Evolving the Semantic Web with Mangrove

Luke McDowell, Oren Etzioni, Steven D. Gribble, Alon Halevy,
Henry Levy, William Pentney, Deepak Verma, and Stani Vlasseva
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195 U.S.A
{lucasm,etzioni,gribble,alon,levy,bill,deepak,stani}@cs.washington.edu

ABSTRACT

Despite numerous proposals for its creation, the *semantic web* has yet to achieve widespread adoption. Recently, some researchers have argued that participation in the semantic web is too difficult for "ordinary" people, limiting its growth and popularity.

In response, this paper introduces MANGROVE, a system whose goal is to evolve a portion of the semantic web from the enormous volume of facts *already* available in HTML documents. MANGROVE seeks to emulate three key conditions that contributed to the explosive growth of the web: ease of authoring, instant gratification for authors, and robustness of services to malformed and malicious information. In the HTML world, a newly authored page is immediately accessible through a browser; we mimic this feature in MANGROVE by making semantic content instantly available to services that consume the content and yield immediate, tangible benefit to authors.

We have designed and implemented a MANGROVE prototype, built several semantic services for the system, and deployed those services in our department. This paper describes MANGROVE's goals, presents the system architecture, and reports on our implementation and deployment experience. Overall, MANGROVE demonstrates a concrete path for enabling and enticing non-technical people to enter the semantic web.

Categories and Subject Descriptors

H.3.m [**Information Storage and Retreival**]: Miscellaneous;
C.2.4 [**Computer Communication Networks**]: Distributed Systems

Keywords

Semantic Web, applications, gradual evolution, structured data, RDF, annotation, tagging, instant gratification

1. INTRODUCTION AND MOTIVATION

Today's web was built to present documents for humans to view, rather than to provide data for software-based processing and querying. In response, numerous proposals for creating a *semantic web* have been made in recent years (e.g., [4, 8, 24]), yet adoption of the semantic web is far from widespread.

Several researchers have recently questioned whether participation in the semantic web is too difficult for "ordinary" people [13, 35, 23]. Indeed, a key barrier to the growth of the semantic web is the need to *structure* data: technical sophistication and substantial effort are required whether one is creating a database schema or authoring an ontology. The database and knowledge representation communities have long ago recognized this challenge as a barrier to the widespread adoption of their powerful technologies. The semantic web exacerbates this problem, as the vision calls for large-scale and decentralized authoring of structured data. As a result, the creation of the semantic web is sometimes viewed as a discontinuous divergence from today's web-authoring practices — technically sophisticated people will use complex tools to create new ontologies and services.

While the discontinuous approach will certainly yield many useful semantic web services, this paper is concerned with the *evolutionary approach: a significant part of the semantic web can evolve naturally and gradually as non-technical people structure their existing HTML content.* In fact, the two approaches are not competing but complementary. Each focuses on a different user base, data sources, and services. Furthermore, each approach gives rise to a somewhat different set of challenges. A key question for the evolutionary approach is *how do we entice people to structure their data?* Structuring must be made *easy*, *incremental*, and *rewarding*; it must not require duplicating existing HTML content, and must support easy maintenance of consistency between related HTML and semantic information over time.

This paper presents the architecture of MANGROVE, a semantic web system that embodies an evolutionary approach to semantic content creation and processing. In particular, MANGROVE seeks to emulate three key conditions that contributed to the explosive growth of the web. The first condition is *ease of authoring:* MANGROVE includes a convenient syntax for semantic markup, accompanied by a graphical web-page tagger that enables users to tag existing HTML content without having to replicate any data. The second condition is *instant gratification:* in the HTML world, a newly authored page is immediately accessible through a browser; we mimic this feature in MANGROVE by making tagged content instantly available to services. We posit that semantic tagging will be motivated by services that consume the tags and result in immediate, tangible benefit to authors. MANGROVE provides several such services and the infrastructure to create additional ones over time. The third condition is *robustness:* when authoring an HTML page, authors are not forced to consider the contents of other, pre-existing pages. Similarly, MANGROVE does not require authors of semantic content to obey integrity constraints, such as data uniqueness or consistency. Data cleaning is deferred to the services that consume the data.

As one example of the MANGROVE approach, consider the homepage of an individual, a university course, or a small organization. Such pages contain numerous facts including contact information, locations, schedules, publications, and relationships to other information. If this information could be easily tagged *without* disrupting standard web activities, then the *very same* pages and text could be used to support both the semantic web and standard

Technical Report UW-CSE-03-02-01
February 2003
.

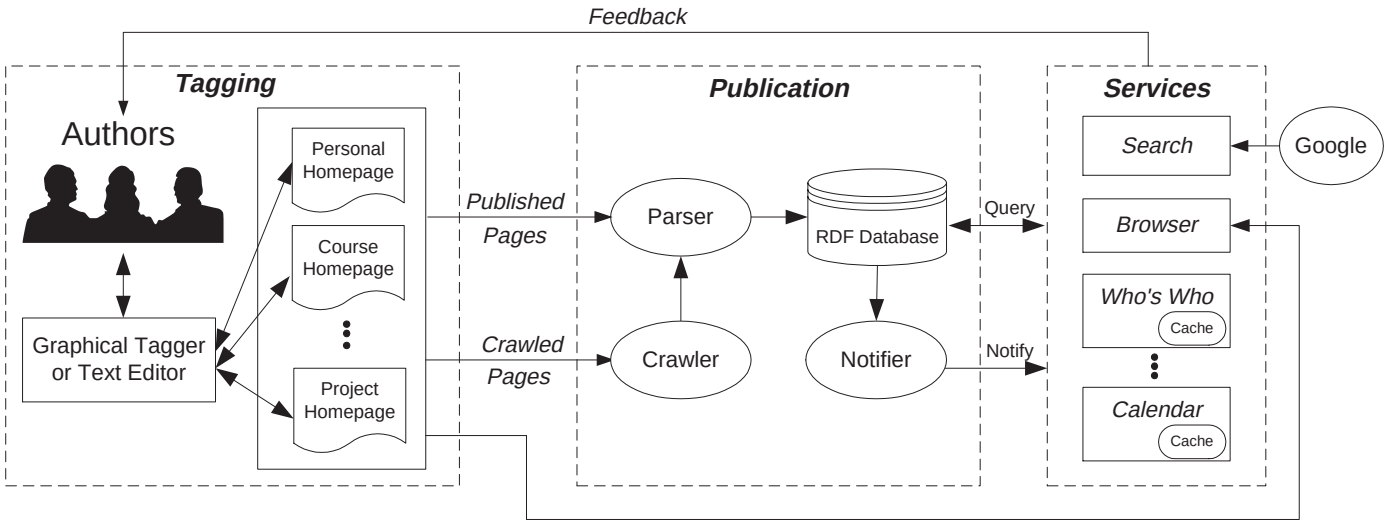

Figure 1: The MANGROVE architecture and sample services.

HTML-based browsing and searching. For example, we can easily produce a departmental phone list by extracting phone numbers from semantically-tagged home pages for the faculty and students in our department. Similarly, we have created a departmental calendar that draws on tagged information found on existing web pages, which describe courses, seminars, and other events. Both services instantly consume tagged facts and help motivate people to participate in the semantic web.

This paper uses MANGROVE as a testbed to answer several fundamental architectural questions about the support of semantic content on the web. We also describe our initial experience with MANGROVE services and present measurements that help us to characterize the behavior of our prototype implementation. Overall, we aim to introduce the MANGROVE architecture and show how its novel features support the evolutionary approach to semantic web creation.

The remainder of this paper is organized as follows. The next section introduces MANGROVE's architecture and explains how it supports instant gratification. Section 3 describes our initial semantic services and explains some of the ways that they facilitate the gradual evolution of the web. Section 4 examines how MANGROVE maintains robustness in the presence of malformed or malicious data, while Section 5 presents some of our initial experience using and evaluating our MANGROVE prototype. Section 6 discusses related work on this problem, and Section 7 concludes.

2. THE ARCHITECTURE OF MANGROVE

This section presents the high-level architecture of the MANGROVE system, details some of the key components, and relates the architecture and design to MANGROVE's goals. As noted earlier, MANGROVE seeks to enable and entice authors to structure their data, by mimicking the conditions that led to the explosive growth of content creation on the web. In particular, we aim to provide:

- **Ease of Authoring:** we aim to enable today's web authors to conveniently and incrementally transform their billion or more web pages into semantic web content without disrupting standard HTML-based authoring, browsing, and searching.
- **Instant Gratification:** we seek to motivate semantic tagging by providing an immediate, tangible benefit to authors from services that consume their tagged content.
- **Robustness:** in contrast with logic-based systems where contradictions result in failure, we seek to make MANGROVE robust to malformed data and to malicious misinformation.

As we consider the above principles, we also need to keep in mind the scalability of the system. Our goal is to enable distributed authoring at web scale and efficient access to the data once it is structured.

2.1 Architecture Overview

Figure 1 shows the architecture of MANGROVE organized around the following three phases of operation:

- **Tagging:** Authors use our *graphical tagger* or an editor to insert tags into existing HTML documents. The syntax we use for tagging, MTS, is explained in Section 2.2.1.
- **Publication:** Authors can explicitly *publish* tagged content, causing the *parser* to immediately parse and store the contents in a *RDF database*. The *notifier* then notifies registered services about relevant updates to this database. Services can then send *feedback* to the authors in the form of links to updated content (or diagnostic messages in case of errors). In addition, MANGROVE's *crawler* supplies data to the parser periodically, updating the database when authors forego explicit publishing.
- **Service Execution:** Newly published content is immediately available to a range of *services* that access the content via database queries. The services are representative of the ways users interact with the web today, but boosted by semantics: we support semantic browsing, search, and more complex services such as an automatically generated department calendar.

These three phases are overlapping and iterative. For instance, after tagging, publication, and service execution, an author may refine her tagged documents to add additional tags or to improve data

usage by the service. Supporting this complete life-cycle of content creation and consumption is important to fueling the semantic web development process.

Below, we describe each of the elements of MANGROVE's architecture in more detail. We focus on the novel elements and omit many aspects of components that use standard technology, such as our crawler, parser, and tagger.

2.2 Semantic Tagging in MANGROVE

Semantic tagging of HTML content is central to MANGROVE. In our context, manual tagging of a significant portion of the content is a requirement. We considered the use of "wrapper" technology (e.g., [12, 38]) for automatically extracting structured data from HTML. However, such technology relies on heavily regular structure; it is appropriate for *recovering* database structure that is obscured by HTML presentation (e.g., Amazon.com product descriptions that are automatically generated from a database), but not for analyzing pages authored by hand. Thus, MANGROVE utilizes a small number of wrappers to generate "seed" data to initially populate semantic services, but this is not sufficient for solving the general problem. MANGROVE also supports the use of wrappers to "read" semantic information from external databases and applications (e.g., an Exchange server containing calendar and contact information). We also considered natural language processing and, specifically, information extraction techniques (e.g., [49]), but such approaches are domain-specific and often unreliable. As a result, we formulated the more pragmatic approach described below.

2.2.1 *The* MTS *Syntax*

We developed MTS (the MANGROVE Tagging Syntax) to enable easy tagging while deviating as little as possible from existing HTML authoring practices. Ideally, we would have liked to use RDF for annotating data.[1] However, RDF is currently inadequate for our purposes, because RDF cannot intermingle with HTML tags; that is, existing data must be *replicated* in a separate RDF section (as also noted by [23]). Since HTML documents are frequently updated by their authors, this data replication can easily lead to inconsistency between the RDF and its data source, particularly if "semantically-unaware" tools (e.g., Microsoft FrontPage) are used for editing.

In essence, MTS is a syntax that enables authors to *embed* RDF within their HTML document.[2] As a consequence, MTS does not have the aforementioned redundancy problem, because HTML and MTS tags may interleave in any fashion, permitting "inline" annotation of the original data. Therefore, factual changes to the annotated data using any tool will result in a seamless update to the semantic content on the page. In contrast to much of the related work on semantic web languages, the focus in the design of MTS is on syntax and convenience rather than semantics and expressive power. MTS's expressive power is equivalent to that of basic RDF; for simplicity we omitted advanced RDF features such as containers and reification. Finally, we note that the goal of MTS is to express base data rather than models or ontologies of the domain (as in RDF Schema, DAML+OIL [26], OWL [7]).

MTS consists of a set of XML tags chosen from a simple local schema. The tags enclose HTML or plain text. For instance, a phone number that appears as "543-6158" on a web page would become "<uw:workPhone>543-6158</uw:workPhone>". Here "uw" is the namespace prefix for our local domain and "workPhone" is the *tag name*. Nested tags convey property information. For instance, Figure 2 shows a sample tagged document where the <workPhone> tag is a property of the <instructor> named "Prof. John Fitz." Our syntax also permits RDF-like *about* attributes (e.g., of the <course> element) that enable information about an object to appear in multiple locations and be fused later.

In order to enable users to tag existing data regardless of its HTML presentation, the MTS parser disregards HTML tags when parsing (though images and links are reproduced in the parser output to permit their use by services). Finally, in order to reduce the annotation burden for items such as lists and tables that exist inside HTML documents, MANGROVE provides a simple regular expression syntax. For instance, the <reglist> element in Figure 2 enables the table to be automatically tagged based on existing HTML patterns.

```
<html xmlns:uw="http://wash.edu/schema/example">
<uw:course about="http://wash.edu/courses/692">
<h1><uw:name>Networking Seminar
    </uw:name></h1>

<p>Office hours for additional assistance:
<uw:instructor>
    <uw:name><b>Prof.</b> John Fitz</uw:name>
    <uw:workPhone>543-6158</uw:workPhone>
</uw:instructor>
<uw:instructor>
    <uw:name><b>Prof.</b> Helen Long</uw:name>
    <uw:workPhone>543-8312</uw:workPhone>
</uw:instructor>

<table> <tr><th>2003 Schedule</tr>
   <uw:reglist=
      '<tr><uw:event>
         <td><uw:date>*</uw:date>
         <td><uw:topic>*</uw:topic>
      </uw:event></tr>'>
      <tr> <td>Jan 11 <td>Packet loss</tr>
      <tr> <td>Jan 18 <td>TCP theory</tr>
  </uw:reglist>
</table>
</uw:course> </html>
```

Figure 2: Example of annotated HTML. The uw: tags provide semantic information without disrupting normal HTML browsing. The <reglist> element specifies a regular expression where '*' indicates the text to be enclosed in MTS tags.

2.2.2 *The Graphical Tagger*

To facilitate semantic tagging, we developed the simple graphical tagger shown in Figure 3. The tool displays a rendered version of the HTML document alongside a tree view of the relevant schema. Users highlight portions of the HTML document, and the tool automatically pops up a list of MTS tags that may be selected, based on the current annotation context. The tool also displays a simplified tree version of the tagged portion of the document, showing the value of each property. This enables the author to easily verify the semantic interpretation of the document or, by clicking on a node in the tree, to browse through the document based on its semantics. The graphical tagger can be downloaded from http://www.cs.washington.edu/research/semweb/tagger.html.

2.2.3 *Schema in* MANGROVE

Currently, MANGROVE provides a predefined set of XML schemas to support the tagging process. Providing schemas is crucial, as we can't expect casual users to design their own (and that would certainly not entice people to use the system). The intent

[1] We use the terms 'tag' and 'annotate' interchangeably.

[2] In fact, the MTS parser converts MTS into RDF for storage in our RDF database.

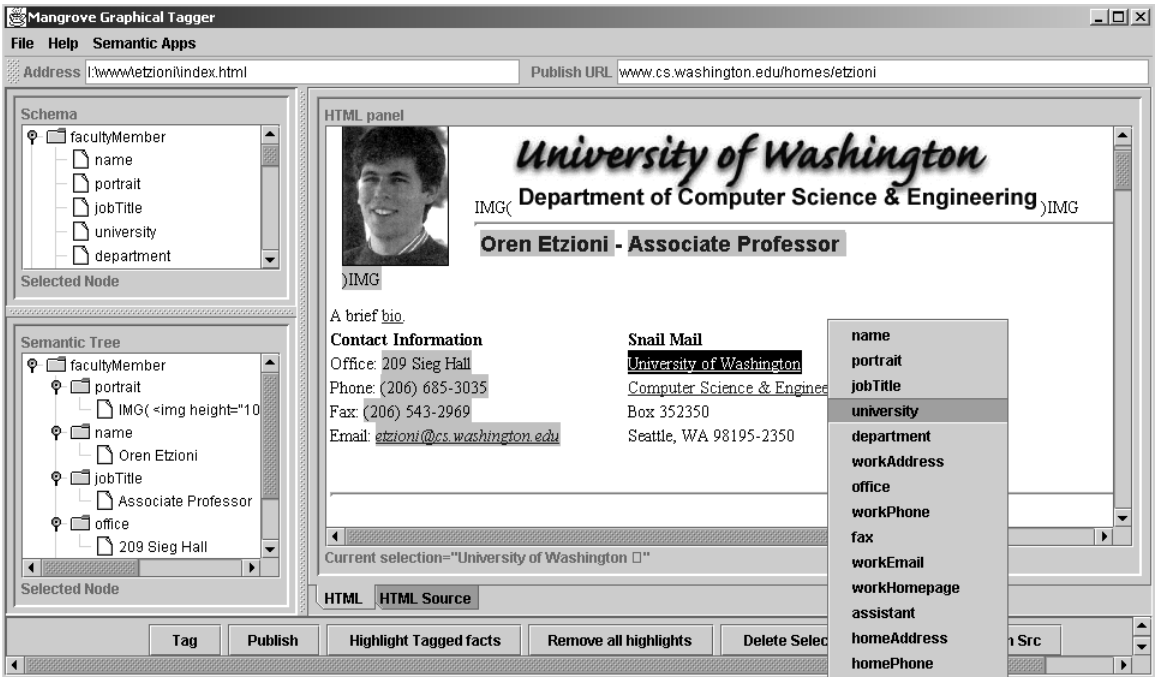

Figure 3: The MANGROVE graphical tagger. The pop-up box presents the set of tags that are valid for tagging the highlighted text. Items in gray have been tagged already, and their semantic interpretation is shown in the "Semantic Tree" pane on the lower left. The user can navigate the schema in the upper left pane.

of the schemas is to capture most aspects of the domain of interest. The pages being annotated do not have to contain *all* the details of a certain schema, and more importantly, they do not have to conform to the nesting structure of a particular XML schema. Instead, they map the data on their page to the appropriate tags in the schema. In the future, we anticipate a process by which users can collectively evolve the schema as necessary.

2.3 Document Publication

In today's web, changes to a web page are immediately visible through a browser. We create the analogous experience in MANGROVE by *publishing* semantically tagged content, which instantly transmits that content to MANGROVE's database and from there to services that consume the content.

MANGROVE authors have two simple interfaces for publishing their pages. They can publish by pressing a button in the graphical tagger, or they can enter the URL of a tagged page into a web form. Both interfaces send the URL to MANGROVE's parser, which fetches the document, parses it for semantic content, and stores that content in the RDF database. This mechanism ensures that users can immediately view the output of relevant services, updated with their newly published data, and then iterate either to achieve different results or to further annotate their data. In addition, before adding new content, the database purges any previously published information from the corresponding URL, allowing users to retract previously published information (e.g., if an event is canceled).

Some services, such as search, compute their entire result in response to a user's input. These services obtain data solely through RDF database queries. Other services, however, maintain a cache of precomputed information in order to provide faster response to user queries. These services specify data of interest by providing a query to the MANGROVE notifier. When the database is updated by a new data publication or a web crawl, the notifier forwards data matching that query to the corresponding services for processing. For instance, the calendar service registers its interest in all pages that contain `<event>` tags (or that had such tags deleted). When it receives notification of relevant new data, the calendar processes that data and updates its internal data structures, ensuring that viewers of the calendar see fresh data with minimal delay.

Crawling or polling all potentially relevant pages is an obvious alternative to explicit publication. While MANGROVE does utilize a crawler, it seems clear that crawling is insufficient given a reasonable crawling schedule. This is an important difference between MANGROVE and current systems (e.g., [8, 24]) that do not attempt to support instant gratification and so can afford to rely exclusively on crawlers. MANGROVE's web crawler regularly revisits all pages that have been previously published, as well as all pages in a circumscribed domain (e.g., `cs.washington.edu`). The crawler enables MANGROVE to find semantic information that a user neglected to publish. Thus, publication supports instant gratification as desired, while web crawls provide a convenient backup in case of errors or when timeliness is less important.

Conceivably, we could leave the data in the HTML files and access them only at query time. In fact, several data integration systems (e.g., [17, 19, 1, 31, 28]) do exactly this type of polling. The difference between MANGROVE and data integration systems is that in the latter, the system is given *descriptions* of the contents of every data source. At query time, a data integration system can therefore prune the sources examined to only the relevant ones (typically a small number). In MANGROVE we cannot anticipate *a priori* which data will be on a particular web page, and hence we would have to access every page for any given query – clearly not a scalable solution.

An additional reason why we chose publishing to a database over query-time access is that the number of queries is typically much higher than the number of publication actions. For example, people consult event information in the department calendar much more frequently than announcing new events or changing the

events' time or location.

2.4 Scaling MANGROVE

Scalability is an important design consideration for MANGROVE, and it has influenced several aspects of MANGROVE's architecture. For example, our explicit publish/notification mechanisms were chosen to improve scalability, as well as to support our instant gratification goals. Nevertheless, the scalability of our current prototype is limited in two respects. First, at the logical level, the system does not currently provide mechanisms for composing or translating between multiple schemas or ontologies. Second, at the physical level, the central database in which we store our data could become a bottleneck.

We address both scalability issues as part of a broader project described in [20]. Specifically, once a department has tagged its data according to a local schema, it can collaborate with other structured data sources using a *peer-data management system* (PDMS) [21]. In a PDMS, semantic relationships between data sources are provided using *schema mappings*, which enable the translation of queries posed on one source to the schema of the other. Our group has developed tools that assist in the construction of schema mappings [9, 10], though these tools are not yet integrated into MANGROVE.[3]

Given a network of peers and a query posed over one of them, the algorithms described in [21] will *chain* through the semantic mappings and obtain data from any relevant peer. In doing so, the system will only retrieve data that is relevant to the query. Relying on a PDMS resolves the potential bottleneck of querying a central database.[4]

3. SEMANTIC SERVICES IN MANGROVE

One of the goals of MANGROVE is to demonstrate that even modest amounts of tagging can significantly boost the utility of the web today. To illustrate this, MANGROVE supports a range of semantic services that represent several different web-interaction paradigms, including Google-style search, novel services that aggregate semantically tagged information, and semantic browsing. Below, we introduce our service construction template, and then consider each of the above services in turn.

3.1 Service Construction Template

Services are written in Java and built on top of a MANGROVE service template that provides the basic infrastructure needed for service creation. Currently, we use the Jena [36] RDF-based storage and querying system, which enables our services to pose RDF-style queries to extract basic semantic information from the database via a JDBC connection. Alternatively, services may use higher-level methods provided by the template. For instance, the template contains methods to retrieve all relevant information about a given resource from the RDF database, augmented with a summary of the sources of that information. The template also provides methods to assist with data cleaning and interpretation, as explained in Section 4.

The MANGROVE service template also aids service construction with support for incrementally computing and caching results. First, the template provides a standard `runUpdate()` method; this method is invoked by the MANGROVE notifier, which passes in a handle to the complete RDF dataset as well as a handle to the new RDF data for which the notification has occurred. Upon notification, invoked services rely primarily on the new data and local cached state, but an application can consult the complete dataset as necessary. Second, the template also provides a simple caching mechanism that maintains pre-computed information (e.g., processed event descriptions) and a mapping between each piece of information and its source page(s). For instance, when the calendar is invoked by the notifier, it uses those source mappings to determine what events may have changed, then updates the cache with the new information. The calendar viewer then uses this cache to quickly access the information requested by users.

Overall, MANGROVE makes services substantially easier to write by encapsulating commonly-used functionality in this service template.

3.2 Semantic Search

We believe that tagging will be an incremental process starting with "light" tagging of pages and gradually increasing in scope and sophistication as more services are developed to consume an increasing number of tags. It is important for this "chicken and egg" cycle that even light tagging yield tangible benefit to users. One important source of benefit is a Google-style search service that responds appropriately to search queries that freely mix tags and text. The service returns the set of web pages in our domain that contain the text and tags in the query.

The interface to the service is a web form that accepts standard textual search queries. The service also accepts queries such as "assistant professor" <facultyMember> <portrait>?, which combines the phrase "assistant professor" with MTS tags. Like Google, the query has an implicit AND semantics and returns exactly the set of pages in our domain containing the phrase "associate professor" and the specified tags. The `?` after the `<portrait>` tag instructs the service to extract and return the HTML inside that tag (as with the SELECT clause of a SQL query).

The service is implemented by sending the textual portion of the query (if any) to Google along with instructions to restrict the results to the local domain (`cs.washington.edu`). The MANGROVE database is queried to return the set of pages containing all the tags in the query (if any). The two result sets are then intersected to identify the relevant set of pages. When multiple relevant pages are present, their order in the Google results is preserved to enable more prominent pages to appear first in the list. Finally, any extraction operations indicated by one or more question marks in the query are performed and included in the result (see Figure 4). Like Google, not every result provides what the user was seeking; the search service includes *semantic context* with each result — a snippet that assists the user in understanding the context of the information that was extracted. The snippet is the `name` property of the enclosing object for the extracted tag. For instance, when extracting the `<portrait>` information as shown in Figure 4, the snippet is the name of the faculty member whose portrait is shown.

With its ability to mix text and tags, this kind of search is different from the standard querying capability supported by the MANGROVE database and other semantic web systems such as SHOE [24] and WebKB [34]. Our search service has value to users even when pages are only lightly tagged, supporting our design goal of gradual evolution. In Section 5 we attempt to quantify the added value of our semantic search.

3.3 Aggregation Services

In this section we describe some of the additional services that

[3] See [46] for a recent survey on the topic of schema mapping in the database literature.

[4] The peer-to-peer architecture of a PDMS accommodates as special cases several data-sharing architectures that have been discussed in the literature, such as federated databases and data integration systems.

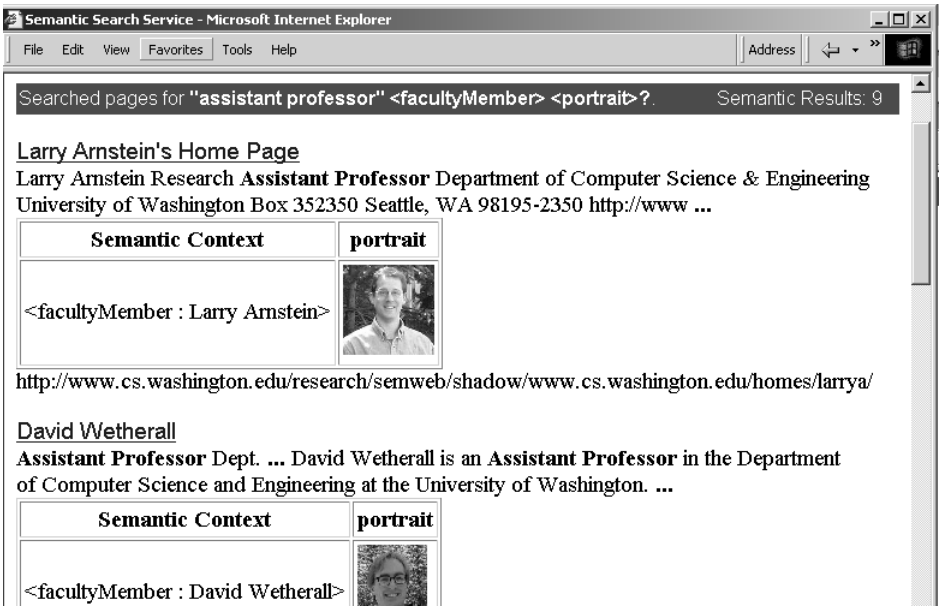

Figure 4: The semantic search service's results page. The page reproduces the original query and reports the number of results returned at the top. Matching pages contain the phrase "assistant professor" and the tags <facultyMember> and <portrait>. The ? in the query instructs the service to extract the <portrait> from each matching page.

we have deployed in our department using MANGROVE.[5] We focus, in particular, on our automatically-generated department calendar.

First, our *Who's Who* service compiles pictures, contact information, and personal information about people within an organization. In our department, a static *Who's Who* had existed for years, but was rarely updated (and was woefully out-of-date) because of the manual creation process required. Our dynamic *Who's Who* directly uses more up-to-date information from users' home pages, permitting users to update their own data at any time to reflect their changing interests.

Our experience with the *Who's Who* service illustrates an important advantage of the MANGROVE approach over asking users to enter information into databases via web forms. A large amount of useful data already exists in hand-crafted personal and organizational web pages, and the active viewing of this data over the web motivates users to keep this information up-to-date. Once these pages are tagged, MANGROVE automatically *leverages* the author's HTML updates to keep the information in its database up-to-date without additional effort on the author's part. Thus, manually maintained databases often become stale over time whereas MANGROVE's information is as fresh as HTML.

Whereas *Who's Who* merely collects information from a set of web pages, our Research Publication Database compiles a searchable database of publications produced by members of our department based on the information in home pages and project pages. This service is able to infer missing information (e.g. the author of a paper) from context (e.g., the paper was found on the author's home page) and applies simple heuristics to avoid repeated entries by detecting duplicate publications. Consistent with our goal of enabling light, incremental tagging, only a single `<publication>` tag enclosing a description of the publication is required in order to add an entry to the database. However, users may improve the quality of the output and the duplicate removal by specifying additional tags such as `<author>` and `<title>`.

Our most sophisticated service, the department calendar (shown in Figure 5), automatically constructs and updates a unified view of departmental events and displays them graphically. As with our other services, the calendar requires only a date and name to include an event in its output, but will make use of as much other information as is available (such as time, location, presenter, etc.).

Department members are motivated to tag their events' home pages in order to publicize their events. The calendar has only been in active use for a few weeks but already averages about 27 distinct visits per weekday, with an average of 1.9 page views per visit.[6] The number of visits has been rising steadily as members of the department have begun to rely on the service.

We initially seeded the calendar with date, time, and location information for courses and seminars by running a single wrapper on a university course summary page. Users then provide more detail by annotating a page about one of these events (e.g., users have tagged pre-existing HTML pages to identify the weekly topics for seminars). Alternatively, users may annotate pages to add new events to the calendar (e.g., an administrator has annotated a web page listing qualifying exams). Typically, users annotate and publish their updated pages, the calendar is immediately updated, and users then view the calendar to verify that their events have been included. For changes (e.g., when an exam is re-scheduled), users may re-publish their pages or rely on the MANGROVE web crawler to capture such updates at a later time.

It is easy to conceive of other services as well. We view the services described above as informal evidence that even light tagging can facilitate a host of useful services, which motivates further tagging, etc. Of course, more experience with deployed services and actual users in the context of multiple organizations is necessary to validate and fine-tune our architecture.

3.4 Semantic Browser

MANGROVE interleaves tags with HTML in a manner that is invisible to standard browsers such as Internet Explorer or Netscape.

[5] Accessible at http://www.cs.washington.edu/research/semweb.

[6] These statistics exclude traffic from MANGROVE project team members.

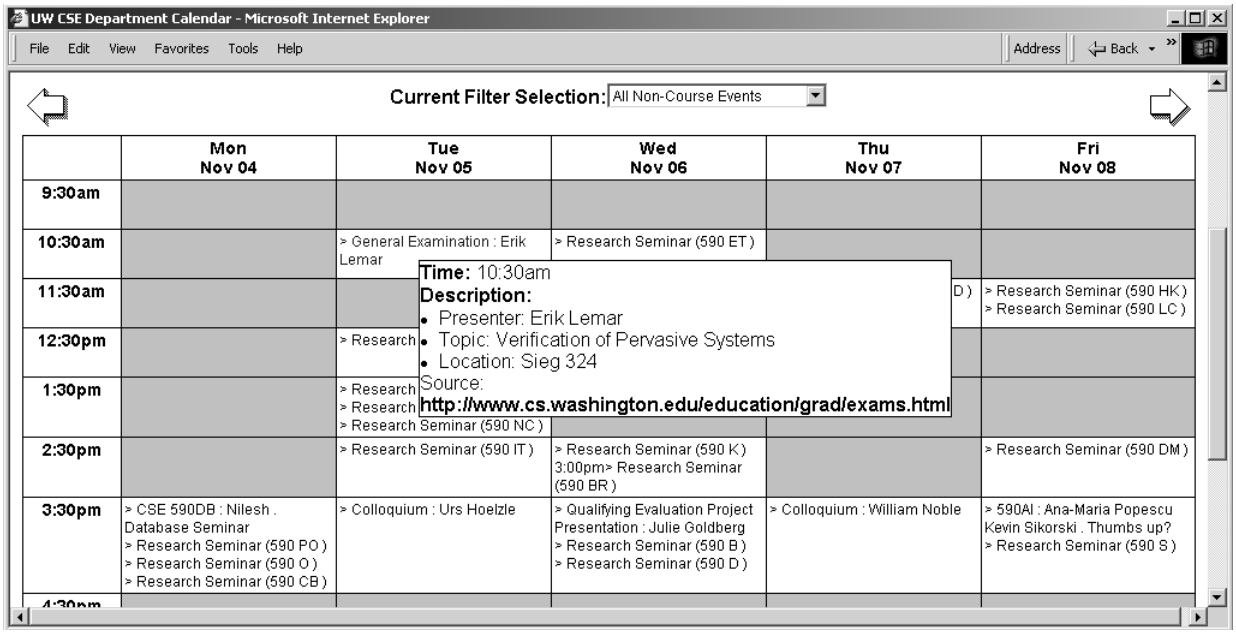

Figure 5: The calendar service as deployed in our department. See `http://www.cs.washington.edu/research/semweb` for the live version. The popup box appears when the user mouses over a particular event, and displays additional information and its origin.

Other researchers have developed browsers that either display only semantic content [39] or simply insert comments about an HTML document [29]. Instead, we developed a semantic browser that dynamically transforms its page presentation based on user preferences. The browser's transformation set includes highlighting selected items or links, adding links to a page, and omitting elements of a page. For example, items that match user preferences may be highlighted in color. In addition, MTS tags enable the browser to turn text into links. For example, when a faculty member lists students under the <advisedStudent> tag, the student's name can become a link to the student's homepage.

The omission of information can improve the browsing experience for cluttered pages, particularly on devices such as PDAs. We refer to this practice as *subwebbing*. A user can rely on one of two models of subwebbing. The browser can omit any content that is *not* explicitly tagged to be of interest; e.g., a teaching assistant accessing student pages may ask to see only content that is relevant to her course. Alternatively the browser can omit content if it *is* explicitly tagged as information that is *not* of interest; e.g., a prospective student may wish the browser to omit information that is tagged as being only of local interest.

We utilize a plug-in to Internet Explorer that enables a user to indicate his preferences through toolbar selections. For example, the user may only be interested in research content, or in content relevant to undergraduates. That information is stored in a cookie that is transmitted to a proxy server with every page request. The server queries the MANGROVE database to decide what portion of the page to transmit to the user.[7] The server relies on caching to improve efficiency for computationally expensive queries. Our semantic browser is only partly implemented at this point.

4. PRACTICAL DATA MAINTENANCE

Database and knowledge base systems have a set of mechanisms that ensure that the contents of a database are clean and correct. For example, database systems enforce integrity constraints on data entry, and therefore eliminate many opportunities for entering "dirty" data. In addition, database applications control carefully who is allowed to enter data, and therefore malicious data entry is rarely an issue. One of the novel aspects of a system like MANGROVE is that we can no longer rely on such mechanisms because we do not have a central administration of our database. Below we describe how MANGROVE handles such issues in large-scale data sharing.

4.1 Malformed Data

On the HTML web, a user can put his phone number on a web page without considering whether it already appears anywhere else (e.g., in an employer's directory), or how others have formatted or structured that information. Despite that, users can effectively assess the correctness of the information they find (e.g., by inspecting the URL of the page) and interpret the data according to domain-specific conventions. In contrast, existing systems often restrict the way information may be expressed. For instance, in WebKB-2 [34], a user may not add information that contradicts another user unless the contradictions are explicitly identified first. Likewise, in SHOE [24], all data must conform to a specified type (for instance, dates must conform to RFC 1123).

MANGROVE purposefully does not enforce any integrity constraints on tagged data or restrict what claims a user can make. With the calendar, for instance, annotated events may be missing a name (or have more than one), dates may be ambiguous, and some data may even be intentionally misleading. Instead, MANGROVE *defers* all such integrity constraints to allow users to say anything they want, in any format. Furthermore, MTS allows users to decide how extensively to tag their data. For instance, the <instructor> tag may refer to a resource with further properties such as <name> and<workPhone> (e.g., of John Fitz in Figure 2), or simply to a string literal (e.g., "<instructor>John Fitz</instructor>"). These features simplify the tagging and gradual evolution of existing HTML.

The primary burden of *cleaning* the data is passed to the service consuming the data, based on the observation that different

[7] Architecturally, the browser is simply another service that consumes tags.

services will have varying requirements for data integrity. In some services, clean data may not be as important because users can tell easily whether the answers they are receiving are correct or not (possibly by following a hyperlink). For other services, it may be important that data be consistent (e.g., that an event have the correct location), and there may be some obvious heuristics on how to resolve conflicts. The source URL of the data is stored in the database and can serve as an important resource for cleaning up the data. To assist with this process, MANGROVE's basic service template enables services to apply a simple rule-based *cleaning policy* to results from the database. For example, for calendar course and seminar events, our calendar specifies a simple policy that prefers information from the department's domain over information from elsewhere on the web. Thus, factual conflicts are resolved in the department's favor. This policy also helps the calendar to deal with different degrees of tagging. For instance, to identify the instructor for a course lecture, the calendar simply requests the value of the <instructor> tag, and the template library automatically returns the <name> sub-property of the instructor if it exists, or the complete value of that tag if sub-properties are not specified. Services may create their own cleaning policy or use a default provided by the system.

Even when data is consistent and reliable, services still face the problem of *interpreting* the semantic data. For instance, dates found on existing web pages are expressed in natural language and vary widely in format. We note that while this problem of data interpretation is difficult in general, once users have explicitly identified different semantic components (e.g., with a <date> tag), simple heuristics are sufficient to enable useful services for many cases. For instance, MANGROVE's service template provides a simple date and time parser that we have found very effective for the calendar service. In addition, semantic context can assist the cleaning process, e.g., to provide a missing year for an event specified as part of a course description.

4.2 Overcoming Malicious Misinformation

The highly distributed nature of the web can lead to abuse, which popular services such as search engines have to grapple with on a regular basis. Potential abuse is an issue for semantic services as well. What is to prevent a user from maliciously publishing misleading information? Imagine, for example, that a nefarious AI professor purposefully publishes a misleading location for the highly popular database seminar in an attempt to "hijack" students and send them to the location of the AI seminar.

We have considered several approaches to combating this kind of "semantic spoofing." We could have an administrator verify information before it is published, creating a "moderated" semantic web. However, this non-automated approach prevents instant gratification and does not scale. Alternatively, we could enable automated publishing for password-authenticated users, but investigate complaints of abuse and respond by disabling an abuser's publishing privileges. This approach, however, is more complicated for users and prevents the same data from being easily shared by more than one semantic domain. Instead, we chose a fully automated system that mirrors the solution adopted by search engines — associating a URL with every search result and leaving decisions about trust to the user's discretion.

Thus, an important design principle for MANGROVE services is to associate an easily-accessible source (i.e., a URL) with each fact made visible to the user. For example, as shown in Figure 5, a user can "mouse over" any event in the calendar and see additional facts including one or more originating URLs. The user can click on these URLs to visit these pages and see the original context for the facts. Naturally, service writers are free to implement more sophisticated policies based on freshness, URL, or further authentication. For instance, in case of conflict, our department calendar uses its previously mentioned cleaning policy to enable facts published from pages whose URL starts with `http://www.cs.washington.edu/education/` to override facts originating elsewhere.

4.3 Service Feedback

Another fundamental difference between MANGROVE and traditional data management applications is that authors who enter data may not be aware of which services consume their data and what is required in order for their data to be well formed. Hence, an author may annotate data and publish it, but then fail to find the desired event in the calendar (perhaps because the calendar did not understand the dates specified). The challenge is to create an environment where a novice user can easily understand and rectify this situation.

To address this problem, we provide a *service feedback* mechanism that accepts a URL as input. Services registered with the notifier then process all tagged data at that URL and output information about problems encountered (e.g., a date was ambiguous or missing) and/or about successful processing (e.g., a link to the newly added calendar event).[8] As a convenience, we invoke the service feedback mechanism whenever authors publish new data. Thus, this mechanism supports both robustness (by helping authors to produce well-formed data) and instant gratification (by making it easier for authors to find the tangible output resulting from their new semantic data). Furthermore, the mechanism enables authors to receive serendipitous feedback from services previously unknown to the author, thus creating a discovery mechanism for potentially relevant semantic services.

4.4 Discussion

MANGROVE is designed to support the goals listed in the beginning of Section 2. First, MANGROVE supports *ease of authoring* with the convenient MTS syntax, the graphical tagger, and the ability to seamlessly inline tags inside pre-existing HTML documents. In addition, deferring integrity constraints to the application, and the ability to tag HTML lightly and incrementally also help to ease the burden of semantic tagging. Next, MANGROVE supports *instant gratification* with a loop that takes freshly published semantic content through the parser, to the database, through the notifier, to MANGROVE services, and then back to the user through the service feedback mechanism described above. Finally, MANGROVE supports *robustness* with its service feedback mechanism, by associating a URL with every fact in the database, and by providing the service construction template, which assists services in cleaning and interpreting the data based on these URLs.

5. EXPERIENCE WITH MANGROVE

This section presents some of our initial experience using and evaluating our MANGROVE prototype. It is not meant to be conclusive: the system and its services are new and still evolving. Therefore, our goals here are modest, namely, to address some basic questions about the MANGROVE approach:

1. Feasibility: Can MANGROVE be used to successfully tag and extract the factual information found in existing HTML pages?

[8] We restrict the processing to a limited number of services for tractability, and because not all services will have this feedback capability.

Search Objective	Google f (Prec.,Rec.)	Tag-only Search f (Prec.,Rec.)	Tag+Text Search f (Prec., Rec.)
Assistant Professor photos	0.75 (**100%**,60%)	0.82 (**100%**,70%)	**0.84** (89%, **80%**)
Associate Professor photos	0.52 (75%,40%)	0.89 (**100%**,80%)	**0.91** (83%,**100%**)
Assistant Professor phone numbers	0.64 (58%,70%)	0.89 (**100%**,80%)	**0.95** (91%,**100%**)
Associate Professor phone numbers	0.29 (19%,60%)	0.67 (75%,60%)	**0.80** (**80%**, **80%**)

Table 1: Comparison of Search Services. In each box, the first value is the *f-score* of the query, followed by the *precision* and *recall* in parentheses. Within each row, the values in bold represent the maximum value for that metric.

2. Benefit: Can MANGROVE services actually benefit users when compared with popular commercial services? Specifically, we attempt to quantify the potential benefit of MANGROVE's semantic search service as compared with Google.

These questions are mostly qualitative, however we develop some simple measures in an attempt to quantify the feasibility and the potential benefits of our approach.

5.1 Information Capture

To test the extent to which (1) our system can successfully extract a range of information from existing HTML pages, and (2) our existing web actually contains the information of interest, we created a copy of our department's web space for experimentation. The department web consists of about 68,000 pages whose HTML content is about 480 MB. We then tagged the home pages of all 44 faculty members using the graphical tagger, focusing particularly on adding 10 common tags such as <name>, <portrait>, and <advisedStudent>. Four graduate students were instructed to tag and publish each document, but not to make *any* other changes to the original HTML. The students were familiar with MANGROVE, though some had previously tagged only their own home page.

We evaluate tagging success by examining the output of our *Who's Who* service and comparing it with the original HTML documents. Of the 440 possible data items (e.g., a faculty member's name or picture), 96 were not present in the original HTML. For instance, only half of the professors had their office location on their home page. Of the remaining 344 facts, the vast majority (318, or 92.4%) were correctly displayed by *Who's Who*, while 26 had some sort of problem. Nine of these problems were due to simple oversight (i.e., the data was present but simply not tagged), while 8 items had tagging errors (e.g., using an incorrect tag name). For 6 data items, it was not possible to succinctly tag the data with MTS. For instance, MTS presently cannot tag a single string as both a home and office phone number. Finally, three tagged items revealed minor bugs with the *Who's Who* service itself.

Thus, despite the variety of HTML pages (we have no standard format for personal home pages) and the presence of some inevitable annotation errors, we successfully extracted a large amount of relevant information and constructed a useful service with the data. This simple measurement suggests that while additional MANGROVE features may improve the tagging process, the overall annotation and extraction approach is feasible in practice.

5.2 Benefits of MANGROVE Search

Using the tagged data discussed above, we examined the effectiveness of MANGROVE's search service (described in Section 3.2). As a simple search exercise, we issued a small set of queries to retrieve the picture and phone number of all assistant and associate professors in our department. For comparison, we sent the same search queries to Google and to MANGROVE's tag-only search, which accepts sets of tags as queries. For example, we issued the query:

<facultyMember> <jobTitle="assistant professor"> <portrait>?

Obviously, Google has different goals than our search service, so the results are not necessarily comparable, however the exercise helps to illustrate the potential benefit from even a modest amount of tagging.

When sending queries to Google, we included `site:cs.washington.edu` to restrict the query to our site. We tried several variants of each query (e.g., `"assistant professor"`, `"assistant professor" phone`, `"assistant professor" phone OR voice`, etc.). For finding photos, we also queried the Google image search directly. In each case, we inspected all results returned by Google for the desired photo or phone number.

In addition, we wanted to assess how robust MANGROVE is to tagging omissions or errors. What happens, for example, when the <jobTitle> is omitted or applied incorrectly? Users can fall back on MANGROVE's tag+text search, which filters Google results using tag information as explained in Section 3.2. In our tag+text queries, we combined a phrase (e.g., `"assistant professor"`) with the tag <facultyMember> and the tag to be extracted (<workPhone> or <portrait>). Figure 4 shows the results of one such query.

Table 1 summarizes the results for our three experimental conditions: Google, tag-only search, and tag+text search. We use the standard information retrieval metrics of precision (p) and recall (r), and combine them into an f-score (f) as follows:

$$f = 2(\beta + 1)pr/2(\beta p + r)$$

The f-score is a standard method of combining recall and precision to facilitate comparison [48].[9] As is commonly done, we set the parameter β to 1 in order to weight precision and recall equally. In this experiment, precision is the percentage of the results, for each engine, that were correct; recall is the percentage of the total correct answers returned by each engine.

The table supports several tentative observations. First, we see that tags can substantially improve precision over Google's results. Second, and more surprising, tags often result in improved recall over Google as well. The reason is that querying Google with a query such as `"assistant professor"`, restricted to our site, returns 176 results with very low precision. A query that yields much better precision and a much higher f-score for Google is `"assistant professor" phone OR voice`; this longer query yields lower recall than the tag-based searches, though, because it only returns pages that contain the word 'phone' or the word 'voice'. Since the tag-based searches yield both better precision and better recall, it is not surprising that their f-score is substantially better than Google's. Of course, far more extensive ex-

[9] To be fair to Google, we tried multiple formulations of each query as mentioned above. The results reported for Google in each row are the ones whose f-score was maximal.

periments are necessary to see whether the above observations are broadly applicable.

The table also enables us to compare the two variants of tag-based search. We see that tag-only search tends to have very high precision, but lower recall when compared to tag+text search. Tag+text has higher recall because it is more robust to omitted or incorrect tags. For example, in some cases a professor's rank was not tagged properly due to human error. Tag+text search also offers the ability to search based on data that was not tagged because a suitable tag did not exist, as would be the case if we had omitted the <jobTitle> from the schema.

Both of our tag-based searches have the further benefit that they can extract the tagged information from web pages (see Figure 4), whereas Google only sometimes manages to extract this information with its image search or in its result snippets. This feature of our service makes it much simpler to quickly view the results and facilitates the use of search queries as building blocks for more sophisticated services.

5.3 Discussion

All of our measurements are preliminary and the results, while thought provoking, are far from definitive. Nevertheless, the measurements do provide evidence for the feasibility of MANGROVE and its potential for supporting value-added services for users. One might argue that the comparison of MANGROVE's search with Google is not fair because the semantic search makes use of additional information in the form of tags that have to be inserted into HTML pages manually. However, our goal is not to argue that MANGROVE has a better search algorithm, but rather to quantify the benefit that can result from this tagging effort.

6. RELATED WORK

Much of the related work on the semantic web has focused on designing single components or analyzing particular semantic issues whereas in MANGROVE we have designed, implemented, and deployed one of only a handful of complete semantic web systems. As a result, we have been able to report on measurements validating aspects of our design. Furthermore, MANGROVE services are used daily in our department providing the pragmatic lessons discussed earlier.

This paper is the first to articulate and focus on *instant gratification* as a central design goal for a semantic web system. Many of the key differences between MANGROVE's architecture and that of related semantic web systems follow from this distinct design goal. We discuss these differences in more detail below.

The systems most closely related to our work are OntoBroker [8] and SHOE [24], both of which make use of annotations inside HTML documents. Although SHOE's language did permit users to inject annotations into HTML pages, their annotations do not actually use the existing HTML content. Thus, both with SHOE and with OntoBroker's RDF-based annotations, all factual HTML data must be repeated in the semantic annotation, leading to the redundancy and maintenance problems discussed earlier.[10]

SHOE's services, like those of many other systems, primarily consisted of tools to simply search or view semantic data, although their "Path Analyzer" [25] provided a convenient interface for exploring relationships among concepts. OntoBroker did implement a number of services, such as a Community Web Portal [50] and services intended to assist business processes [42]. However, with both SHOE and OntoBroker, it is not clear whether these services were used in practice, nor have we found any measurements relating to them. In addition, MANGROVE has the advantage of enabling useful services even when content is only lightly tagged. For instance, while OntoBroker's "SoccerSearch" service [42] tries a semantic search and then a textual search if the former fails, MANGROVE's tag+text search service can profitably combine both types of information.

SHOE and OntoBroker primarily rely upon periodic web crawls to obtain new information from annotated HTML, thus preventing instant gratification and content creation feedback. Alternatively, some systems provide a web interface for users to directly enter semantic knowledge [34, 8] or to instruct the system to immediately process the content of some URL [34]. However, we are aware of no existing systems that support this feature in a manner that provides instant gratification for typical web authors. For instance, the WebKB-2 system supports a command to load a URL, but this command must be embedded within a script, and existing data must be manually deleted from the repository before a (modified) document can be reprocessed.

WebKB-1 and WebKB-2 [33, 34] also provide a way to embed semantic knowledge in HTML documents, this time using expressive conceptual graphs and an extensive ontology, but generally require the duplication of information in those documents. In addition, their services are currently limited to either information browsing or a semantic-only search. The OntoSeek project [18] addresses goals of information retrieval similar to WebKB but does not support the encoding of information in HTML pages and does not provide any services beyond search.

Many semantic systems strive to provide a rich set of inferencing capabilities, some of which might be a useful addition to MANGROVE. However, these capabilities can also cause a performance bottleneck, without necessarily improving results (e.g., the experience of OntoBroker in [14]). MANGROVE instead focuses on providing simple, scalable mechanisms that can provide useful services in practice.

Other researchers have constructed semantic web systems that are restricted to a single application domain, for instance ITTALKS [43] and SemTalk [16]. These systems provide useful services but their overall architecture, including storage mechanisms, is domain specific. Recently, QuizRDF [6] introduced a search service that combines textual and semantic content. QuizRDF's service has an elegant user interface, but is not yet deployed on the web for general use, nor have they reported on any measurements of its precision and recall.

Since MANGROVE represents data internally as RDF, we can leverage existing work for processing and querying RDF, and export RDF content to other systems. In particular, MANGROVE utilizes Jena [36], a toolkit we have found very useful for storing and accessing semantic information. Other systems that also offer centralized RDF storage include Kaon [37] and Sesame [5]. Edutella [41] extends these approaches to provide RDF annotation, storage, and querying in a distributed peer-to-peer environment, and proposes some services [40], but primarily assumes the pre-existence of RDF data sources rather than considering the necessary architectures and services to motivate semantic web adoption. We view these systems as valuable modules for complete semantic web systems such as MANGROVE. In contrast, MANGROVE supports the complete cycle of content creation, real-time content aggregation, and execution of services that provide instant gratification to content authors.

Other systems that have permitted the annotation of HTML doc-

[10] Early work with OntoBroker's HTML-A annotation language and OntoPad [50] originally permitted annotations to reuse existing data; however, later work with CREAM [22] lost this very useful feature as the group focused on an RDF encoding.

uments include the "lightweight databases" of [11] and the annotation tool of [51], but both systems are merely modules for complete semantic web systems. CREAM [22] provides a sophisticated graphical tool that allows annotation of HTML documents similar to our graphical tagger, but it must replicate data due to its use of RDF. Some systems (i.e., Annotea [29]) avoid redundancy by using XPointers to attempt to track which part of a document an annotation refers to, but this approach may fail after document revisions and only applies to XHTML documents, which makes it incompatible with the majority of information on the web. Haustein and Pleumann [23] instead store all semantically relevant information in a database and dynamically generate both RDF and human-readable versions from the database, but we believe this is too unwieldy for average users in today's web environment.

Ideally, one would not need to annotate HTML at all, and would rely on automated techniques to handle this. In a sense, this is the motivation for information extraction and wrapper induction techniques [30, 47, 32, 3]. These techniques — especially those that support automatic learning of what data to extract (e.g., [30]) — would be a very useful complement to MANGROVE's graphical tagger.

MTS syntax is not based on XHTML because most existing pages are in HTML and hence would require reformatting to become valid XHTML, and many users are averse to any such enforced reformatting. Furthermore, a large fraction of HTML documents contain HTML syntax errors (generally masked by browsers) and thus would require manual intervention to reliably convert to legal XHTML. However, existing XHTML (or XML) documents that are annotated with MTS will still be valid XHTML (XML) documents, and thus tools that produce or manipulate these formats may still be freely used with MTS-annotated documents.

The W3C and many others have advocated the use of digitally signed RDF to ensure the reliability of RDF data [53, 54, 27]. This approach may be logical in cases where data integrity is essential, but is too heavyweight for average users, and is not necessary for most services (where usability and freshness are more important). Furthermore, signed RDF only solves half of the data authentication problem — the part MANGROVE solves by simply maintaining the source URL for all content. The more difficult problem is, given the known source of all data, how should services decide what data sources and data are reliable, and how should this information be presented to the user? This paper highlights how some simple policies can work well for common services and argues for revealing the source of the data to the end user, similar to the way users ascertain the validity of web content today.

Finally, our semantic browser is inspired by work on adaptive web sites originally proposed by [44]. Adaptive web sites automatically modify and customize their layout and organization based on their analysis of traffic patterns [45, 2]. Several papers have noted that such efforts would benefit from the sort of information available in the semantic web [45, 15]. Our browser's transformations are also similar to those achievable with XSL stylesheets [52] for XML documents, but our technique provides an easy way for browser users, not just content providers, to control the document presentation.

7. CONCLUSION

This paper presented MANGROVE as a means of demonstrating how a component of the semantic web can evolve over time from today's HTML-based web. Specifically, the paper reports on the following contributions:

1. We introduced the MANGROVE architecture that supports the complete semantic web "life-cycle" from content authoring to semantic web services. MANGROVE's key design goals are *ease of authoring*, support for *instant gratification* (*i.e.*, immediate, tangible value resulting from semantic tagging), and *robustness* to malformed data and malicious misinformation. We showed how elements of the architecture support each design goal.

2. We sketched a catalog of deployed semantic services that motivate the tagging of HTML content by consuming tagged information. We showed how to update tagged information over time, and how to overcome "semantic spoofing" (the attempt to introduce erroneous information into the system). We reported on preliminary measurements that lend credence to the claim that our services are both feasible and beneficial.

3. We analyzed the impact of semantic markup in MANGROVE on web authoring, browsing, and searching. We demonstrated how our semantic tags can be interleaved with existing HTML in a non-redundant manner that is invisible to today's browsers. We showed how the markup can improve search precision and recall over Google. We also explained how MANGROVE can improve the browsing experience particularly for devices with limited screen size such as PDAs.

Our goal in designing MANGROVE and in deploying it locally has been to test our design on today's HTML web against the requirements of ordinary users. Clearly, additional deployments in different universities, organizations, and countries are necessary to further refine and validate MANGROVE's design. New instant gratification services are necessary to drive further adoption, and a broad set of measurements is essential to test the usability and scalability of the system. Finally, we plan to incorporate MANGROVE as part of a peer-data management system to achieve web scale.

8. REFERENCES

[1] S. Adali, K. Candan, Y. Papakonstantinou, and V. Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proc. of SIGMOD*, pages 137–148, Montreal, Canada, 1996.

[2] C. R. Anderson, P. Domingos, and D. S. Weld. Personalizing web sites for mobile users. In *World Wide Web*, pages 565–575, 2001.

[3] R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with Lixto. In *VLDB 01*, 2001.

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[5] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: An architecture for storing and querying RDF data and schema information, 2001.

[6] J. Davies, R. Weeks, and U. Krohn. QuizRDF: Search technology for the semantic web. In *Workshop on Real World RDF and Semantic Web Applications, WWW*, 2002.

[7] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein. OWL web ontology language 1.0 reference, 2002. Manuscript available from `http://www.w3.org/2001/sw/WebOnt/`.

[8] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In *Eighth Working Conference on Database Semantics (DS-8)*, pages 351–369, 1999.

[9] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: a machine learning approach. In *Proc. of SIGMOD*, 2001.

[10] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proc. of the Int. WWW Conf.*, 2002.

[11] S. A. Dobson and V. A. Burrill. Lightweight databases. *Computer Networks and ISDN Systems*, 27(6):1009–1015, 1995.

[12] R. B. Doorenbos, O. Etzioni, and D. S. Weld. A scalable comparison-shopping agent for the world-wide web. In W. L. Johnson and B. Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 39–48, Marina del Rey, CA, USA, 1997. ACM Press.

[13] O. Etzioni, S. Gribble, A. Halevy, H. Levy, and L. McDowell. An evolutionary approach to the semantic web. In *Poster presentation at the First International Semantic Web Conference*, 2002.

[14] D. Fensel, J. Angele, S. Decker, M. Erdmann, H.-P. Schnurr, R. Studer, and A. Witt. On2broker: Lessons learned from applying AI to the web. Technical Report Institute AIFB Research report no. 383, 1998.

[15] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. Declarative specification of web sites with Strudel. *VLDB Journal*, 9(1):38–55, 2000.

[16] C. Fillies, G. Wood-Albrecht, and F. Weichardt. A pragmatic application of the semantic web using SemTalk. In *WWW*, pages 686–692, 2002.

[17] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. *Journal of Intelligent Information Systems*, 8(2):117–132, March 1997.

[18] N. Guarino, C. Masolo, and G. Vetere. Ontoseek: Content-based access to the web, 1999.

[19] L. Haas, D. Kossmann, E. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proc. of VLDB*, Athens, Greece, 1997.

[20] A. Halevy, O. Etzioni, A. Doan, Z. Ives, J. Madhavan, L. McDowell, and I. Tatarinov. Crossing the structure chasm. In *First Biennial Conferenece on Innovative Data Systems Research, Asilomar, CA, January 5-8, 2003*.

[21] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proc. of ICDE*, 2003.

[22] S. Handschuh and S. Staab. Authoring and annotation of web pages in CREAM. In *World Wide Web*, pages 462–473, 2002.

[23] S. Haustein and J. Pleumann. Is participation in the semantic web too difficult? In *First International Semantic Web Conference, Sardinia, Italy, June 2002*.

[24] J. Heflin, J. Hendler, and S. Luke. SHOE: A knowledge representation language for internet applications. Technical Report CS-TR-4078, 1999.

[25] J. Heflin, J. A. Hendler, and S. Luke. Applying ontology to the web: A case study. In *IWANN (2)*, pages 715–724, 1999.

[26] I. Horrocks, F. van Harmelen, and P. Patel-Schneider. DAML+OIL. http://www.daml.org/2001/03/daml+oil-index.html, March 2001.

[27] D. Huynh, D. Karger, and D. Quan. Haystack: A platform for creating, organizing and visualizing information using RDF.

[28] Z. Ives, D. Florescu, M. Friedman, A. Levy, and D. Weld. An adaptive query execution engine for data integration. In *Proc. of SIGMOD*, pages 299–310, 1999.

[29] J. Kahan and M.-R. Koivunen. Annotea: an open RDF infrastructure for shared web annotations. In *World Wide Web*, pages 623–632, 2001.

[30] N. Kushmerick, R. Doorenbos, and D. Weld. Wrapper induction for information extraction. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence(IJCAI)*, 1997.

[31] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB*, pages 251–262, Bombay, India, 1996.

[32] L. Liu, C. Pu, and W. Han. XWRAP: An XML-enabled wrapper construction system for web information sources. In *ICDE '00*, pages 611–621, 2000.

[33] P. Martin and P. W. Eklund. Embedding knowledge in web documents. *WWW8 / Computer Networks*, 31(11-16):1403–1419, 1999.

[34] P. Martin and P. W. Eklund. Large-scale cooperatively-built KBs. In *ICCS*, pages 231–244, 2001.

[35] B. McBride. Four steps towards the widespread adoption of a semantic web. In *First International Semantic Web Conference, Sardinia, Italy, June 2002*.

[36] B. McBride. Jena: Implementing the RDF model and syntax specification. http://www-uk.hpl.hp.com/people/bwm/papers/20001221-paper/, 2001. Hewlett Packard Laboratories.

[37] B. Motik, A. Maedche, and R. Volz. A conceptual modeling approach for building semantics-driven enterprise applications. In *Proceedings of the First International Conference on Ontologies, Dataases and Application of Semantics (ODBASE-2002)*, 2002.

[38] I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In O. Etzioni, J. P. Müller, and J. M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 190–197, Seattle, WA, USA, 1999. ACM Press.

[39] A. Naeve. The concept browser - a new form of knowledge management tool. In *2nd European Web-based Learning Environments Conference (WBLE 2001), Lund, Sweden*.

[40] W. Nejdl. Semantic web and peer-to-peer technologies for distributed learning repositories. In *17th IFIP World Computer Congress, Intelligent Information Processing /IIP-2002*.

[41] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmr, and T. Risch. Edutella: a P2P networking infrastructure based on RDF. In *WWW*, pages 604–615, 2002.

[42] Ontoprise. Demo applications. http://www.ontoprise.de/com/co_produ_appl2.htm.

[43] F. Perich, L. Kagal, H. Chen, S. Tolia, Y. Zou, T. W. Finin, A. Joshi, Y. Peng, R. S. Cost, and C. Nicholas. ITTALKS: An application of agents in the semantic web. In *ESAW*, pages 175–194, 2001.

[44] M. Perkowitz and O. Etzioni. Adaptive web sites: an AI challenge. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 1997.

[45] M. Perkowitz and O. Etzioni. Adaptive web sites: Conceptual framework and case study. In *Proceedings of the Eighth Int. WWW Conference*, 1999.

[46] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[47] A. Sahuguet and F. Azavant. Building light-weight wrappers for legacy web data-sources using W4F. In *VLDB '99*, pages 738–741, 1999.

[48] W. M. Shaw Jr., R. Burgin, and P. Howell. Performance standards and evaluations in IR test collections: Cluster-based retrieval models. *Information Processing and Management*, 33(1):1–14, 1997.

[49] S. Soderland. Learning to extract text-based information from the World Wide Web. In *Knowledge Discovery and Data Mining*, pages 251–254, 1997.

[50] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H.-P. Schnurr, R. Studer, and Y. Sure. Semantic community web portals. *WWW9 / Computer Networks*, 33(1-6):473–491, 2000.

[51] M. Vargas-Vera, E. Motta, J. Domingue, S. B. Shum, and M. Lanzoni. Knowledge extraction by using an ontology-based annotation tool. In *K-CAP 2001 workshop on Knowledge Markup and Semantic Annotation, Victoria*, 2001.

[52] W3C. The extensible stylesheet language (XSL). http://www.w3.org/Style/XSL/.

[53] W3C. W3C issues recommendation for resource description framework (RDF). W3C press release 24 February 1999. http://www.w3.org/Press/1999/RDF-REC.

[54] W3C. XML digital signatures activity statement. http://www.w3.org/Signature/Activity.html, 2002.