Mangrove: Enticing Ordinary People onto the Semantic Web via Instant Gratification

Luke McDowell, Oren Etzioni, Steven D. Gribble, Alon Halevy, Henry Levy, William Pentney, Deepak Verma, and Stani Vlasseva

University of Washington, Department of Computer Science and Engineering, Seattle, WA 98195 USA,
{lucasm,etzioni,gribble,alon,levy,bill,deepak,stani}@cs.washington.edu
http://www.cs.washington.edu/research/semweb

Abstract. Despite numerous efforts, 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 entice non-technical people to semantically annotate their existing HTML data. MANGROVE seeks to alter the cost-benefit equation of authoring semantic content. To increase the benefit, MANGROVE is designed to make semantic content instantly available to services that consume the content and yield immediate, tangible benefit to authors. To reduce the cost, MANGROVE makes semantic authoring as painless as possible by transferring some of the burden of schema design, data cleaning, and data structuring from content authors to the programmers who create semantic services.

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.

1 Introduction and Motivation

Numerous proposals for creating a *semantic web* have been made in recent years (e.g., [3,6,17]), 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 [9,24,16]. 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

D. Fensel et al. (Eds.): ISWC 2003, LNCS 2870, pp. 754–770, 2003.

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 utilizing such technical people will certainly yield many useful semantic web services, this paper is concerned with the non-technical people who drove the explosive growth of the HTML-based web. We consider the question *how do we entice non-technical people to structure their data?* This paper presents the architecture of MANGROVE, a system designed to enable and entice these "ordinary users" to contribute to the semantic web. In particular, MANGROVE seeks to emulate three key conditions that contributed to the rapid growth of the web:

- **Instant Gratification:** In the HTML world, a newly authored page is immediately accessible through a browser; we mimic this feature in MANGROVE by making annotated content instantly available to services. We posit that semantic annotation will be motivated by services that consume the annotations and result in immediate, tangible benefit to authors. MANGROVE provides several such services and the infrastructure to create additional ones over time.
- **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.
- **Ease of Authoring:** MANGROVE provides a graphical web-page annotation tool that enables users to easily and incrementally annotate existing HTML content.

As one example of the MANGROVE approach, consider the web site of our computer science department. The web pages at this site contain numerous facts including contact information, locations, schedules, publications, and relationships to other information. If users were enabled and motivated to semantically annotate these pages, then the pages and annotations could be used to support both standard HTML-based browsing as well as novel semantic services. For example, we have created a departmental calendar that draws on annotated information found on existing web pages, which describe courses, seminars, and other events. The calendar instantly consumes annotated facts as explained below. Because the calendar is authoritative and prominently placed in the department's web, events that appear in it are more likely to receive the attention of the department's community. As a result, people seeking to advertise events (e.g., seminars) are motivated to annotate their pages, which leads to their automatic inclusion in the department's calendar and in the semantic web.

The remainder of this paper is organized as follows. The next section introduces MANGROVE's architecture and explains how it supports its design goals. Section 3 describes our first semantic services and our initial experience from deploying MANGROVE. Section 4 discusses related work on this problem, and Section 5 concludes.

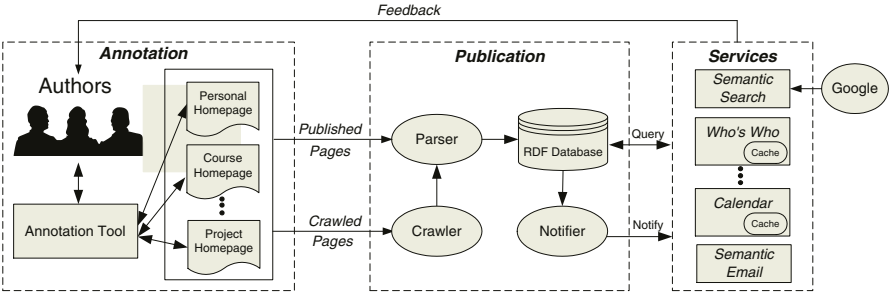

Fig. 1. The MANGROVE architecture and sample services.

2 The Architecture of MANGROVE

This section presents the high-level architecture of MANGROVE, details some of the key components, and relates them to MANGROVE's design goals.

2.1 Architecture Overview

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

- **Annotation:** Authors use our *graphical annotation tool* or an editor to insert annotations into existing HTML documents. The annotation tool provides users with a list of possible properties from a local schema based on the annotation context (e.g., describing a person or course), and stores the semantic data using a syntax that is simply syntactic sugar for basic RDF.
- **Publication:** Authors can explicitly *publish* annotated 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. For example, we support semantic search, semantic email, and more complex services such as the automatically-generated department calendar.

These three phases are overlapping and iterative. For instance, after annotation, publication, and service execution, an author may refine her documents to add additional annotations 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 MANGROVE in more detail. We focus first on architectural features that support instant gratification and robustness. Section 3 then

describes the semantic services that make use of MANGROVE to provide instant gratification to content authors. We omit many aspects of components that use standard technology, such as our crawler, parser, and annotation tool. See [26] for more details.

2.2 Supporting Instant Gratification

In today's web, changes to a web page are immediately visible through a browser. We create the analogous experience in MANGROVE by enabling authors to *publish* semantically annotated 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 MANGROVE's graphical annotation tool, or they can enter the URL of an annotated 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).

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., [6,17]) 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.

Notification: Services specify data of interest by providing a query to the MANGROVE notifier.[1] 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`> properties (or that had such properties deleted). When it receives notification of relevant new data, the calendar processes that data and updates its internal data structures, ensuring that content authors see their new data on the calendar with minimal delay.[2]

[1] For simplicity, we assume in this paper that such a query consists of just a set of "relevant" RDF properties. More complex queries can be also supported efficiently [30].

[2] Note that while only registered services receive such notifications, any service that follows a simple API (of Mangrove or Jena[25]) may query the database for content.

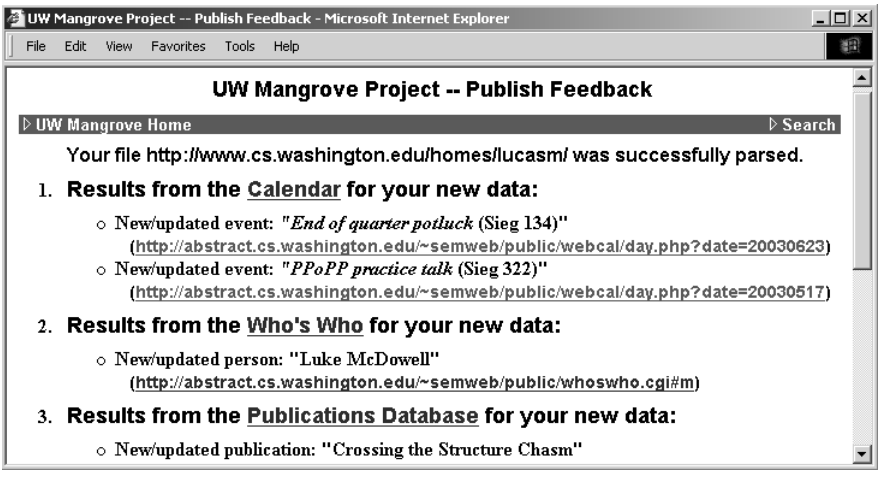

Fig. 2. Example output from the service feedback mechanism. Services that have registered interest in a property that is present at a published URL are sent relevant data from that URL. The services immediately return links to their resulting output.

Service feedback: MANGROVE provides a *service feedback* mechanism that is a key element of its architectural support for instant gratification. As noted earlier, services can register their interest in arbitrary RDF properties (e.g., `event`). Then, when a URL that contains such a property is published by an author, the services are automatically notified about the new information. Each notified service can return feedback to the author as shown in Figure 2. The feedback can identify problems encountered (e.g., a date was ambiguous or missing) or can confirm that the information was successfully "consumed" by the service.

The feedback mechanism supports instant gratification by making it easier for authors to immediately see the tangible output resulting from their new semantic data. Authors can click on any of the links shown in Figure 2 and they will be directed to a web page that shows how the information they *just* annotated is being used by a semantic service. For example, as soon as an event page is annotated and published, the organizer can click on a link and see her event appearing in the department's calendar. To be true to the 'instant' in 'instant gratification', publishing a page returns feedback to authors in about two seconds. We are working on further reducing that delay to a fraction of a second.

Because services and information are created independently in MANGROVE by different sets of people, there is the potential in the future for authors to be unaware of services that consume their information and that would provide further motivation for them to author more semantic information. The service feedback mechanism acts as a service discovery mechanism that addresses this problem. Once a service registers its interest in a particular property, an author that publishes relevant information will be notified about that service's interest

in the property.[3] We expect that users will typically publish content with a particular service in mind, and then decide whether or not to investigate and possibly annotate additional content for the services that they learn of from this feedback. As the number of services grows, an author can avoid "feedback spam" by explicitly selecting the services that send her feedback, by limiting their number, or by filtering them according to the criteria of her choice (e.g., by domain or category). Additional techniques for supporting useful feedback across a very large numbers of services, content providers, and distinct ontologies is an interesting area for future work. Note that since the author is publishing information with the hope of making it broadly available, privacy does not seem to be a concern in this context.

The service feedback mechanism also supports robustness by helping authors to produce well-formed data. We discuss support for robustness further below.

2.3 Supporting Robustness

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, thereby eliminating 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. On the semantic web, such mechanisms are impractical. First, we do not have a central administration of the data on the semantic web, and hence integrity constraints are difficult if not impossible to define. Second, enforcing integrity constraints would create another hurdle preventing people from joining the semantic web, rather than enticing them. Third, on the semantic web 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, a design goal of MANGROVE is *robustness*: authors should be able to add content without considering constraints, and services should be able to consume data that is cleaned and consistent as appropriate for their needs. Furthermore, when users do intend their data to be consumed by certain services, there should be a feedback loop that ensures that their data was in a form that the service could consume. Below we describe how MANGROVE supports robustness in such a large-scale data sharing environment.

Deferring integrity constraints: 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 [23], a user may not add information that contradicts another user unless the

[3] Very loosely speaking, this is analogous to checking which web pages link to your page – a service that is offered through search engines such as Google.

contradictions are explicitly identified first. Likewise, in SHOE [17], 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 annotated 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, MANGROVE allows users to decide how extensively to annotate their data. For instance, the `instructor` property may refer to a resource with further properties such as `name` and `workPhone`, or simply to a string literal (e.g., "`John Fitz`"). Permitting such "light" annotations simplifies the annotation of existing HTML and allows authors to provide more detail over time.

To complement the deferral of integrity constraints, MANGROVE provides three mechanisms that facilitate the creation of appropriate data for services: service feedback (discussed earlier), data cleaning, and inspection of malicious information.

Data cleaning: The primary burden of *cleaning* the data is passed to the service consuming the data, based on the observation that different 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 (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 provides a *service construction template* that enables services to apply a simple rule-based *cleaning policy* to the raw results obtained from the RDF database. For instance, for course events, our calendar specifies a simple policy that prefers data from pages specific to a particular course over data from general university-provided pages. Thus, factual conflicts (e.g., a location change not registered with the university) are resolved in the course-specific page's favor. The cleaning policy also helps the calendar to deal with different degrees of annotation. For instance, to identify the instructor for a course lecture, the calendar simply requests the value of the <`instructor`> property, and the template library automatically returns the <`name`> sub-property of the instructor if it exists, or the complete value of that property if sub-properties are not specified. Finally, the template also provides other rules to assist with data interpretation (e.g., to parse different formats of dates and times commonly found on the web, or those often found in a university setting, such as ‘‘`MWF 10-11 a.m.`’’). To utilize these features, services may create their own cleaning policy or use a default from the service template.

Malicious information: Another reason that we store the source URL with every fact in the database is that it provides a mechanism for partially dealing with malicious information. The highly distributed nature of the web can lead

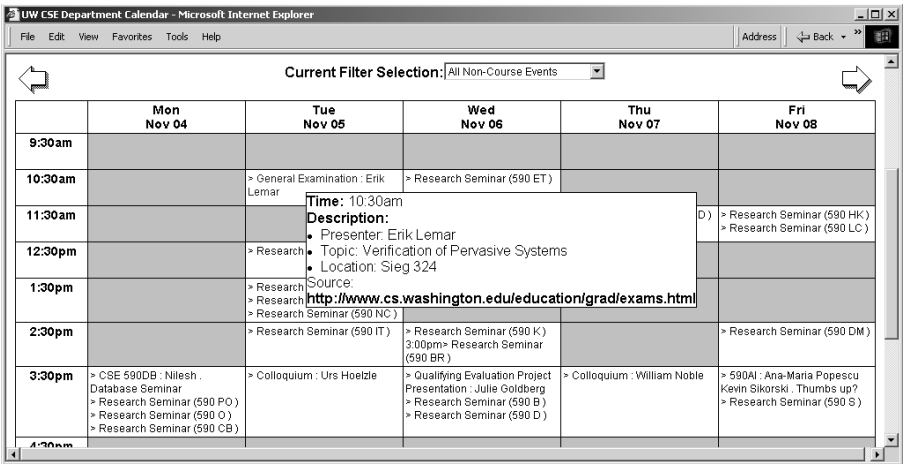

Fig. 3. The calendar service as deployed in our department. The popup box appears when the user mouses over a particular event, and displays additional information and its origin. For the live version, see `www.cs.washington.edu/research/semweb`.

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.

Thus, MANGROVE services associate an easily-accessible source (i.e., a URL) with each fact made visible to the user. For example, as shown in Figure 3, 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. Naturally, service writers are free to implement more sophisticated policies for identifying malicious information, 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 `www.cs.washington.edu/education/` to override facts originating elsewhere.

2.4 Discussion

As noted in the introduction, MANGROVE is designed to enable and entice authors to structure their data by mimicking some of the conditions that led to the explosive growth of content creation on the web. First, MANGROVE supports *instant gratification* with a loop that takes freshly published semantic content to MANGROVE services, and then back to the user through the service feedback mechanism. Next, MANGROVE supports *robustness* by postponing the enforcement of integrity constraints, associating a source URL with every fact in the database, and with the service construction template, which assists services in

cleaning and interpreting the data based on these URLs. Finally, MANGROVE supports *ease of authoring* by providing a simple graphical annotation tool, deferring integrity constraints to the services, and permitting authors to annotate HTML lightly and incrementally.

3 Semantic Services in MANGROVE

One of the goals of MANGROVE is to demonstrate that even modest amounts of annotation 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 annotated information, and semantic email. Below, we briefly discuss service construction and then consider each of the above services.

Services are written in Java and built on top of the MANGROVE service template that provides the basic infrastructure needed for service creation. Our implementation uses the Jena [25] RDF-based storage system, which enables our services to extract basic semantic information from the database by posing queries. In addition, the MANGROVE service template provides methods to assist with data cleaning and interpretation, as explained in Section 2.3. The template also aids service construction with support for incrementally computing and caching results. Overall, MANGROVE makes services substantially easier to write by encapsulating commonly-used functionality in this service template. At runtime, these services are then invoked by a Jakarta Tomcat servlet engine.

3.1 Semantic Search

We believe that annotation will be an incremental process starting with "light" annotation of pages and gradually increasing in scope and sophistication as more services are developed to consume an increasing number of annotations. It is important for this "chicken and egg" cycle that even light annotation 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 properties and text. The service returns the set of web pages in our domain that contain the text and properties 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 properties. 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 properties. The `?` after the `<portrait>` property instructs the service to extract and return the HTML inside that property (as with the SELECT clause of a SQL query). Users select appropriate properties for the search from the simple schema available on the search page; future work will consider ways to make this selection even easier.

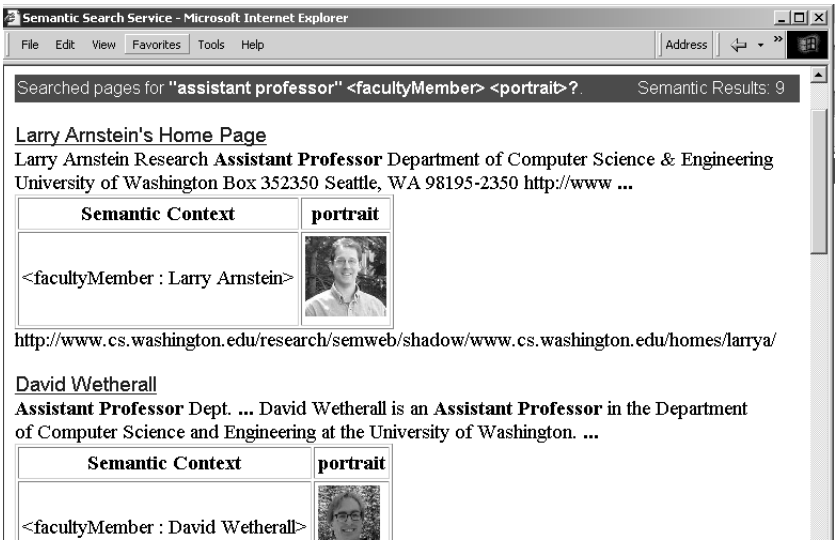

Fig. 4. The semantic search results page. The page reproduces the original query and reports the number of results returned at the top. Matching pages contain the phrase "assistant professor" and the properties <facultyMember> and <portrait>. The ? in the query instructs the service to extract the <portrait>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

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 properties 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 extracted information. The snippet is the `name` property of the extracted property's subject. 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 properties, this kind of search is different from the standard querying capability supported by MANGROVE's underlying RDF database and other semantic web systems such as SHOE [17] and WebKB [23]. Our search service has value to users even when pages are only lightly annotated, supporting our goal of enticing users onto the semantic web.

3.2 Aggregation Services

Aggregation services provide useful views on data from the semantic web. We describe the aggregation services we implemented with MANGROVE below.

First, our Who's Who service compiles pictures, contact information, and personal data 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, enabling users to update their own data at any time to reflect their changing interests.

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. Only a single <publication> property enclosing a description of the publication is required in order to add an entry to the database, which facilitates light, incremental annotation. However, users may improve the quality of the output and the duplicate removal by specifying additional properties such as <author> and <title>.

Our most sophisticated service, the department calendar (shown in Figure 3), 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 annotate their events' home pages in order to publicize their events. (In fact, the current contents of the calendar are projected on the wall in the lobby of our department.) 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 annotated 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 modified pages, the calendar is immediately updated, and users then view the calendar to verify that their events are 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 later.

3.3 Semantic Email

While the WWW is certainly a rich information space in which we spend significant amounts of time, many of us spend even more time on email. In the same spirit as the semantic web, adding some semantics to email also has the potential for increasing productivity. In fact, we often use email for tasks that are reminiscent of lightweight data collection, manipulation, and analysis. Because email is not set up to handle these tasks effectively, accomplishing them manually can be tedious, time-consuming, and error-prone. As another example of where instant

gratification can entice people to add more semantic structure to their data, we developed *semantic email processes* (SEPs) [10] using MANGROVE.

As an example of semantic email, consider the process of organizing a potluck by sending an email to a list of people, asking who will attend and what dish they plan to bring, and then automatically collecting the responses and tallying them up. The benefits of such an automated process provide significant incentive for people to structure their original request.[4] We model a SEP as an RDF *data set* affected by *messages* from a set of *participants*, controlled by a set of *constraints* over the data set. For instance, when executing we may constrain the potluck so it results in a balanced number of appetizers, entrees, and desserts.[5]

Implementing SEPs within MANGROVE enables us to synergistically leverage data from the web and email worlds in one system. For instance, our calendar service accepts event information from annotated pages that are published via MANGROVE or via semantic email. Likewise, SEPs such as our "RSVP" process could accept event descriptions from an annotated web page, then monitor this web data for location or time changes to include in a reminder email. Finally, human responses to semantic email queries (e.g., requesting a phone number) can be used to gradually acquire semantic knowledge over time. See [10] for a more complete description of semantic email, including the formal model of SEPs and a description of important inference problems that arise in this context.

3.4 Discussion

MANGROVE and our services have been deployed in our department for only a few months, but already permit a few observations. First, simple services such as the calendar can offer substantial added value over other forms of accessing the same information. For instance, in the five months the online calendar has been operational, it has received more than 2700 distinct visits, with an average of about two page views per visit.[6] Second, users are willing to annotate their documents if the process is easy and interesting services exist to use the annotations. For instance, a small but growing number of users have annotated their personal home pages in order to be included in the WHO'S WHO and to promote their publications. In addition, administrators, students, and faculty have all utilized annotation to promote a wide range of events, ranging from official departmental events to visitor schedules to informal events at a local pub.

These observations are not meant to be conclusive: the system and its services are new and still evolving. Nonetheless, our initial experience strongly suggests that the MANGROVE system and services are both feasible and beneficial. See [26] for additional measurements demonstrating that simple annotation of existing documents is feasible and that it can potentially improve both the precision and recall of search compared to Google.

[4] While the *organizer* must perform some structuring, we do not require *participants* in this process to understand semantics or use any special tools; see [10] for details.

[5] This SEP and many others are available for public use, see http://www.cs.washington.edu/research/semweb/email.

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

Scalability is an important design consideration for MANGROVE, and it has influenced several aspects of MANGROVE's architecture, such as our explicit publish/notification mechanisms. 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 (all users annotate data with a common local schema). 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 [13]. Specifically, once a department has annotated its data according to a local schema, it can collaborate with other structured data sources using a *peer-data management system* (PDMS) [14]. 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 [7,8], though these tools are not yet integrated into MANGROVE. Relying on a PDMS also distributes querying across a network of peers, eliminating the bottleneck associated with a central database.

4 Related Work

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.

Haustein and Pleumann [16] note the importance of semantic data being "immediately visible" in a way that yields benefit to content authors. Their system, however, primarily provides this benefit by eliminating redundancy between HTML and semantic data, and then using this data and templates to dynamically generate attractive HTML or RDF content. While these features potentially make maintaining interrelated HTML and RDF data more convenient, their system is very different from MANGROVE. Specifically, they have a different architecture that doesn't support explicit publication, notification, or service feedback. In addition, we have identified and deployed a set of instant gratification services as an essential part of MANGROVE, which are absent from their system.

Two other projects most closely related to our work are OntoBroker [6] and SHOE [17], both of which make use of annotations inside HTML documents. SHOE's services, like those of many other systems, primarily consisted of tools to simply search or view semantic data, although their "Path Analyzer" [18] provided a convenient interface for exploring relationships among concepts. OntoBroker did implement a number of services, such as a Community Web Portal [31] and services intended to assist business processes [29]. SHOE and OntoBroker, however, primarily rely upon periodic web crawls to obtain new information from annotated HTML, thus preventing instant gratification and

content creation feedback. In addition, MANGROVE has the advantage of enabling useful services even when content is only lightly annotated. For instance, while OntoBroker's "SoccerSearch" service [29] tries a semantic search and then a textual search if the former fails, MANGROVE's semantic+text search service can profitably combine both types of information.

As an alternative to crawling, some systems provide a web interface for users to directly enter semantic knowledge [23,6] or to instruct the system to immediately process the content of some URL [23]. 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.

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., [11,1,19]) do exactly this type of polling. The difference between MANGROVE and such 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.

In MANGROVE we chose to store annotations within the original HTML pages, for simplicity and to enable easy updates of the annotations when the source data changes. However, the overall architecture is also consistent with external annotation, where a user may annotate any page and the annotations are transmitted directly to a semantic database, as possible with CREAM [15], Annotea [20], or COHSE [2]. A side effect of these tools is that they automatically aggregate data as with our explicit publish operation; MANGROVE completes the necessary features for instant gratification by providing service notification, feedback, and a host of useful services.

The TAP semantic search [12] executes independent textual and semantic searches based on traditional text queries. This service is easy to use but cannot currently exploit information from one search in the other, nor can the user specify the type of semantic information that is desired. Recently, QuizRDF [5] introduced a search service that does combine textual and semantic content. QuizRDF's searches are more restricted than those provided by MANGROVE's search service, making it more difficult to use as a building block for other services. However, QuizRDF has an elegant user interface that more readily assists users in identifying relevant properties.

Information Lens [22] used forms to enable a user to generate a single email message with semi-structured content that might assist recipients with filtering and prioritizing that message. Mangrove's SEPs generalize this earlier work by enabling users to create an email *process* consisting of a set of interrelated messages governed by useful constraints. In addition, Mangrove extends Information Lens's rule-based message processing to support more complex reasoning based on information from multiple messages and data imported from web sources. Consequently, Mangrove's SEPs support a much broader range of applications than those possible with Information Lens [10]. More recently, Kalyanpur et al. [21] proposed having users semantically annotate messages to improve mail search, sorting, and filtering. This approach can potentially result in rich semantic content, but requires users to invest significant annotation effort for some *potential* future benefit (e.g., in improved searching for an old email) or primarily for the benefit of the recipient. SEPs instead generate both the semantic content and the text of the email message directly from simple forms, and provide instant gratification by immediately utilizing this content for simple but time-saving email processes.

For storing and accessing RDF data, we utilize the Jena toolkit [25]. Other systems that also offer centralized RDF storage include Kaon [27] and Sesame [4]. Edutella [28] extends these approaches to provide RDF annotation, storage, and querying in a distributed peer-to-peer environment, and proposes some services, 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.

5 Conclusion

This paper presented MANGROVE as a means of demonstrating how to entice non-technical people to contribute content to the semantic web. Specifically, the paper reports on the following contributions:

1. We highlighted three key conditions that are essential for the growth of the semantic web: *instant gratification* (*i.e.*, immediate, tangible value resulting from semantic annotation), *robustness* to malformed data and malicious misinformation, and *ease of authoring*.
2. We introduced the MANGROVE architecture that supports the complete semantic web "life-cycle" from content authoring to semantic web services. We demonstrated how elements of the architecture support each of our three design goals, particularly the explicit publish mechanism, service feedback, and deferral of integrity constraints.
3. We described several deployed semantic services that motivate the annotation of HTML content by consuming semantic information. We showed how these services can provide tangible benefit to authors even when pages are

only sparsely annotated. These are some of the first "semantic services" that are invoked by ordinary users as part of their daily routine. This deployment lends credence to the claim that our services are both feasible and beneficial.

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.

Acknowledgments. This research was partially supported by NSF ITR Grant IIS-0205635, DARPA contract NBCHD030010 for Oren Etzioni, NSF CAREER Grant IIS-9985114 for Alon Halevy, and by a NSF Graduate Research Fellowship for Luke McDowell. Thanks to Google and Corin Anderson for their assistance with our search service. We are also grateful to Abraham Bernstein, Natasha Noy, Valentin Razmov, Dan Weld, Oren Zamir, and the anonymous reviewers for their helpful comments on improving the paper. Mangrove's parser utilizes code from HTMLParser (http://htmlparser.sourceforge.net/) and Xerces-J (http://xml.apache.org/xerces-j/), and the calendar interface is based on WebCalendar (http://webcalendar.sourceforge.net/).

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. S. Bechhofer and C. Goble. Towards annotation using DAML+OIL. In *K-CAP 2001 Workshop on Knowledge Markup and Semantic Annotation*, 2001.
3. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.
4. J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: An architecture for storing and querying RDF data and schema information, 2001.
5. J. Davies, R. Weeks, and U. Krohn. QuizRDF: Search technology for the semantic web. In *Workshop on Real World RDF and Semantic Web Applications*, 2002.
6. 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.
7. A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: a machine learning approach. In *Proc. of SIGMOD*, 2001.
8. 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.
9. 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.

10. O. Etzioni, A. Halevy, H. Levy, and L. McDowell. Semantic email: Adding lightweight data manipulation capabilities to the email habitat. In *Sixth International Workshop on the Web and Databases*, 2003.
11. 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*, March 1997.
12. R. Guha, R. McCool, and E. Miller. Semantic search. In *World Wide Web*, 2003.
13. 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*.
14. A. Halevy, Z. Ives, I. Tatarinov, and P. Mork. Piazza: Data management infrastructure for semantic web applications. In *Proc. of the Int. WWW Conf.*, 2003.
15. S. Handschuh and S. Staab. Authoring and annotation of web pages in CREAM. In *World Wide Web*, pages 462–473, 2002.
16. S. Haustein and J. Pleumann. Is participation in the semantic web too difficult? In *First International Semantic Web Conference, Sardinia, Italy, June 2002*.
17. J. Heflin, J. Hendler, and S. Luke. SHOE: A knowledge representation language for internet applications. Technical Report CS-TR-4078, 1999.
18. J. Heflin, J. A. Hendler, and S. Luke. Applying ontology to the web: A case study. In *IWANN (2)*, pages 715–724, 1999.
19. 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.
20. J. Kahan and M.-R. Koivunen. Annotea: an open RDF infrastructure for shared web annotations. In *World Wide Web*, pages 623–632, 2001.
21. A. Kalyanpur, B. Parsia, J. Hendler, and J. Golbeck. SMORE – semantic markup, ontology, and RDF editor. http://www.mindswap.org/papers/.
22. T. Malone, K. Grant, F. Turbak, S. Brobst, and M. Cohen. Intelligent information-sharing systems. *Communications of the ACM*, 30(5):390–402, 1987.
23. P. Martin and P. W. Eklund. Large-scale cooperatively-built KBs. In *ICCS*, pages 231–244, 2001.
24. B. McBride. Four steps towards the widespread adoption of a semantic web. In *First International Semantic Web Conference, Sardinia, Italy, June 2002*.
25. 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.
26. L. McDowell, O. Etzioni, S. D. Gribble, A. Halevy, H. Levy, W. Pentney, D. Verma, and S. Vlasseva. Evolving the semantic web with Mangrove. Technical Report UW-CSE-03-02-01, February 2003.
27. B. Motik, A. Maedche, and R. Volz. A conceptual modeling approach for building semantics-driven enterprise applications. In *First International Conference on Ontologies, Dataases and Application of Semantics (ODBASE-2002)*, 2002.
28. W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. Edutella: a P2P networking infrastructure based on RDF. In *WWW*, pages 604–615, 2002.
29. Ontoprise. Demo applications. http://www.ontoprise.de/com/co_produ_appl2.htm
30. D. Reynolds. RDF-QBE: a Semantic Web building block. http://www.hpl.hp.com/semweb/publications.htm.
31. 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.