF, HTML or a generic list. Specific Magpie tasks inherit from this generic request task.

```
magpie-explain thing-task  (magpie-generic-request-task)
  input roles:  has_oid          // task is invoked for a given item
                has_user         // task is invoked by a given user
                has_rendering    // result rendered as …

  output role:    has_result     // the result of the service

  constraints:  type-of (has_oid) = (or Climatology Meteorology Physics Chemistry)   ❶
                cardinality (has_oid) = 1
                value (has-pretty-name) = "Explain concept"                         ❷

  has-goal-expression:   value = (kappa (?task ?sol)                                ❷
                                   (= ?sol  (magpie-render-function  (role-value ?task has-rendering)
                                              (request-explain-thing-function  (role-value ?task has-oid)))))
```

Fig. 5. Schematic task description for Magpie service 'Explain concept' in IRS-II

A task description for the service '*Explain concept*' containing optional graphics (from section 2) is shown in Fig. 5. Marker '❶' highlights parts that re-define the generic task; e.g. restricting the concept OID to be an instance of Climatology or Meteorology classes. These classes come from a specific, climate course reference ontology. Hence, the '*Explain concept*' task is defined in an ontology that inherits from both a generic ontology of Magpie tasks and from the course ontology.

The definition of the magpie-explain-thing-task contains two extra slots (see markers '❷'): has-pretty-name labels the task in the displayed semantic menu, and has-goal-expression plays two roles. Firstly, it specifies what the task does for a human reader. Secondly, the expression enables knowledge-level reasoning about the task for the purposes of automatic service location or composition.

4.3 Defining Magpie PSMs

Having described what a particular task can do for a particular ontology, the semantic web application developer describes the actual methods tackling it. Fig. 6 shows a PSM description, which looks similar to that of a task. The main difference is that the

developer defines different PSMs to implement a particular task for different types of input. For instance, let's assume that we have already defined a generic (empty) PSM for the '*Explain concept*' service, and now want to narrow it for the class '*Climatology*'. Constraining a PSM for a particular class means introducing an additional applicability condition (the input has to be a Climatology concept) that must be satisfied before invoking this method. This condition semantically annotates the code that implements it and makes it applicable only to this class. In practice this may mean that different providers can handle different categories of data using mutually incompatible approaches or techniques. Our PSMs take this fully into account.

```
magpie-explain-climatology-provider     (magpie-explain-thing-provider)
  tackles-task:                  magpie-explain-thing-task
  has-applicability-condition:   (kappa (?p)
                                   (climatology-category  (role-value  ?p has-oid)))
```

Fig. 6. PSM description for Magpie service tackling the task '*Explain concept*' (see Fig. 5) for the '*Climatology*' class (category)

Applicability conditions enable the broker to select from several PSMs tackling the same task. There may be several distinct descriptions (e.g. with or without graphical explanation), but the user needs to refer to a single task ('Explain concept'). The service manager can invoke a specific PSM and its implementation depending on the type of the submitted argument or user preference. This also explains why semantic menus for different categories may use the same label to identify a service (task) that delivers different functionality in a different context (through different PSMs).

4.4 Publishing and Invoking Magpie Services

The final step is to write a snippet of code in Java (for example), and to make it available by publishing it via IRS-II. Publishing is explained in [13]; here it suffices to say that it essentially means creating wrappers turning a piece of proprietary code into a web service and associating that service with an appropriate PSM. The associations are stored in a registry, which is referred to whenever a user makes a request to achieve a particular task with a particular set of arguments. The IRS-II takes care of invoking the appropriate service provider code, passing the necessary arguments, and processing the results.

All the activities described so-far were done by service or semantic web application developers – outside of the Magpie end user scope of attention. The implications of such a standard means of publishing services for the end user are in the simplification of the entire interaction. Once the user decides which classes in the selected ontology he/she is interested in, the Magpie plug-in uses the Dispatcher to request the semantic services applicable to each chosen class. The generator of semantic menus is itself a service that takes a (top-level) class name as an input, and sends back a list of URIs to invoke applicable tasks. The generator takes all tasks published for a particular ontology, where at least one argument matches a given class; i.e. the input type is either the given class or is a subclass of the given class. Each service is rendered as a pair ⟨pretty-name, URI⟩; for example (the 'XXXN' replaces the actual values):

Explain concept
http://irsserver:3000/achieve-task?ontology=climateprediction-kb& task=request-explain-thing-task&has_oid=XXX1& has_user=XXX2 & has_rendering=XXX3

The actual execution of a particular service from a semantic services menu in a Magpie-enabled web browser invokes the task behind the 'pretty label' (through URI). The benefit to the user is the task-based reference to semantic services. Instead of knowing (and labeling) the individual methods, the user refers to a particular functionality using a single label, regardless of the context. This reduces the maintenance overhead. New variations of a service for different contexts can be added without changing the structure of the displayed menus; the implementation or location of an individual service can be changed transparently.

4.5 Specifics of Trigger Services

Magpie trigger services can be defined following the same procedure as for the on-demand services (described in previous sections). However, there is one major difference; a generic 'trigger task' has no particular concept (OID) as its input. It does not make sense for a trigger service to invoke it for a single specific item (e.g., 'airmass'). An access to regularly updated semantic log, which resides on one of the central Magpie servers, is required instead. Since services (including trigger ones) can be defined by anyone, and may be distributed, there is a problem in accessing the central semantic log. This cannot be replicated because of security issues, and for this reason, each trigger service needs to monitor it for a particular (approved) pattern.

As a result, both the task and PSM definitions of trigger services use a simple extension to the standard service specification mechanism. First, the author declares that a particular trigger name is associated with a particular pattern. The pattern is then typically defined in terms of applicable antecedents for an 'IF…THEN…' rule. For example, a trigger may generate experiment guidelines to investigate concepts found in a web page (for which some guideline exists). The pattern would look like this:

(and (climatology-category ?X) (has-link-to ?X ?Y) (experiment-guide ?Y))

The author then associates a task invocation with the pattern. The actual implementation of the trigger task calls a central, shared service monitoring the semantic log with this specific pattern as one of the arguments. In other words, once a user registers with a particular trigger task, the respective pattern is forwarded to the semantic log monitor, where it is applied whenever the log is updated. Otherwise, the trigger task behaves as a standard service; it takes the asserted data as an input and distributes an alert to all those users who subscribed to the trigger. An example of how this trigger functionality has been used to facilitate team collaboration is discussed in [6].

5 RELATED WORK

A tool that functionally resembles Magpie is the KIM plug-in for Internet Explorer [14]. Knowledge and Information Management (KIM) is a platform for automatic

semantic annotation, web page indexing and retrieval. As Magpie, it uses named entities as a foundation for document semantics, and assigns ontological definitions to the entities in the text. The platform uses a massive populated ontology of common 'upper-level' concepts (e.g. locations, dates, organizations or money) and their instances.

Unlike Magpie, KIM is based on the GATE platform [4] but it extends its flat NER rules with ontological hierarchies. The entities are recognized by the KIM proxy, and in parallel they are associated with respective instances in the ontology. GATE supports the recognition of acronyms, incomplete names and co-references thus enabling KIM to work with both already-known and new named entities.

Magpie differs from KIM in a number of respects. While KIM is coupled with a specific, large knowledge base, Magpie is open with respect to ontologies, allowing users to select a particular semantic viewpoint and use this to enrich the browsing experience. Another important difference is that while KIM is very much steeped in the classic 'click&go' hypermedia paradigm, Magpie is open with respect to services, as discussed in this paper. Hence, as already pointed out Magpie goes beyond KIM in the direction of providing a framework for building semantic web applications, rather than simply supporting semantic annotation and semantic web browsing.

Magpie also differs from 'free-text' document annotation tools [9, 10] by intertwining entity recognition, annotation and ontological reasoning. Annotation using ontological lexicons outperforms 'free-text' annotations in terms of >90-95% recall rate and similar precision for in-domain resources. Yet, free-hand annotations are useful for ad-hoc, personal, customized interpretation of the web resources. Magpie does not currently support manual semantic annotation, which is a limitation. To address this issue we will shortly begin work on integrating Magpie with MnM, a semantic annotation framework developed at the Knowledge Media Institute [16].

From user interface adaptability perspective Magpie is relevant to projects such as Letizia [11] with its reconnaissance agents. This type of agent "looks ahead of the user on the links on the current web page". Such pre-filtering may use semantic knowledge to improve the relevance and usefulness of browsing. Magpie implements functionality similar to that of Letizia ("logged entities reconnaissance") through semantic logging and trigger services, and thus provides a more general and flexible framework for implementing push services, than the one provided by Letizia.

Another system superficially similar to Magpie is COHSE, which implements an open hypermedia approach [2]. The similarity between the two systems is due to the fact that (at a basic level of analysis) both work with web resources and use similar user interaction paradigms ('click&go'). However, beyond the superficial similarity there are very major differences between these two systems. The main goal of COHSE is to provide dynamic linking between documents – i.e., the basic unit of information for COHSE is a document. Dynamic linking is achieved by using the ontology as a mediator between terms appearing in two different documents. COHSE uses a hypertext paradigm to cross-link documents through static and dynamic anchors. In contrast with COHSE, Magpie is not about linking documents.

As emphasized in the introduction, there are three ways we can look at Magpie: as a way to support semantic web browsing, as a tool to support interpretation of web resources through 'ontological lenses' and as a framework for building semantic web applications. In particular, if we take the latter perspective, Magpie goes beyond the notion of hypermedia systems, by providing a platform for integrating semantic ser-

vices into the browsing experience, both in pull and push modalities. In a nutshell, Magpie uses a different paradigm. It views web as a knowledge-based servicing of various user needs. Using Magpie's "web as computation" paradigm, we not only provide information about one concept, but can easily offer knowledge dependent on *N-ary relationships* among concepts. This is impossible in any hypermedia system – one can't use one click to follow multiple anchors simultaneously, and reach a single target document or a piece of knowledge.

Moreover, Magpie supports the publishing of new services without altering the servers or the plug-in. Service publishing leaves the users in control by allowing them to subscribe to selected services. It also makes the development of a semantically rich application more modular; thus cheaper and easier for domain experts rather than knowledge engineers. This is more powerful than the mechanisms used by open hypermedia systems, which are largely based on the "editorial choice of links". Magpie explores the actual knowledge space as contrasted with navigating through hypertext as one of its explicit manifestations. Mere link following (in open or closed hypermedia) is not sufficient to facilitate document interpretation. We complement the familiar 'click&go' model by two new models: (i) 'publish&subscribe' (for services) and (ii) 'subscribe&acquire' (for data and knowledge).

To conclude we want to note a growing recognition by the research community of the need to make the semantic web accessible to "ordinary" users. Two approaches follow similar, lightweight and near-zero overhead principles as Magpie; albeit for different purposes. The authors of Haystack [15] and Mangrove [12] argue that the major issue with Semantic Web is the gap between the power of authoring languages such as RDF(S) or OWL and sheer simplicity of HTML. In response to this concern, Magpie separates the presentation of semantic knowledge, service authoring, and publishing from the underlying knowledge-level reasoning mechanisms.

6 CONCLUDING REMARKS

In this paper we described the Magpie framework focusing on how semantic services can be deployed and used in an open fashion. Magpie is an *open architecture* in respect to the technological infrastructure; it is not bound to a single 'services standard'. The separation of the user interface from a 'thin communication client', the Client-Side Service Dispatcher, offers several benefits, and enables us to implement dynamically defined/updated on-demand and trigger services. By combining semantic web, browsing technologies and semantic web services we created an open framework that maximizes flexibility. A Magpie user is free to select both the overall viewpoint, captured within an ontology, and the associated services. Semantic web application developers are free to customize and extend the available services including a number of core Magpie services such as the NER and lexicon generation.

As we showed in our example, Magpie enables lay members of the public to explore rich scientific resources (such as climatology and climate prediction, for example). Thus, the semantic browsing capabilities of Magpie may serve as an *enabling technology* for the increased public understanding of science. In the past papers, we presented Magpie as a tool for browsing the Semantic Web. However, as Tim Bern-

ers-Lee argues: "Semantic Web is about integration of resources rather than browsing."[2]. Leaving aside the philosophical issue of what constitutes "*the semantic web browser*", the extended Magpie framework can be seen as a display *integrating* knowledge resources *distributed throughout* the Semantic Web. Web services offer small, easier to maintain modules of a larger semantic web application, could be authored independently, and stored as a distributed system.

Whether the Semantic Web is about browsing, computation or integration, the main contribution of our research is in allowing users to browse the standard Web whilst utilizing the concepts and relationships captured within a selected ontology. The semantic services (whether on-demand or triggered) that are offered through the Magpie framework enhance web interoperability and user interaction with knowledge. Magpie plug-in acts more as an end-user interface for accessing and interacting with the distributed semantic web services rather than a mere "web browser".

For the future, a number of issues remain open. As mentioned earlier, we want to integrate Magpie with a framework for semantic annotation [16], to allow seamless integration of browsing, service invocation and annotation. This would enable the users to extend and/or customize the existing ontological commitments. We are also working on more powerful named entity recognition mechanisms, both ontology-based and general-purpose (such as GATE [4]). Finally, we are working on a set of support tools that would enable web developers to publish their applications for the Magpie users quickly and simply. This is critical in order to reduce the development time for any subsequent applications of our framework.

Once this publishing facility is in place, a comprehensive usability study needs to be performed to back our assumptions and design principles. Nonetheless, our initial experiments with tools supporting the application developers seem to support our experience-driven requirement for reducing the complexity of interacting with semantic web services. Magpie is clearly geared towards high recall/precision annotation *within a specific domain*. Early evidence suggests there is a benefit for naïve users and novices in interacting with the domain knowledge in such a constrained way. However, to measure the value-added of Magpie more objective usability study is planned.

7 ACKNOWLEDGMENTS

The Magpie effort is supported by the climate*prediction*.net and the Advanced Knowledge Technologies (AKT) projects. Climate*prediction*.net is sponsored by the UK Natural Environment Research Council and UK Department of Trade e-Science Initiative, and involves Oxford University, CLRC Rutherford Appleton Labs and The Open University. AKT is an Interdisciplinary Research Collaboration (IRC) sponsored by the UK Engineering and Physical Sciences Research Council by grant no. GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and The Open University.

[2] Quote from Tim Berners-Lee's keynote speech delivered at the 2nd International Semantic Web Conference, Sanibel Island, Florida, US, October 2003.

8 REFERENCES

[1] Berners-Lee, T., Hendler, J., and Lassila, O., *The Semantic Web.* Scientific American, 2001. **279**(5): p.34-43.

[2] Carr, L., Bechhofer, S., Goble, C., *et al. Conceptual Linking: Ontology-based Open Hypermedia*. In *Proc. of the 10th Intl. WWW Conf.* 2001. Hong-Kong.

[3] Ciravegna, F., Chapman, S., Dingli, A., *et al. Learning to Harvest Information for the Semantic Web*. In *1st European Semantic Web Symposium*. 2004. Greece.

[4] Cunningham, H., Maynard, D., Bontcheva, K., *et al. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications.* In *Proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL)*. 2002. Pennsylvania, US.

[5] Domingue, J., Dzbor, M., and Motta, E. *Magpie: Supporting Browsing and Navigation on the Semantic Web*. In *Proc. of the Intelligent User Interfaces Conf. (IUI)*. 2004. Portugal.

[6] Domingue, J., Dzbor, M., and Motta, E. *Collaborative Semantic Web Browsing with Magpie*. In *1st European Semantic Web Symposium*. 2004. Greece.

[7] Dzbor, M., Domingue, J., and Motta, E. *Magpie: Towards a Semantic Web Browser*. In *Proc. of the 2nd Intl. Semantic Web Conf.* 2003. Florida, USA.

[8] Fensel, D. and Motta, E., *Structured Development of Problem Solving Methods.* IEEE Transactions on Knowledge and Data Engineering, 2001. **13**(6): p.913-932

[9] Handschuh, S., Staab, S., and Maedche, A. *CREAM - Creating relational metadata with a component-based, ontology driven annotation framework*. In *Intl. Semantic Web Working Symposium (SWWS)*. 2001. California, USA.

[10] Kahan, J., Koivunen, M.-R., Prud'Hommeaux, E., *et al. Annotea: An Open RDF Infrastructure for Shared Web Annotations*. In *Proc. of the 10th Intl. WWW Conf.* 2001. Hong-Kong.

[11] Lieberman, H., Fry, C., and Weitzman, L., *Exploring the web with reconnaissance Agents.* Comm. of the ACM, 2001. **44**(8): p.69-75.

[12] McDowell, L., Etzioni, O., Gribble, S.D., *et al. Mangrove: Enticing Ordinary People onto the Semantic Web via Instant Gratification*. In *Proc. of the 2nd Intl. Semantic Web Conf.* 2003. Florida, USA.

[13] Motta, E., Domingue, J., Cabral, L., *et al. IRS-II: A Framework and Infrastructure for Semantic Web Services*. In *Proc. of the 2nd Intl. Semantic Web Conf.* 2003. Florida, USA.

[14] Popov, B., Kiryakov, A., Kirilov, A., *et al. KIM - Semantic Annotation Platform*. In *Proc. of the 2nd Intl. Semantic Web Conf.* 2003. Florida, USA.

[15] Quan, D., Huynh, D., and Karger, D.R. *Haystack: A Platform for Authoring End User Semantic Web Applications.* In *Proc. of the 2nd Intl. Semantic Web Conf.* 2003. Florida, USA.

[16] Vargas-Vera, M., Motta, E., Domingue, J., *et al. MnM: Ontology Driven Semi-automatic and Automatic Support for Semantic Markup*. In *Proc. of the 13th European Knowledge Acquisition Workshop (EKAW)*. 2002. Spain.