Semantic Layering with Magpie

John Domingue
Knowledge Media Institute
The Open University
Milton Keynes, UK
+44-1908-655014
J.B.Domingue@open.ac.uk

Martin Dzbor
Knowledge Media Institute
The Open University
Milton Keynes, UK
+44-1908-858524
M.Dzbor@open.ac.uk

Enrico Motta
Knowledge Media Institute
The Open University
Milton Keynes, UK
+44-1908-653506
E.Motta@open.ac.uk

ABSTRACT
Browsing the web involves two main tasks: finding the right web page and then *making sense* of its content. A significant amount of research has gone into supporting the task of finding web resources through 'standard' information retrieval mechanisms, or semantics-enhanced search. Much less attention has been paid to the second problem. In this paper we describe Magpie, a tool which supports the interpretation of web pages. Magpie acts as a complementary knowledge source, which a reader can call upon to quickly gain access to any background knowledge relevant to a web resource. Magpie works by automatically associating an ontology based semantic layer to web resources, allowing relevant services to be directly invoked within a standard web browser. The functionality of Magpie is illustrated using examples of how it has been integrated with our lab's web resources.

Categories and Subject Descriptors
H.5.4 [**Hypertext/Hypermedia**]: Semantic Web – *navigation and architecture, ontologies, semantic services, entity annotation.*

General Terms
Algorithms

Keywords
Semantic web, semantic services, ontology, smart browsing.

1. INTRODUCTION
Browsing the web involves two main tasks: finding the right web page and then *making sense* of its content. A significant amount of research has gone into supporting the task of finding web resources, either by means of 'standard' information retrieval mechanisms, or by means of semantics-enhanced search [13, 19]. Much less attention has been paid to the second problem, supporting the *interpretation* of web pages. Annotation technology [17, 23, 27] allows users to associate meta-information with web resources, which can then be used to facilitate their interpretation. While this technology provides a useful way to support group discussion and shared interpretation, it is nevertheless very limited. Annotation is normally carried out manually, which means that the quality of the sensemaking support is dependent on the willingness of stakeholders to provide annotation and their ability to provide valuable information. This is of course even more of a problem, if a formal approach to annotation is assumed, based on semantic web technology [1].

WWW 2003, May 20-24, 2003, Budapest, Hungary.
ACM xxx.

In this paper we describe Magpie, a tool supporting the interpretation of web pages. Magpie acts as a complementary knowledge source, which a reader can call upon to quickly gain access to any background knowledge relevant to a web resource. Magpie follows a different approach from that used by the aforementioned annotation technology: it automatically associates a semantic layer to a web resource, rather than relying on manual annotations. This process relies on the availability of an *ontology* [12], an explicit, declaratively specified representation of a domain of discourse. Ontologies are the cornerstone of the emerging semantic web: they provide conceptual interoperability needed to allow semantic agents to make sense of information on the web and to collaborate with other semantically aware agents. Magpie uses ontologies in a similar way: they make it possible for Magpie to associate meaning with the items of information found on a web page and then, on the basis of the identified meaning, to invoke the relevant services, or offer the user the appropriate functionalities.

The Magpie-mediated association between an ontology and a web resource essentially provides an *interpretative viewpoint* or *context* over the resource in question. Indeed the overwhelming majority of web pages are created within a specific context. For example, the personal home page of a member of the Knowledge Media Institute would have normally been created within the context of that person's affiliation and organizational role. Of course, some readers would be very familiar with such context, while others would not. In the latter scenario the use of Magpie is especially advantageous, given that the context would be made explicit to the reader and context-specific functionalities will be provided. Naturally, because different readers have differing levels of familiarity with the information shown in a web page and with the relevant background domain, they require different level of sensemaking support. A semantic layer in Magpie is consequently designed with a specific type of user in mind.

In a seminal study of how users browse the web, Tauscher and Greenberg [26] found the following statistics on the types of actions users typically carry out:

- 58% of pages visited are revisits,
- 90% of all user actions are related to navigation,
- 30% of navigation actions are through the 'Back' button,
- less than 1% of navigation actions use a history mechanism

A fairly obvious conclusion of these statistics is that web users need support in capturing what they have seen previously. Current history mechanisms, 'Back' button aside, are of little help. Magpie, automatically tracks interesting items found in a browsing session within a *semantic log*. The semantic log allows *trigger services* to be created, which are activated when a specific pattern of items has been found. One type of trigger service offered in

Magpie is a *collector*, which collects items from a browsing session using an ontology-based filter. Some example collectors are shown in the following section.

The rest of this paper is structured as follows. In the next section we give an overview of the functionality of Magpie through a scenario. Sections 3 and 4 describe the Magpie design principles and architecture in detail. We then describe the different types of semantic services available in section 5. In section 6, we briefly mention pre-requisites for a successful deployment of Magpie, especially the population of ontologies. Finally, in sections 7 and 8, we review related research and draw the main conclusions from this research.

2. A MAGPIE USAGE SCENARIO

Imagine a journalist is writing an article on the Knowledge Media Institute (KMi) for a magazine. One of her tasks is to collect information about the most important projects led by senior KMi staff. Using a web browser with a Magpie extension, she starts with a visit to the home page of the lab's director Enrico Motta. After loading the page, she wants to highlight interesting concepts denoting researchers, collaborating organizations, projects and research areas in the page. These concepts draw upon an existing ontology of academic and research organizations that has been populated with instances representing the people and projects and research areas of KMi and the organizations that we collaborate with. Figure 1 shows the journalist's browser with the concepts of interest highlighted using the Magpie toolbar. A key requirement for Magpie design was that any web page viewed through the system should look the same as when viewed through a standard web browser. This constraint reduces the confusion, which can occur when the content and/or appearance of a web page are altered. The Magpie toolbar (see close-up in Figure 2) allows users to toggle background highlighting for the specified types of entities. The 'Services' button in the toolbar activates a context dependent *Semantic Services* menu, which replaces the standard web browser's right-click menu.

On the right-hand side of Figure 1 are three Magpie *collectors*. These are automatically filled by Magpie *trigger services* as the user browses. During a browsing session, all entities found on accessed web pages are asserted into a *semantic log* knowledge base (KB). Collectors are set up to show a particular, semantically filtered view of the semantic log. For instance, the top two collectors in Figure 1 show the people and projects that have been recognized. So far, only one person and five projects have been explicitly mentioned. The bottom collector shows the projects associated with any people recognized during the session. Note that these projects have not been mentioned explicitly on any page. Rather, they originate from the populated domain ontology. As we can see from Figure 1, Enrico Motta is associated with six additional projects not mentioned on his web page.

From the content of the web page our journalist can see that the ScholOnto project might be one of the sought-after projects to be included in her report. Hence she wonders if any related projects could be included in the same section. She right-clicks the 'ScholOnto' term, and the *semantic services menu* shown in Figure 3A appears. The choices in the menu depend on the class of the selected item/entity within the selected ontology. In our case, 'ScholOnto' is classified as a project, so project-related op-

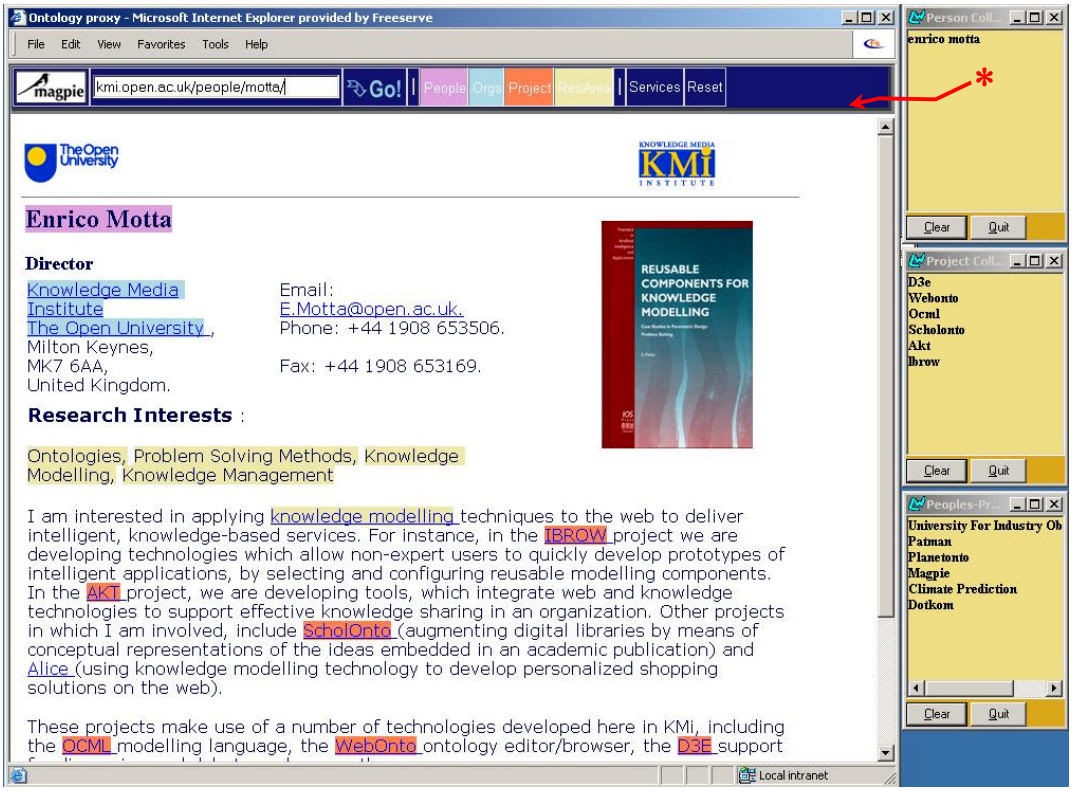

Figure 1. Enrico Motta's home page viewed through Magpie. Known *people*, *organisations*, *projects* and *research areas* are highlighted using the Magpie toolbar (marked by '*'). On the right-hand side are three Magpie collectors – the top two log the people and projects found in the browsing session. The bottom one shows the (not explicitly mentioned) projects associated with the people found.

Figure 2. Details of the Magpie toolbar after selecting '*People*' and '*Project*' entity types (classes). The button labeled '*Services*' (marked by '♠') toggles a right-click 'Semantic services' menu in the browser.

tions are displayed. The journalist selects an option labeled 'Shares Research Areas With' to get an answer to her question. Magpie responds to this request by displaying projects that share one or more research areas with ScholOnto. The results are ordered – by the number of common research areas and alphabetically (as shown in Figure 3B, foreground window). The journalist notices that two of the projects related to ScholOnto – Climate Prediction and Magpie, appear also in the third collector, and are therefore related to Enrico. She decides to view the Climate Prediction project's web page by selecting 'Climate Prediction' in the collector, and selecting the 'Web Page' option in the displayed menu. Selecting items in collectors brings up the same semantic services menu as when items are selected on a web page.

3. MAGPIE DESIGN PRINCIPLES

The overall goal in designing Magpie is to support the interpretation of arbitrary web documents through the addition of an ontology-derived semantic layer. Let us now unpack this overall goal into a set of design principles. These principles may be treated as high-level functional requirements for a tool providing ontology based sensemaking support for navigating the web. The implementation of the individual principles is detailed in sections 4 and 5. Each principle is listed below together with an applicable part of the scenario that provides a justification for it:

- Magpie should run in and extend a *standard web browser* – we want to minimize the steps that users go through to use our tool; also many large organizations mandate a specific web browser.
- Magpie should *preserve the appearance* of a web page – users would quickly get confused if web pages browsed through a semantic browser did not look the same as when browsed traditionally.
- Magpie should *separate* the mark-up (the populated ontology) from the documents – this enables different viewpoints (from different communities) to be layered on top of the same web resources.
- Magpie should work with *any web page* – this means that it should work without the aid of manual pre-processing or relying on richly marked-up content (e.g. XML or RDF).
- Magpie users should not incur any significant *time penalty* – downloading a web page through Magpie should take only one or two extra seconds.
- Magpie should support the *interactive mediation* of semantic content through customizable *semantic services* – in effect the populated ontology becomes an auxiliary knowledge source, which supports the user in interpreting web documents and in information gathering tasks.

4. MAGPIE ARCHITECTURE

The architecture of Magpie is shown in Figure 4. Magpie acts as a bridge – a *mediator* between formally structured ontology-based descriptions and semantically unstructured HTML documents. The Magpie server provides HTTP access to a library of knowledge models consisting of domain ontologies, populated knowledge bases, semantic services and a semantic log KB. HTTP access is handled by a customized web server [25], which offers a library of high-level functions to reason about the content and dynamically generate appropriate HTML pages. Magpie accepts ontologies represented in RDF(S) [2], DAML+OIL [5], Ontolingua [10] and OCML [20]. The latter is the representation used internally by Magpie to perform reasoning. Within the forthcoming year, we intend to include ontologies represented in OWL [24]. The services, such as those shown in Figure 3A and Figure 3B, are defined in a *Services* module of the Magpie server. The *semantic log KB* – the last component of the server, is used by the Magpie trigger services, which are described in section 5.2.

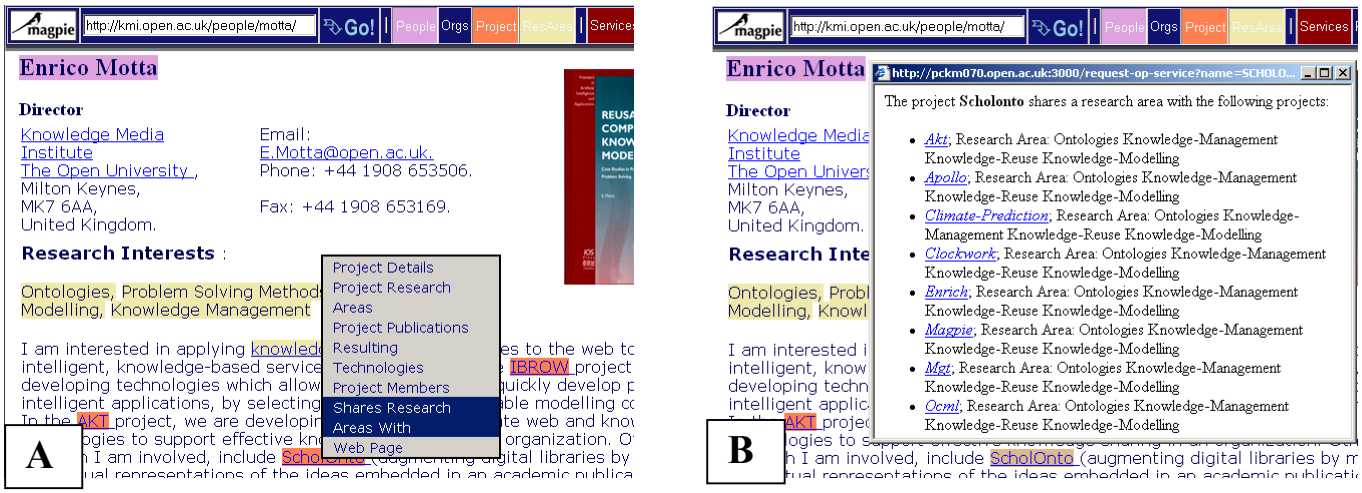

Figure 3. A. Specific 'semantic services' menu associated with the selected concept '*ScholOnto*'. Choices displayed depend on the class of the item selected – in this case a '*Project*'. B. Results of the '*Shares Research Areas With*' semantic query for the '*ScholOnto*' project.

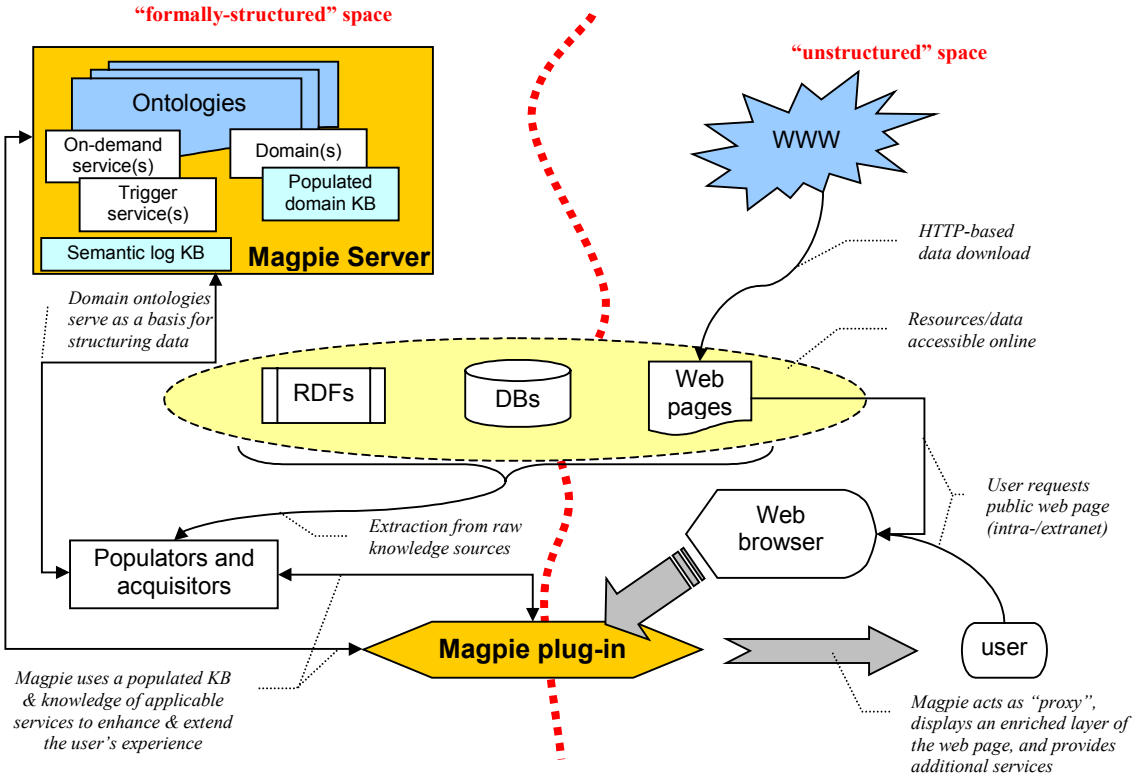

Figure 4. Overall architecture of the Magpie framework for semantic browsing

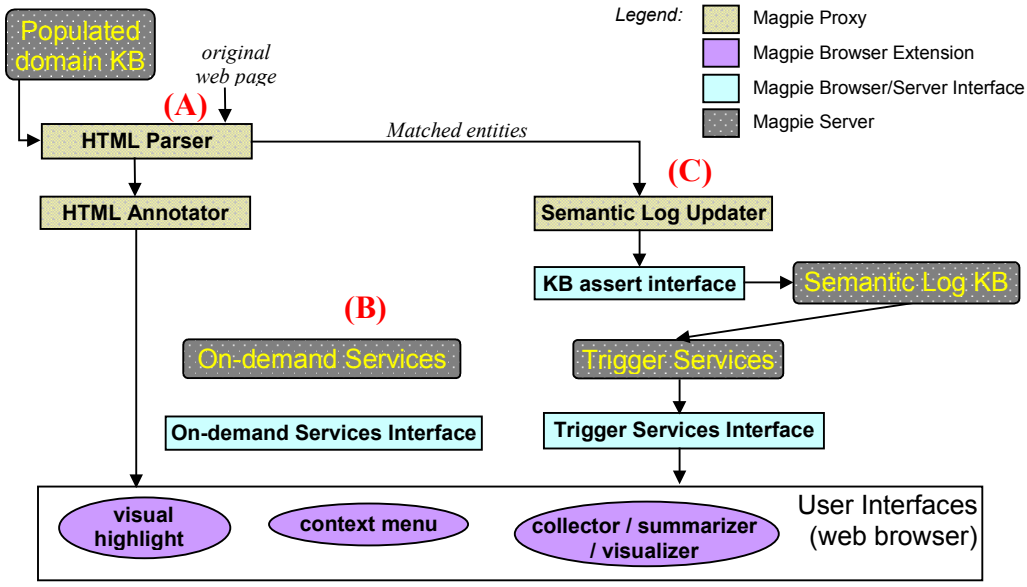

Figure 5. Interaction model of the Magpie plug-in architecture

A set of techniques (populators) is used to populate the ontology from heterogeneous data stored in web accessible RDF documents, ODBC compliant databases and from standard web pages. Details of the ontology population process are given in section 6.

4.1 Magpie Plug-In

We will now describe the Magpie plug-in in detail. As can be seen in Figure 5, the architecture of the Magpie plug-in is broadly composed of three main parts:

- *The Magpie Proxy* – this component is responsible for parsing HTML-based web pages and annotating them according to an ontology-derived lexicon. All this takes place on the fly, before a document is displayed in a user's browser.
- *The Magpie Browser Extension* – this part of the plug-in sits in the browser and controls all the interactions with Magpie. Specifically, it contains the user interface com-

ponents which visualize entities found in a web page and enables users to interact with the semantic services

- *The Magpie Browser/Server Interface* – this component mediates between the Magpie Browser Extension and the Magpie Server. It handles both user-requested (from a right mouse click) and trigger semantic services.

The Magpie proxy has two parts. The HTML parser parses incoming web pages and applies pre-defined tag-specific parsing rules to each HTML tag type. The content of the web page is matched against an ontology-derived lexicon residing on the Magpie server. Simple generic transformation rules generate lexicon entries from the instances within the populated ontological knowledge base. Additionally, ontology specific transformation rules can be defined. The scenario shown in section 2 uses the AKT reference ontology[1], an ontology, which describes academic life. The top-level classes in this ontology have `pretty-name` and `variant-names` slots, which are used to generate the lexicon, and consequently, to recognize the concepts of interest on a web page. In addition, some common-sense variants to peoples' names are generated, for example "Motta, E." from "Enrico Motta". We are currently investigating how robust named entity extraction mechanisms, such as those found in Amilcare [4], can be incorporated into the parser to enhance its capabilities.

Once an entity has been recognized in the web page, the second component of the Magpie proxy – the HTML annotator, embeds it in a customized `<SPAN…>` tag identifying the relevant instance and its class within the chosen ontology. Following the principle of not altering the appearance of web pages the new tags are initially not visible. If the user moves the mouse over a semantically layered tag, or if the class of entities is selected in the Magpie toolbar (see Figure 2), the corresponding text on the page is highlighted. Our approach to visualizing the semantically layered tags means that users remain in control of the types of entity that are visible at any time. We strongly believe that this improves navigation through the content. Recognized entities are then passed to a *Semantic Log Updater* (a component from the Magpie browser/server interface), where, through the *KB Assert Interface,* they are asserted into the *Semantic Log KB* that resides on the Magpie server. The purpose of semantic logging is addressed later in section 5.2.

The Magpie Browser Extension incorporates user interface components, which allow users to interact with the semantic services. First, the *Visual Highlighter* highlights the matched entities found in web pages. Second, the semantic services menu (shown in Figure 3A) is generated by the *context* component that closely liases with the *Services* module of the Magpie server. Finally, the Magpie *trigger* services are provided by the *Collector*, *Summarizer* and *Visualizer* components of the browser extension. Trigger services and these components are described in section 5.2. The Magpie Browser Extension also includes the main Magpie toolbar shown in Figure 2.

The Magpie Browser/Server interface handles all interactions between the Magpie server, which holds the ontological semantic knowledge (including knowledge of available semantic services), and the user interfaces in the Magpie Browser extension. We should emphasize (again) that within Magpie the ontology provides a specific viewpoint onto the web. We envisage that users will select a particular ontology depending on their current task. For example, the same set of web resources may cater for the very different interests of a graphic design community or a JavaScript programming community. Obviously, these two groups would have different viewpoints onto the 'shared' web resources.

[1] More information available at http://www.aktors.org

4.2 Dealing with Lexical Clashes

As we mentioned earlier the lexicon used for matching against items in a web page is generated from an ontology during a setup phase. We will now discuss how Magpie deals with lexical clashes – when two distinct entities within an ontology generate the same lexical term. The Magpie toolbar contains the names of the important or interesting classes for the user. In our scenario, described in section 2, the interesting concepts were People, Projects, Organizations and Research areas. Before outlining how lexical clashes are dealt with, it should be noted that each of the top level classes, displayed in the Magpie toolbar, typically contains sub-classes. For example, class 'People' contains sub-classes such as 'Academic' and 'Professor'.

Two types of lexical clashes can occur, and each of them would be treated slightly differently. The types of clashes are:

1. the entities are instances of different top level classes, for example, a Project and Organization;
2. the entities are instances of the same top level class;

If a clash of the first type occurs, the ontology designer deploying Magpie (see section 6 for details on the deployment of Magpie) is offered two options:

- one entity takes precedence over the other, or
- the end-user is asked to indicate, which top-level class the matched item belongs to.

Clashes of the second type result in the following two options for the ontology engineer:

- one entity takes precedence over the other, or
- any semantic services carried out on the term return the combined results for both instances.

Within the OCML representation, instances of different classes can have the same name. With Magpie it is also possible for instances of the same class to generate a lexical clash. For example, instances with the names `enrico-motta` and `emanuela-motta` would both generate a 'common sense' pretty name "Motta, E." according to the heuristics on variants to peoples' names used in our scenario example (see section 4.1).

5. SEMANTIC SERVICES

In section 4, we presented the conceptual architecture of Magpie, and showed how a semantic layer is created, displayed and activated. The main benefits of using Magpie however are generated from the ability to deploy semantic services on top of the semantic layer. These services are provided to the user as a physically independent layer generated on top of a particular HTML document. Magpie distinguishes between two types of semantic services, each having a specific user interaction model. Services are shown in Figure 5 as interaction threads (B) and (C).

According to the process model in Figure 5, the parser identifies entities from a chosen ontological KB in the original web document (thread A). Each discovered entity is annotated and recorded in a semantic log, and the annotated document is displayed in the user's web browser. The annotation has been described in section

4.1, semantic logging will be addressed in section 5.2. Thread (B) represents services activated on a user's request; i.e. a reader explicitly selects an entity s/he is interested in, and using a right mouse click invokes a *services menu*. This layer of *on-demand* semantic services is described in section 5.1. Alternatively, semantic services may be based upon certain patterns or *footprints* of the entities that co-occur in a particular document or during a particular browsing session. In section 5.2, we refer to this history-based functionality as *trigger services*.

5.1 On-Demand Semantic Services

As already mentioned, semantic services are enabled by clicking on the 'Services' button in the Magpie toolbar (see the marker '♠' in Figure 2). When the semantic services are activated, the contextual (right-click) menu of a web browser is overridden by an *on-demand services menu*. The 'on-demand services' menu is also context-dependent as could be expected; however, in this case, we are dealing with a semantic context defined by the membership of a particular entity to a particular ontological class. The lexicon containing the information on these memberships is generated from an ontology in a setup phase. Details on the lexicon generation process can be found in sections 4.1 and 4.2.

In addition to domain ontologies, Magpie also uses a *Services* module (see a box labeled 'Services' in the top-left corner of Figure 5). This knowledge model formally defines what operations can be performed on particular class(es) of domain entities, and the semantics of each operation. In the scenario of using Magpie as a semantic portal for organizational research, the services for ontological class *project* included the following operations (see menu displayed in Figure 3A):

- Show a project's details;
- Show the research areas tackled by a particular project;
- Show the project publications and bibliographic data;
- Show the technologies resulting from a project;
- Show projects tackling *similar* issues (share research scope), and finally
- Show the project website.

As was the case with document parsing and annotation, the 'on-demand services' menu is also generated on the fly. When a right click occurs, it is handled by the Magpie browser extension first, and consequently, through the Magpie browser/server interface the Magpie server is asked for the services that are available for a particular entity. The interface deploys the information about class membership of a particular entity that was created in the process of annotating the content (see section 4.1). If the list of applicable semantic services is non-empty, Magpie displays the services as a menu, and lets the user choose what s/he is interested in. The user selects a particular option, and this leads to another request to the '*Services*' module of the Magpie server to perform the requested reasoning and/or execute the applicable method. The knowledge-level reasoning provides the requested context for a particular entity. This is delivered to the web browser, is annotated in the Magpie proxy (as any other web page), and finally, is displayed in a dedicated browser window. The user can then semantically browse the results similarly as with any other web page.

Thus, Magpie provides two complementary methods for web browsing. First, it implements syntactic browsing by following the `<A HREF=…>` anchors inserted into a web document by its author. A document accessed via such a tag is treated as described in section 4; i.e. it is parsed, annotated, and displayed with a Magpie toolbar to facilitate semantically enriched user interaction. The second browsing mechanism in Magpie follows the customized semantic anchors created during the processing phase by the Magpie proxy, and the applicable, dynamically generated semantic services. While the first method give access to physically linked content, the second method makes available the semantic context of a particular entity. The two methods are visually differentiated so that any confusion is minimized, and they provide complementary functionality. Figure 3A shows a sample semantic services menu for term 'ScholOnto' (which belongs to the 'Project' class). The semantic context corresponding to the user's request for similar projects, is displayed in Figure 3B, and in this case contains an ordered list of ontologically related projects that are displayed in a new browser window.

5.2 Trigger Semantic Services

User-requested (or on-demand) semantic services are not the only means for interacting with the relevant background knowledge. A number of researchers emphasize the importance of active or push services. Active services take many different forms – for example, activity critics within domain dependent design environments [11], or content guides in an online shopping context [7]. The main feature distinguishing the active services from the user-requested ones is that they tend to "look over the user's shoulder", gather useful facts, and present appropriate conclusions.

Active services are represented in Figure 5 by the interaction thread (C) on the right hand side. As can be seen, a pre-condition for having active services is to keep *history logs* of browsing, particularly a log of the recognized entities. The label 'browsing history' is more than appropriate because a log may accumulate findings not only from the current web page, but also from previously visited documents in the same browsing session.

As shown in Figure 5, the process of semantic logging runs in parallel with the web page annotation. While an annotated web page is displayed in a browser, the data from the log are delivered to the Magpie server component responsible for semantic log maintenance. The server asserts the data from the logs as *facts* into a 'working' knowledge base. Several *watchers* are designed to monitor and respond to patterns in the asserted facts. When the required assertions have been made for a particular watcher, a semantic service response is *triggered* and applicable information is delivered to the dedicated window of the user's web browser. A few examples of the results of a trigger service firing are shown on the right-hand side of Figure 1 ('People', 'Projects' and 'People's Projects' collector windows).

Figure 6 shows the definition for the `collect-peoples-projects` watcher that is a part of the trigger service `peoples-projects`, shown at the bottom right of Figure 1. Whenever, a web page is viewed in Magpie the Semantic Log Updater (see Figure 5) asserts `found-item` facts, for each of the items found, into the Semantic Log KB. The `collect-peoples-projects` watcher is triggered if a person is found in the log, and this person is a member of a project, which is not yet present in the log. When triggered, the project and the URL of the page that the person was found on are collected. Future work will enable Magpie users to create watchers through a direct manipulation interface.

```
(def-watcher collect-peoples-projects peoples-projects
  (found-item ?time ?address ?page-url ?person)
  (person ?person)
  (has-project-member ?project ?person)
  (not (found-item ?time2 ?address ?page-url2 ?project))
  action
(collect ?project ?page-url)))
```

Figure 6. Watcher definition for "People's Projects" trigger service (see also bottom right collector in Figure 1).

The information that can be delivered in this fashion may range from a simple collection of relevant items to sophisticated guidance on browsing or browsing history visualization. Since the reasoning server component taps into a knowledge base constructed potentially from the logs of community members, the guidance or history visualization may draw on community knowledge and behaviors. This is an important benefit, especially if we follow the argument of Tauscher and Greenberg [26] mentioned in the introduction that 58% of all visits to web documents are to sites visited previously, but that history mechanisms are used infrequently. This significant number of re-occurring visits calls for a more sophisticated approach to the management of browsing histories. Indeed, one of design recommendations following from the study was that bookmarks should have a meaningful representation. We believe that history management based upon the semantics of the visited pages, and implemented by a *triggered* semantic layer would perform better than current, purely syntactic and linear (access time ordered) methods.

Although the design goal for our two types of services is the same – to provide users with additional knowledge sources to support the interpretation of web pages and to assist in information gathering – the underlying frameworks are different. The 'on-demand' services are only invoked by a specific user request. Backward chaining (goal based) reasoning from the user's query results in a response, which is typically presented as a new web page. The trigger services are invoked when a watcher matches a pattern within the semantic logs. The pattern causes forward chaining (data-driven) reasoning to occur. The results are displayed by a change in the interface. In Figure 1, trigger services result in additions to one of the three collectors. The differences between the two types of services are briefly summarized in Table 1.

Table 1. Magpie on-demand and trigger services

Feature	On-demand	Trigger
Source of service	User's request	Pattern in semantic log
Reasoning method	Backward chaining	Forward chaining
Delivery form	'Pull' (on-demand)	'Push' (triggered)

It is important to differentiate the Magpie Trigger services from the growing number of web logging tools that exist [28]. The goal of these web-logging tools is to monitor user's activity with the intention to measure the usability of a particular web site. Within the Magpie architecture, the semantic logs are kept to provide trigger services that could support the interpretation of a web document and/or information gathering.

6. BEFORE DEPLOYING MAGPIE

Deploying Magpie within a particular domain involves the following two steps that typically precede any meaningful user's interaction with the Magpie tool:

i) *Choosing or implementing an ontology* – within the Magpie context it is important that a) the ontology represents the intended viewpoint (with respect to the web resources, and b) the ontology is able to support the desired semantic services.

ii) *Population of the ontology* – the ontology itself is a skeleton for clarifying the domain structure; it should be populated with specific entities, which represent items of interest to the intended users. For scalability, the ontology population should be an automated process, and, when applicable, integrated with working practices. We discuss some techniques for population below.

If Magpie is targeted at a specific user community (e.g. members of a specific organization) it is important that the users are able to participate in the ontology design process. However, we firmly believe that the ontology should be implemented by knowledge engineers since the careful design of the ontology is crucial to ensure the success of any knowledge intensive system. In addition to specifying the communal viewpoint, the ontology circumscribes the range of phenomena we want to deal with, and defines the terminology used to acquire domain knowledge. In our experience, small errors/inconsistencies in any of these aspects can make the difference between success and failure. Moreover, ontology design requires specialist skills, which are normally not possessed by potential target user communities. Examples of how we have constructed ontologies for specific user communities in other domains are given in [7, 9, 21].

Once the ontology has been constructed, it needs to be populated; generic domain terms must be specified in detail. We currently use three techniques for ontology population, each of them suitable for different types of domain and content:

i) *Importing from online semantic resources* – as we mentioned earlier, Magpie can accept ontologies represented in RDF(S), DAML+OIL and Ontolingua.

ii) *Importing from an ODBC compliant database* – the Magpie server incorporates a tool for converting data stored within an ODBC compliant database into OCML instances.

iii) *By information extraction* – we also have used a dedicated tool – MnM [27] developed in-house, which enables information extraction engines, such as Amilcare [4] to be integrated with the Magpie knowledge model component.

We will now briefly outline how we used techniques ii) and iii) to populate the AKT reference ontology within the context of the Knowledge Media Institute.

Within our department, the secretarial staff are responsible for updating the people and project pages using simple web forms. Lab members are also able to edit their own entries or update entries for their projects. In our case, each form contains a list of research areas, which are identical to those found in the AKT reference ontology. Consequently, the structured information about each person or project is easy to translate into the terminology of the AKT reference ontology, and populate it automatically. More specifically, once every 24 hours a Magpie process uses an ODBC interface to create instances representing the people and projects within KMi and store them on the Magpie server.

MnM [27], is a tool, which incorporates a web browser and interfaces to web based ontology libraries and to information extraction engines. MnM enables users to attach information extraction mechanisms to ontological classes. As web pages are marked up a selected information extraction (IE) engine extracts items

according to a template derived from a class within a chosen ontology. Through a plug-in mechanism, MnM currently interfaces to the WebOnto [6] ontology server and is also able to read web accessible ontologies written in RDFS or DAML+OIL. MnM has been used to populate our ontology from our web based newsletter, KMi Planet [8]. KMi Planet contains stories about interesting events within our lab. Lab members write these stories as free text, and thus ontology population necessitates the use of IE technology.

Another technique, which is sometimes used for populating ontologies is *screen scraping* – using scripts to access the content of web pages. Screen scraping techniques have been used to construct an RDF resource encoding data from all UK computer science departments[2].

7. RELATED WORK

One of the inspirations for Magpie was the COHSE system [3]. COHSE combines an Open Hypermedia System with an ontology server, to provide a framework for ontological linking. As with Magpie, an ontology-derived lexicon is used to add links to arbitrary web pages. The links are added either by proxy server or by an augmented Mozilla™ browser. The distinctions between Magpie and COHSE are due to their differing design goals. The design goals for COHSE were to a) separate web page links from the web pages and b) to make these links conceptual – i.e. they could be potentially generated from ontology. The overall goal for Magpie is to support interpretation and information gathering. Thus, Magpie's interface enables ontological differences to be highlighted easily using the toolbars, and the services provided are dependent on the class of entity found. Magpie also offers *trigger* services dependent on the semantic log. Neither type of Magpie service is intended to replace traditional links; they are designed to be used as an auxiliary source of knowledge available at the user's fingertips.

In the last few years, a number of tools have emerged which support the annotation of web pages. A classic example is the Amaya HTML editor, which implements the Annotea infrastructure [17]. Annotea facilitates the RDF-based mark-up of web documents as they are created. In other words, authors or viewers may add various meta-statements to a document, which are separate from the document itself and are accessible to collaborating teams via a centralized annotation server. The annotation in this sense centres on attaching additional information to a chunk of content on an arbitrary web page. This feature of Annotea makes it a powerful tool for the joint authoring of documents where a small group of collaborating agents share a common goal. However, the same feature may make it more difficult to facilitate a similar form of annotation sharing in 'open' user communities. In these cases, there is no guarantee that a freely articulated annotation would convey the same meaning to the different users.

Another difference of the Annotea framework as compared to Magpie approach is in the source of annotations. Annotea assumes that at least one author (human) is willing to invest additional effort into making a page semantically richer. Magpie, on the other hand, is more liberal and assumes a reader subscribes to a particular domain ontology, which is then used to provide relevant background knowledge. It may be argued that ontology creation takes even more effort than simple document annotation. This is true; however, an ontology is a domain model, a shared viewpoint that can be used for many different purposes, not solely for the annotation of a single document. Thus, the effort spent on designing a shared ontology may be greater in the short term but in the longer term, it is a more cost-effective way of recording a shared point of view.

A similar approach to annotating formally structured documents can be found in other research projects. For example, the CEDAR toolkit [22] used a similar strategy for rich web publishing and the construction of organizational memories using a shared organizational ontology. The CREAM-based Ont-O-Mat/Annotizer [14] is a tool similar to MnM (described in section 6), which integrates ontologies and information extraction technologies. As with MnM, Amilcare [4] provides information extraction support, and ontologies are represented in DAML+OIL. Annotations, as understood in this framework, are very close to those advocated in this paper. Basically, any ontological instance, attribute or relation known in a particular ontology may serve as an annotation. A key feature of Ont-O-Mat/Annotizer is its use of *discourse representations* to structure the relatively flat output of Amilcare according to the chosen ontology, thus facilitating the ontology population process.

The CREAM research team voice an important feature of ontology-based annotation and document enrichment. Namely, any annotating tool must be aware of already existing (i.e. recognized) entities and their relationships; otherwise harm can be done with redundancies and multiple definitions. CREAM's annotation inference resembles our trigger semantic services produced by a data-driven reasoning process. On the other hand, our mechanism of 'on-demand' services smoothly and seamlessly addersses the issue identified above – the awareness of the existing relationships and the actual context of any particular ontological instance.

The SHOE project [16] proposed an extension to HTML to allow the specification of ontological information within common HTML-based documents. In addition to the inclusion of a semantically rich, ontological knowledge, SHOE also invested significant effort into making these inclusions re-usable and understandable 'throughout the web'. An editor was developed to support the page annotation process. As with the tools mentioned in the previous paragraphs, and unlike our Magpie framework, SHOE relies on the offline annotation or mark-up of web documents. Once that is accomplished, the enriched documents are published on the web and dedicated tools may make use of the enriched contextual knowledge (e.g. Exposé web crawler [15]).

8. CONCLUSIONS

Reducing the information overload caused by the growing web is often cited as the premise for work on supporting the retrieval of relevant documents. But finding relevant documents is only half of the story. Interpreting a returned document involves a reader in understanding the surrounding context in which the document was created. In order to gain the full understanding of a document a reader will require knowledge of the specific terms mentioned and the implicit relationships contained both within the document and between the document and other external knowledge sources. Magpie addresses this issue by capturing context within an ontology, which then is used to enrich web documents with a semantic layer. Semantic services expose relevant segments of the ontology according to the user's needs.

As we described in section 3, certain design criteria are critical if semantic layering is to be both useful and usable. If desired Mag-

[2] More information available from http://www.hyphen.info

pie users are able to browse the web in a standard way with negligible differences in the user experience. Magpie can achieve this because it works in standard web browsers with standard mark-up languages, presents web pages without altering their layout or appearance, and involves only a small time overhead - the Magpie proxy server takes less than 1 second to parse even relatively large web pages.

Another key set of principles underlying the design of Magpie is that the user is able to control to what extent semantic browsing comes to the fore. The Magpie toolbar enables terms to be made visible according to their ontological category. The Magpie framework also enables arbitrary semantic actions to be triggered on patterns of items found within a semantic log. Trigger services allow certain types of tasks to be delegated. The simple trigger services shown in the scenario enable semantically designated types of entities to be collected for later inspection. More complex trigger services can be implemented easily in the Magpie framework. For example, the Magpie proxy has an option to automatically parse web pages linked to the current page. This allows reconnaissance services [18] to be set up, which would alert the user whenever an interesting neighbouring page was identified.

As we mentioned in the introduction, Magpie addresses some of the issues found by Tauscher and Greenberg in their study [26] of how web pages are browsed. A conclusion of their work was that users need better support in managing the histories of visited web pages. The semantic log in Magpie can help in this as it forms the basis for *semantic bookmarks*. Using Magpie previous pages can be found by a semantic query based on the types of entities found in the page. For example, one could look for "a web page I saw early last week which mentioned a semantic web project funded by a large Petroleum Company".

Attention as opposed to information is now widely acknowledged to be the scarce resource in the Internet age. Consequently, tools that can leverage semantic resources to take some of the burden of the interpretation task from the human reader are going to be of enormous use. We believe that Magpie is a step towards achieving this goal.

9. ACKNOWLEDGMENTS

The Magpie project 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 GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and The Open University.

This paper has benefited from numerous conversations with Marc Eisenstadt, Maria Vargas-Vera, Nigel Shadbolt, and other colleagues from KMi and the University of Southampton. The Magpie graphic design is courtesy of Harriett Cornish from KMi.

10. REFERENCES

[1] Berners-Lee, T., Hendler, J., and Lassila, O., *The Semantic Web.* Scientific American, 2001. **279**.

[2] Brickley, D. and Guha, R., *Resource Description Framework (RDF) Schema Specification*. 2000, World Wide Web Consortium. (URL: http://www.w3.org/TR/2000/CR-rdf-schema-20000327).

[3] Carr, L., Bechhofer, S., Goble, C., *et al. Conceptual Linking: Ontology-based Open Hypermedia*. In *Proc. of the 10th International WWW Conference*. 2001.

[4] Ciravegna, F. *Adaptive Information Extraction from Text by Rule Induction and Generalisation*. In *Proc. of 17th International Joint Conference on AI*. 2001. Washington, USA.

[5] DAML.org, *Reference description of the DAML+OIL ontology mark-up language*. 2001, http://www.DAML.org/2001/03/reference.html.

[6] Domingue, J. *Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web*. In *Proc. of 11th Knowledge Acquisition Workshop*. 1998. Banff, Canada.

[7] Domingue, J., Martins, M., Tan, J., *et al. Alice: Assisting Online Shoppers Through Ontologies and Novel Interface Metaphors*. In *Proc. of the 13th European Knowledge Acquisition Workshop (EKAW)*. 2002. Spain.

[8] Domingue, J. and Scott, P. *KMi Planet: A Web Based News Server*. In *Proc. of Asia-Pacific Computer Human Interaction Conference*. 1998. Japan.

[9] Dzbor, M., Paralic, J., and Paralic, M. *Knowledge Management in a Distributed Organisation*. In *Proc. of the 4th IEEE/IFIP Conference on IT for Balanced Systems (BASYS'2000)*. 2000. Berlin, Germany.

[10] Farquhar, A., Fikes, R., and Rice, J. *The Ontolingua Server: a Tool for Collaborative Ontology Construction*. In *Proc. of Knowledge Acquisition Workshop*. 1996. Banff, Canada.

[11] Fischer, G. *Domain-Oriented Design Environments*. In *Proc. of 7th Knowledge-Based Software Engineering Conference (KBSE'92)*. 1992: IEEE Computer Society.

[12] Gruber, T.R. *A Translation approach to portable ontology specifications* Knowledge Acquisition, 1993. **5**(2):p.199-221

[13] Guarino, N., Masolo, C., and Vetere, G., *OntoSeek: Content-Based Access to the Web.* IEEE Intelligent Systems, 1999. **14**(3): p.70-80.

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

[15] Heflin, J. and Hendler, J., *A Portrait of the Semantic Web in Action.* IEEE Intelligent Systems, 2001. **16**(2): p.54-59.

[16] Heflin, J., Hendler, J., and Luke, S. *Reading Between the Lines: Using SHOE to Discover Implicit Knowledge from the Web*. In *Proc. of AAAI-98 Workshop on AI and Information Integration*. 1998.

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

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

[19] McGuinness, D.L. *Ontological Issues for Knowledge-Enhanced Search*. In *Proceedings of Formal Ontology in Information Systems Conference*. 1998.

[20] Motta, E., *Reusable Components for Knowledge Modelling*. Frontiers in AI and Applications. 1997, The Netherlands: IOS Press.

[21] Motta, E., Buckingham Shum, S., and Domingue, J., *Ontology-Driven Document Enrichment: Principles, Tools and Applications.* International Journal of Human-Computer Studies, 2000. **52**(5): p.1071-1109.

[22] Mulholland, P., Zdrahal, Z., Domingue, J., *et al.*, *Integrating working and learning: a document enrichment approach.* Journal of Behaviour and Information Technology, 2000. **19**(3): p.171-180.

[23] Ovsiannikov, I.A., Arbib, M.A., and Mcneill, T.H., *Annotation Technology.* International Journal of Human-Computer Studies, 1999. **50**: p.329-362.

[24] Patel-Schneider, P.F., Horrocks, I., and van Harmelen, F., *OWL Web Ontology Language 1.0 Abstract Syntax*. 2002, (URL http://www.w3.org/TR/owl-absyn/).

[25] Riva, A. and Ramoni, M., *LispWeb: A Specialised HTTP Server for Distributed AI Applications.* Computer Networks and ISDN Systems, 1996. **28**(7-11): p.953-961.

[26] Tauscher, L. and Greenberg, S., *How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems.* International Journal of Human Computer Studies, 2001. **47**(1): p.97-138.

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

[28] Zaiane, O.R., Xin, M., and Han, J. *Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs*. In *Advances in Digital Libraries (ADL)*. 1998. California, USA**.**