A Semantic Wiki for Continual Collaborative Information Management

John Recker, Tyler Close, Angela Maduko, Craig Sayers
HP Laboratories
HPL-2008-90

Keyword(s):
Wiki, Semantic Web, Ontology, RDF, OWL, Context, Semantic

Abstract:
The Grand Central semantic wiki is designed to merge the benefits of the quick and informal data entry styles common to wiki platforms with the robust data management capabilities of traditional enterprise content management systems. Incorporating many recent advances in web technologies, including wikis, semantic web technologies, RSS, and search as a primary data access interface, it seeks to speed and simplify collaboration in the enterprise. Based on user studies, we have developed an experimental prototype and are currently targeting a deployment to assist some field engineers.

External Posting Date: July 6, 2008 [Fulltext] Approved for External Publication

Internal Posting Date: July 6, 2008 [Fulltext]

Submitted to World Wide Web Conference 2008, WWW'08

A Semantic Wiki for Continual Collaborative Information Management

John Recker
Hewlett-Packard Labs
1501 Page Mill Rd
Palo Alto, Ca, 94304
john.recker@hp.com

Tyler Close
Hewlett-Packard Labs
1501 Page Mill Rd
Palo Alto, Ca, 94304
tyler.close@hp.com

Angela Maduko
LSDIS Lab,
Department of
Computer Science,
University of Georgia
maduko@cs.uga.edu

Craig Sayers
Hewlett-Packard Labs
1501 Page Mill Rd
Palo Alto, Ca, 94304
craig.sayers@hp.com

ABSTRACT

The Grand Central semantic wiki is designed to merge the benefits of the quick and informal data entry styles common to wiki platforms with the robust data management capabilities of traditional enterprise content management systems. Incorporating many recent advances in web technologies, including wikis, semantic web technologies, RSS, and search as a primary data access interface, it seeks to speed and simplify collaboration in the enterprise. Based on user studies, we have developed an experimental prototype and are currently targeting a deployment to assist some field engineers.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia – *Navigation, User issues.*

General Terms

Documentation, Design, Human Factors.

Keywords

Wiki, Semantic Web, Ontology, RDF, OWL, Context, Semantic

1. INTRODUCTION

The initial design of the Wide World Web centered on authors publishing documents using HTML. This was a largely "read-only" environment, with user participation typically limited to reading content and supplying short responses in web forms. Wikis [1], blogs [2], and other "Web 2.0" [3] web tools have changed all this, and they have recently seen explosive growth on the web. These tools focus on providing a more interactive web environment, and encouraging 2-way communications between web sites and users. As such, they are core technologies in social networking and the creation of web communities. However, in spite of their widespread adoption in the consumer arena, these tools have experienced uneven rates of adoption in the enterprise.

This paper examines our investigations into the road blocks slowing the adoption of these tools in the enterprise and our efforts to create a tool optimized for collaboration in an office environment. It begins by presenting the results of a user study of users of the corporate wiki at HP Labs. In particular, issues surrounding information discovery and security of corporate data were raised repeatedly by our interview subjects. Furthermore, our interview subjects include mobile, intermittently connected users whose needs differ from those with full time web access. We then present our approach to address these issues: a browser based collaboration tool designed to import, navigate and annotate pre-existing corporate information schemes. To address the needs of mobile users, the system relies on a local data cache to support off-line access, and uses filtered RSS 1.0 data feeds to provide a secure and flexible scheme for cache synchronization. Lastly, we present the Grand Central Wiki, an implementation of these concepts designed to support the collection of informal information generated by field service engineers.

2. USER STUDIES

The success of our notion of a semantic wiki largely depends on how it is received by users, thus to motivate our work, we interviewed nine users of the TWiki [4] wiki to learn about their experiences, and discover opportunities for system improvements. Interview subjects were selected from both different departments, and job categories with different technical inclination. Furthermore, subjects were selected, based on system logs, to represent different levels of wiki usage – *Readers* (primarily read-only users), *Editors* (users that contribute significantly to page content) or *Designers* (editors that also contribute new pages). Comments made by interview subjects were grouped into the following categories: *editing information*, *information retrieval*, *security/privacy* and *miscellaneous*. Table 1 below shows the coverage of our sample, based on this classification.

Table 1: Coverage of our Sample

	Technical
Designers	3
Editors	5
Readers	1

We summarize the feedback obtained in the following subsections.

2.1 Editing Concerns

Nearly all editors and designers noted a steep learning curve for learning the TWiki Markup language and many asked for a more visual interface (note that TWiki has since added a WYSIWYG editor option). Users also wanted simpler options for setting preferences in the wiki, moving attachments, changing styles, sorting table cells and table management in general, using bullets, in-lining images with text, creating new or appending to existing wiki pages from email contents, inserting mathematical formulae and allowing multiple editing views with an easy way to toggle across views. A near universal complaint was the difficulty in importing HTML into wiki pages. All users noted this feature is lacking in the wiki and pointed out its importance, especially in a scenario where there is a desire to move many existing html pages to the wiki. Some users suggested that the collaborative editing of wiki pages be extended to attachments, in other words, a mechanism for locking attachments that have been downloaded and being viewed or edited by users should be provided.

Most users were of the opinion that TWiki help pages were difficult to locate and use, although there was disagreement about better solutions. Some believe it should be laid out in one page shown during the edit mode while others proposing an indexed help pages with browse and/or search options. Some wanted an optional tutorial that should start up when the wiki home page is loaded. Lastly some more technical users wanted to program the wiki with, for example, a script that performs a particular function within a wiki page.

2.2 Information Retrieval Concerns

Information retrieval concerns were grouped around two topics: wiki inter-page links and information discovery. Several comments were directed at the wiki inter-page linking mechanism. Some designers thought the CamelCase page syntax too restrictive and often incompatible with the natural title for a wiki page. Some noted that, in the event that pages are renamed or moved, external links to the old pages are not updated. To this effect, they suggested supporting redirections from the old pages to the new ones. Some readers found it distracting to always have to follow links to retrieve more information about the anchor text and proposed the use of tooltips to provide additional information about the anchor text or even a summary of the linked page. Suggestions were made to allow wiki pages to age gracefully, targeting pages created for past events which contains information that is no longer needed.

TWiki allows users to classify pages by category. Some technical editors and designers proposed incorporating the ability to detect and suggest the possible categories into which a page can be placed into the wiki, through, for example, an analysis of the page contents. They also proposed applying this process of page content analysis to page similarity detection which they claim is useful for avoiding page replication.

Information discovery in wikis is typically done through unstructured keyword searches or by browsing the link structure of the wiki. One topic in the user interviews focused on the ease or difficulty of information retrieval in wikis. All technical users noted that when a search is performed within a category, results returned are only pages with in that category. Some believe that users should be given the option of choosing if and how searches should be expanded to include results from other categories. In line with this, they preferred a mechanism through which they can specify some similarity level for results from the expanded search. However, some other uses believe that it is not a good idea to expand searches to other categories. Their argument being that by browsing to a particular category before performing a search, a user means to narrow down the search, thus requiring a more focused search. In addition, they believe that such an expansion will be undesirable if it incurs a lot more time overhead. Finally, some users also want searches to be extended to the contents of attached files.

Technical users noted that the categorization mechanism in the wiki is flat with only one fixed way of being browsed. They expressed interest in hierarchical categorization which can be browsed using multiple views that keep a trail of the users browsing activities. In this scenario, a user can browse for a certain set of pages starting from different categories. For example a set of pages related to projects can be reached by browsing from either a project or a customer category. While they want to have a more hierarchical browsing technique, they also desire that the breadth of such hierarchies be coarse so as to avoid information overload. Users noted that another source of information overload stems from the presentation of search results. Search results are mostly wiki pages that are deemed relevant to the search criteria. However, users pointed out that it is often the case that they have to sift through these pages to get to the actual information sought. In other words, the desired information is often embedded in a page along with other non-desirable content. To this effect, they consider having a finer granularity of search results which obviates this sifting process a much more desirable approach.

2.3 Security/Privacy Concerns

While all users are unanimous in the need for security in a wiki, there is a great disparity in users' responses about the level of security that should exist in a wiki. For example most users expressed interest in a personal space that is accessible only to them. Some users thought that users will be encouraged to add comments to pages if they have the ability to restrict access to their comments. Others believed this would hinder the inherent collaborative nature of a wiki. Some wanted the ability to limit changes to a subset of a page – for example, some designers have some reservations about changes being made to page structure and templates and wanted a provision for specifying the level of structural changes expected on wiki pages and templates.

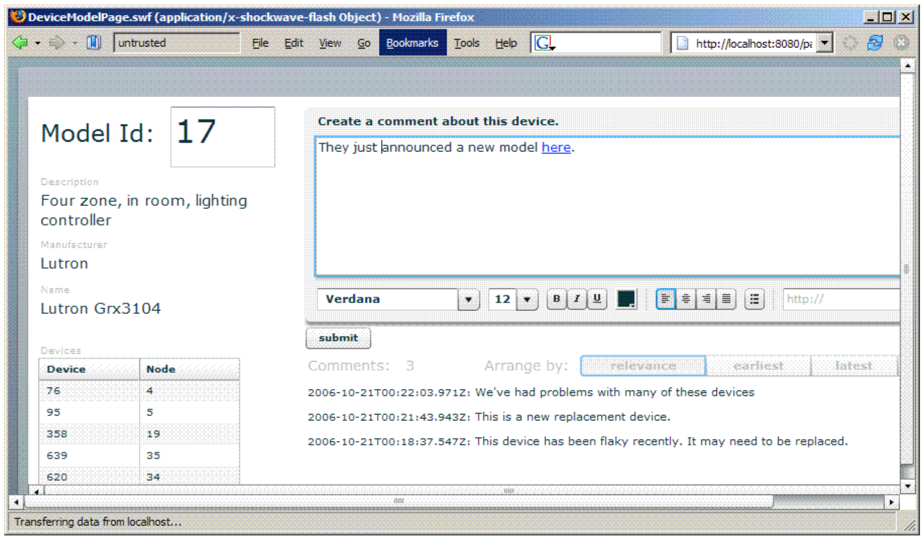
Authentication was a frequently mentioned issue, particularly in view of the intra-company use of the wiki. Comments are currently not automatically signed even when users are logged on.

Not only do most technical users believe that users should be authenticated before they are allowed to add comments to a wiki page, they also want users to be given the option of authenticating to view or edit wiki pages. Furthermore, technical designers want users' activities in the wiki to be audited to dissuade personalization and vandalization.

Technical users also proposed the user of groups of users as well as roles of users for access restriction purposes. In addition, they proposed having update notification integrated with security so that one is able to specify who should be notified of the changes they have made to certain pages and about what topics.

2.4 Other issues

Besides the above issues raised by the users, technical designers also raised a concern of availability of the wiki at all times, especially outside the company's intranet. In the light of this, they proposed having an offline version of the wiki with synchronization of updates when online. Users also want the ability to link up multiple wikis so that content within one wiki can be accessed from another.


DeviceModelPage.swf (application/x-shockwave-flash Object) - Mozilla Firefox
untrusted
File
Edit
View
Go
Bookmarks
Tools
Help
http://localhost:8080/p
Verdana
12
http://
submit
Transferring data from localhost...

Model Id: 17

Description
Four zone, in room, lighting controller

Manufacturer
Lutron

Name
Lutron Grx3104

Devices

Device	Node
76	4
95	5
358	19
639	35
620	34

Create a comment about this device.

They just announced a new model here.

Comments: 3 Arrange by: relevance earliest latest

2006-10-21T00:22:03.971Z: We've had problems with many of these devices

2006-10-21T00:21:43.943Z: This is a new replacement device.

2006-10-21T00:18:37.547Z: This device has been flaky recently. It may need to be replaced.

Figure 1: An example screenshot from our Grand Central Wiki. This shows the view of a lighting controller. On the left hand side is information derived automatically from an operational relational database. On the right a new comment is being created about this model of device. On the lower right are several existing comments sorted by relevance. Significantly, these comments were not entered on this "page" but are relevant to the page (because they are about devices of this model type) and were discovered by search.

A set of questions we asked related to the use of metadata as a means of improving wiki information access. Generally, all the users we interviewed shared our views that there were benefits to be gained with the use of metadata, and that the benefits from collecting information metadata could outweigh the inconvenience that may be associated with metadata collection. However, users indicated their willingness to provide metadata only so long as it is an extremely simple process.

Based on those user comments we have focused on developing a wiki which enabled offline operation, leveraged existing corporate data schemas, and used search as the primary interface. The result is the Grand Central Wiki.

3. THE GRAND CENTRAL WIKI

Many of the issues identified in the user survey are problematic in a corporate environment. Security, for example, is essential – restricting data access to an appropriate set of users can be dictated by legal mandates and premature release of product information can adversely affect the success of a product in the market. On the other hand, the success of the web 2.0 tools suggest that they are effective tools for promoting collaboration – clearly as desirable a goal within an enterprise as without. The Grand Central semantic wiki seeks to preserve the benefits of the quick and informal data entry styles common to wiki platforms while addressing the navigation, security and off-line access concerns of users.

3.1 Navigation

Navigation in a typical wiki can be awkward since the burden of structuring information is placed on contributors. The manner in which information is structured by design evolves in an ad-hoc fashion, and contributors can differ significantly not only in their understanding of the material being discussed but also in their familiarity with the tool used to enter the information – all of which significantly influences the final data layout.

And yet, defining and implementing an optimal structure is hard work. Enterprises already dedicate significant effort to organizing their assets – as seen, for example, in source code hierarchies, relational database schemas, and organization charts. Fundamental to our approach is deriving information organizations from these pre-existing structures. In particular, RDF [5] ontologies are created from the terms and relationships from these existing structures and data, then a skeleton wiki is populated with terminology and relationships already familiar to the target user community, such that contributors need only fill in their comments.

3.1.1 Faceted Browsing

Navigation in Grand Central is performed by faceted browsing [6] of the ontologies derived from enterprise data, where each facet is a node in the skeleton wiki. At each step, the navigation interface presents both an *intensional* and *extensional* view of the facet. The intensional view allows browsing in terms of the hierarchical relationship that exists amongst concepts in the ontology to which they belong. The extensional view presents the set of entities that are instances of the domains or ranges of the facet view. Our choice of this approach stems from users needs for the ability to browse for pages through different categories.

The navigation views are generated from queries of a RDF data store containing RDF ontologies. It is expected that these data ontologies will be created primarily through automatic tools from existing knowledge structures with perhaps some manual intervention. Manual intervention in this step is not seen as a problem. First, the goal of importing external knowledge organizations is to leverage the knowledge of domain experts and move away from ad-hoc information structures. Thus the assistance of experts at this step is essential. Next, while it is seen as crucial to import terminology and relationships from pre-existing user generated information hierarchies, such hierarchies often also include structures unique to the tool managing the data, as opposed to intrinsic to the data itself. Selecting concepts for inclusion in the wiki is best done once by someone intimately familiar with the imported information and its structure.

Grand Central site navigation can then proceed using any of several mechanisms: the wiki can be navigated by moving along the arcs of the ontology graphs loaded into the system, users can specify a more traditional text search, or users can specify a number of classes from the data ontology directly as search criteria. Alternatively, these approaches can be combined: the user can navigate the data ontology while simultaneously restricting the results using keywords or other tests. Once a topic of interest is found, users of the Grand Central software then fill out the skeleton wiki by directly composing their comment there. Such a skeleton wiki can be created for any data source that can be translated to RDF, making most enterprise data sources ready subjects for wiki style commenting and collaboration.

3.2 Comments

Unlike a conventional wiki, Grand Central is not based on web pages, but rather uses a smaller unit of a single comment. Comments are simply information blocks, with any number of formats: text blocks, documents, images, etc… Text comments are entered or edited with a simple WYSIWYG editor, while files are uploaded with a simple form interface.

```
<rdf:RDF
    xmlns:sw="http://hpl.hp.com/SmartWiki.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xml="http://www.w3.org/2001/XMLSchema#"
    xmlns:dc="http://purl.org/dc/elements/1.1/" >
  <rdf:Description rdf:about="sw:ISDNLine">
    <rdf:type rdf:resource="sw:ISDNLine"/>
    <rdf:type rdf:resource="owl:Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="sw:comment-262800539863397">
    <sw:replacedBy rdf:resource="sw:comment-3823569850843"/>
    <rdf:type rdf:resource="sw:Comment"/>
  </rdf:Description>
  <rdf:Description rdf:about="sw:comment-3823569850843">
    <dc:description>ISDN issue description.</dc:description>
    <dc:date rdf:datatype="xml:dateTime">
           2006-10-11T22:43:14.477Z
    </dc:date>
    <dc:title>ISDN Issues</dc:title>
    <sw:commentAbout rdf:resource="sw:ISDNLine"/>
    <rdf:type rdf:resource="sw:Comment"/>
  </rdf:Description>
</rdf:RDF>
```

Figure 2: Typical RDF comment description.

Comments in this system are stored as a collection of RDF statements in a RDF data store. In addition to properties one might expect such as creation time and date and author, comments are identified as relevant to one or more terms from the SQL RDF ontologies described above. These terms associations are automatic and derived from the subject of the page when the comment was created. Text based comment content is also stored in the RDF store, while binary files are assigned a unique name and stored in the file system.

Comments are version controlled (including binary objects), and a history of each comment is maintained and can be retrieved by the system. To support this capability, each revision of a comment is assigned a 128 bit randomly generated ID (GUID) (generated using the java.security.SecureRandom package**)** on creation, which is unique across all Grand Central systems. Comments that are edited are deemed "replaced" by the new revision and the old revision's RDF description is qualified with a "*sw:replacedBy*" attribute identifying the GUID of the next newest revision, resulting in a oldest-to-newest directed graph revision history. Note that a single comment can replace multiple comments, acting, for example, as a summary of the group of replaced comments. Comments currently store the entire new content of each comment. Eventually it would be desirable to transition to a difference based delta-compression scheme (such as implemented by RCS [8] or other version control systems).

3.3 Page Composition

A Grand Central wiki page is composed on the fly by searching the RDF comment data store for comments related to the page topic (where the topic is one or more URI from a supporting ontology) and sorting the result. The search is specified using SPARQL [9], a W3C standard for querying semantic stores. SPARQL supports multiple clauses in a search specification, so a result set can be refined by specifying any number of restrictions. Search criteria might include both URIs from the page topic, as well as items that are members of the super-class of the selected class. Whereas a generic search engine can only offer a sorted list of results, knowledge of the ontology enables special treatment of particular results and inclusion of results that might otherwise require further searching. As a result of broadening the search criteria, user contributions are displayed not only on the page where it was created, but also on all pages about a database record closely related to the original topic.

Comments retrieved from the above search can be formatted differently depending on the target application. By default, the list of user contributions on each page is sorted by relevance. For example, comments directly associated with the URI topics of the page are displayed first, while comments indirectly related to the topic would be displayed second. This fits our initial target application, which is to collect informal comments generated by the user community. However, wikis, blogs and mailing list browsers can be seen as presenting different user interactions of a similar document database. In the case of wikis, the user interface displays the latest version of a document and previous versions are accessible as history. In the case of a blog, messages are displayed in chronological order. Mailing list browsers order messages both chronologically and by subject. Our long term goal is to support each of these interaction methodologies.

3.4 Content Synchronization

The comment database can be synchronized with other Grand Central systems. The approach used is an example of an algorithm for "optimistic replication" [10]. In these algorithms, to update an object, a user submits an operation at some site. The site locally applies the operation to let the user continue working based on that update. The site also exchanges and applies remote operations in the background. Such systems are said to offer "*eventual consistency"*, because they guarantee that the state of replicas will converge only eventually. These systems can further employ "*epidemic propagation*" [11] to let any two sites that happen to communicate exchange their local operations as well as operations they received from a third site - an operation then spreads like a virus does among humans.

Grand Central publishes the content of its comments store as an RSS 1.0 [12] formatted RDF/XML document. The RSS feed published by the system includes the GUID of each comment in the system's comment store. This GUID is used by the system to implement a distributed version control system using RSS 1.0 & HTTP for change discovery and data transport. Systems synchronize their document stores by comparing the RSS feeds from remote systems with the contents of its own document store and selectively download missing document.

Internally, Grand Central generates the RSS feed by issuing SPARQL queries to the RDF data store and reformatting the resulting XML document into RSS 1.0 protocol. To reduce the size of the RSS data feed, the RSS feed URL supports a SPARQL query parameter which allows the queried system to filter items published to the feed. This RSS feed query parameter can then be simply combined with the internally generated query to restrict the domain of the results. Typical query parameters would limit feeds to comments on topics of interest, or comments published after a certain date.

Reverse synchronization (e.g. synchronizing the remote database with the local database) then occurs in one of two ways. In the case where both systems have public addresses, the remote database system initiates its own synchronization operation. In the event where the local machine address is not publicly known or accessible (as is often the case with systems whose address is assigned by DHCP or the like), the local system will push comments present in the local database and missing in the remote database to the remote system by posting comment descriptions to the RSS channel using a HTTP POST.

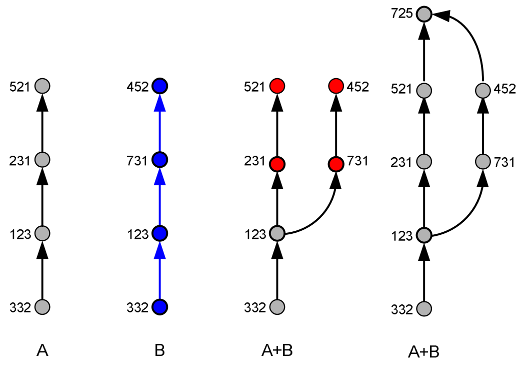


Figure 3: Merging revision trees. A) local comment store. B) remote comment store. C) merged comment store. Nodes in red are "Conflict" comments. D) possible conflict resolution.

Merging comments from two separate stores currently consists of merging the revision history trees from the two stores and identifying conflicts. The synchronization engine flags new comments from an imported tree with ancestors in common with new comments in the local comment store as "*conflict*" comments. Currently the system makes no effort to merge the contents of conflicting comments – the UI identifies conflicting comments and displays them together, and it is left to the user to decide how to best resolve conflicting changes. Users can remove conflict designations by one of two means. A user can simply remove the conflict designation, which applies to the selected comment and all older comments from the revision history. Alternatively, the user can combine multiple comments into one new comment, which similarly removes the conflict designation from comments in the revision history of each combined comment.

A Grand Central instance can synchronize with any number of other Grand Central instances, thus the wiki can span any number of projects or interests. In typical operation, it is expected that a Grand Central instance running on some back-end server will act as a central location for comments related to one or more projects. However, Grand Central does not require on-line access to a back-end wiki server for operation. Offline behavior of our wiki is the same as on-line behavior, with the exception that comments may be stale and not reflect the latest modifications. Updates between, for example, mobile nodes, can be performed when the central system is inaccessible. Eventually, however, it is presumed that the mobile nodes will synchronize with the back-end service. A Grand Central instance need not, therefore, be aware of every Grand Central instance to maintain an up-to-date comment database.

Synchronization is typically performed periodically as RSS syndication provides for regular queries of data feeds (it can also be user initiated), so in on-line use local and remote comment modifications will be synchronized with a latency of a few minutes. As a further benefit of this approach, a Grand Central database can be synchronized with any RSS 1.0 data feed. Thus an alternate view of the Grand Central wiki is as a read-write RSS aggregator.

3.5 Security

Security is managed by restricting content exported through the system's RSS data feed. Allowing users to perform unrestricted searches of a system's comment database would allow any user to access any or all documents in the system. Instead, items are published through a filter, or database view, which allows the system administrator to limit the items published to the RSS data feed. This RDF dataset view mechanism is implemented using the language described in [13]. The component which formats the RSS feeds issues its SPARQL queries to the entry point filtered by this view. The set of items that is published to the RSS feed is then the intersection of a client's query with the view presented by the content publisher. While this provides a powerful mechanism, it is also complicated for end-user control and we plan on exploring alternative user interfaces.

```
<rdf:RDF
    xmlns:sw="http://hpl.hp.com/SmartWiki.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <view:View rdf:ID="ISDNLineCommentView">
     <view:SelectAll/>
     <view:SelectInd rdf:about="#ISDNLineComments" />
  </view:View>
  <view:DefineClass
     rdf:about="ISDNLineComments"
     view:query="SELECT ?x
                WHERE (?x sw:commentAbout sw:ISDNLine);" />
</rdf:RDF>
```

Figure 4: This view specification, when applied to a RSS feed, limits access to comments about sw:ISDNLine devices.

Like many technologies designed for use on the open Web, RSS does not specify a means for access control. Just as any HTML file on a web server, an RSS feed is available to anyone with knowledge of the URL. However, publishing database updates via RSS is attractive due to the widely available client software for receiving and viewing the updates, so an access control technique that preserves this interoperation is needed. We intend to build on the Web Calculus [14] work and secure Grand Central RSS feeds by embedding an un-guessable secret in the https URL identifying the RDF view granted to a particular user. The RSS client software continues to fetch and display content as before, but the server can use the embedded secret to determine the user's authorization and so correctly restrict the RSS items returned by queries against the database. The system will then be capable of exporting many RSS “channels”, where each channel can expose a different view of the underlying database, thus exposing different datasets to different individuals or group of clients.

4. THE IMPLEMENTATION

We have built a prototype implementation with the goal of assisting field engineers for a conferencing product. The goal was to collect and manage informal information generated by these users. The product team already supports two formal data stores: version controlled software and project documentation, managed using a source code control system, and a relational database containing data necessary for the day-to-day operation of a conferencing system including information such as room and device configuration data. Our goal was to collect and manage data that did not fit one of these two criteria. This data came in a variety of forms, and included product documentation for external devices integrated into the system, and practical learnings of technicians in the field – information such as peculiarities of specific devices or even the best restaurants near a specific installation - information which is currently not tracked and non-uniformly spread by email or word of mouth.

Grand Central was customized for this application and was tightly integrated with the product team’s formal data store. SquirrelRDF was used to both extract a data ontology from the project’s relational database, as well as to extract detailed system information. This allowed comments to be associated with “hard” data extracted from the relational database, using the data organization, names & identifiers with which the field service organization is already accustomed. Comments could be associated with video conferencing rooms, locations, individual devices, or models of devices. Furthermore, because the ontology generation was automatic, we didn’t require the services of a knowledge engineer or extensive interviews with the target community to create the ontology for this system. An example screenshot of the current prototype is shown in Figure 1.

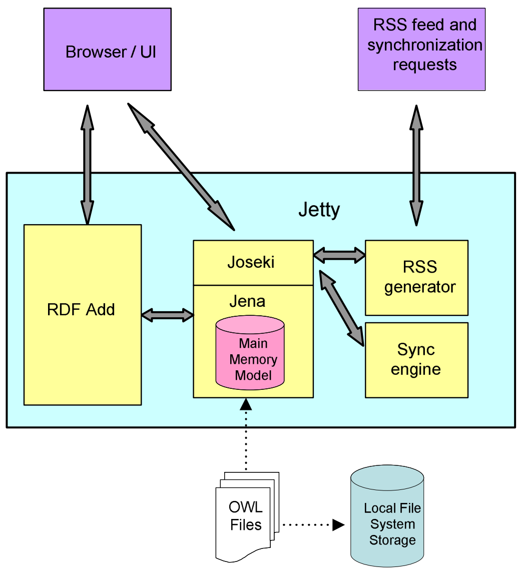


Figure 5: Implementation system architecture.

4.1 System Architecture

The Grand Central system was implemented using a client-server architecture. Our goal was build a system with an “intelligent” client to minimize scaling problems, and a server which minimizes the use of proprietary protocols.

4.2 The Client

The architectural goal of the client was to implement the “intelligence” and user interface for the application in the client, and to rely on the server only for data provisioning. Furthermore, we wanted a browser based solution to maximize portablity. We therefore is chose Adobe’s FLEX [15] system for our client development. The primary functions implemented by the client were page rendering (using the widgets and graphical editor provided with the FLEX development environment), formatting SPARQL queries, populating FLEX widgets from the XML documents returned from the SPARQL queries, and creating new RDF comments from the new comment forms.

The client user interface (see Figure 1) was divided into two components. The left side was for navigation. A HTML frame was composed by the client from SPARQL queries of the underlying data ontology. Headers are supplied for each device class referenced and referencing the class of device currently displayed, linked to the relevant device type. In our RDF dataset, these are statements where the current device is either the subject or object of attributes of another device type. This is the intensional view of the device. The extensional view depends on the page subject type. If the page subject is an individual device, then device specific attributes are displayed. If the page subject is a device type (or device class in our data ontology), then all device instances are displayed in a table. Item descriptions are extracted from the relational database.

The right side of the client user interface was dedicated to managing comments. The page subject is a URI from the data ontology embedded as URL encoded parameter in the page URL. Much like the navigation component, the component first issues a SPARQL query to discover related items. In the case of a device instance, this is the device class type. In the case of a device class, these are the instances of the device. The component then issues a query to retrieve all relevant comments. Comments are sorted first according to relevance to the page topic, then chronologically and displayed on the page. This component also allows users to create a new comment using a FLEX WYSIWYG editor. On input, comments are formatted into an XML-RDF comment description, and submitted to the server. The comment display is then re-rendered.

4.3 The Server

The server was implemented in Java using three Java servlets running in a Jetty [16] web server. The first servlet was a Joseki [7] RDF data store. Joseki implements a SPARQL endpoint, and responds to all SPARQL queries issued to the system. Joseki does not easily support run-time changes to its configuration, so configuration changes made to support changes in security policies such as changing the view specifications associated with individual RSS channels, creating new channels, or modifying the un-guessable secret embedded in a channel URL are performed by editing Joseki’s RDF configuration files. The Java JMX framework, which provides low-level operational control of a web server, and any standard JMX client, such as JConsole or MX4J can then be used to remotely restart the server with the new configuration. [27]

The second servlet was a Jena based application which allows a client to add RDF to the same data store accessed by the Joseki component. This component uses a proprietary protocol, although it is envisioned that this will eventually be replaced by SPARQL/Update [17] or some similar language for RDF graph updates. The last servlet generates RSS 1.0 data feeds from the RDF data store. Rather than accessing the data store directly, this component issues SPARQL queries to the Joseki servlet and the results reformatted into RDF 1.0 protocol. It is likely that this servlet could eventually be replaced by a client based XSLT script.

5. RELATED WORK

The Platypus wiki [18] is probably the first system to describe a wiki page as an RDF resource and qualify inter-page links with RDF properties. The Semantic MediaWiki [19] leveraged this approach to improve search, sorting and re-use of data. [20] examined the importance and explored the reuse of corporate metadata schemas by importing existing ontologies into the Semantic MediaWiki. Their purpose is similar to ours: to bootstrap a skeleton for filling a wiki. [6] discussed navigating along conceptual dimensions using hierarchical faceted metadata, an approach we use extensively in our prototype. [21] explored the idea of building templates, which in turn are used to generate HTML forms, directly from RDFS ontologies. Their goal, like ours, is to allow experts to create data hierarchies and allow users to focus on content authoring. [22] predicts a “Social Semantic Desktop” which merges the Semantic Web, Peer-to-Peer (P2P) Networks, and Online Social Networking. Using RDF encoded metadata to improve data exchange across P2P networks has been explored in projects such as [23]. The query, replication and annotation services featured in their architecture resemble aspects of our implementation. In our system, however, synchronization is explicit and performed one peer at a time, so problems such as distributed search do not apply.

There exist other examples of wikis built on top of distributed databases. Repliwiki [24] is an example of a Wiki that supports a distributed database. However, their purpose is different: Repliwiki seeks to replicate the wiki database for reliability and bandwidth purposes, not personalization and off-line access. Wooki [25] is closer is spirit to our approach. This system maintains wiki page consistency on a P2P network using automatic document merges and local broadcast of page changes on an overlay network. However, the back-end servers that are seen as problematic for the Wooki project are seen as a feature here: safeguarding corporate data is an essential operation to most corporations, which typically have both IT departments and policies to insure that centrally located data is not lost. Lastly, there exists a number of open source distributed version control systems including, for example, Git [26] which is used for the distributed development of the Linux kernel. Like our system, they are based on exchanging uniquely named “patches”, and we originally considered building the synchronization system using such a system. However, this would have made using RSS feeds for advertising content more complicated – important both for users wanting to browse system content with standard RSS readers, and importing data from generic RSS feeds. However, the delta compression, integrated merging and extensive testing of these systems is compelling, and it is likely that our system could benefit from integrating our RSS feed and security mechanism with one of these proven solutions.

6. CONCLUSIONS AND FUTURE DIRECTIONS

We have integrated technologies from the semantic web, semantic wikis, and RSS synchronization into a collaboration system for intermittently connected users. This addressed many of the concerns raised by our potential users. We are now considering how we might generalize the system architecture to address problems from different domains. For example, we would like to parameterize the user interface to make it easier to import different data ontologies, and find easier ways for users to manage data security. The challenge is to expose the flexibility of our semantic web technology based infrastructure without overwhelming non-expert users. We also would like to address the difficulty some users had in learning a tool by better leveraging existing user interface paradigms. If these challenges can be met, then we believe that this framework is appropriate for a next generation of web users: mobile intermittently connected corporate users with overlapping concerns and interests. We see significant value and opportunities in our approach for integrating existing business data structures with easy to use collaborative tools.

7. ACKNOWLEDGMENTS

The system is built using Jena and Joseki – many thanks to all who helped build those tools. Thanks to the many wiki users who consented to be interviewed. Thanks also to Jeff Thielman, Ted Beers, and Mike Bialek as well as to Paolo Castagna and our colleagues here in MMSL and particularly the Personal Collaboration group for their help and advice.

8. REFERENCES

[1] Cunningham, Ward and Leuf, Bo: “The Wiki Way. Quick Collaboration on the Web”. Addison-Wesley, 2001

[2] Wikipedia: blog, http://en.wikipedia.org/wiki/Blog

[3] O’Reilly, Tim, “What is Web 2.0”, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, 09/30/2005

[4] TWiki wiki: www.twiki.com

[5] RSS-DEV Working Group: “RDF Site Summary (RSS) 1.0”, http://web.resource.org/rss/1.0/, May 5, 2001

[6] Yee, K.P., et al, “Faceted Metadata for Image Search and Browsing”, In Proc. SIGCHI ’03, pp. 401–408

[7] Joseki: http://www.joseki.org/

[8] Tichy, Walter, “RCS - A System for Version Control”, Software – Practice and Experience, July 1985,: pp 637-654

[9] Prud'hommeaux, and Seaborne, Andy, editors, “SPARQL Query Language for RDF”, W3C working draft, 4, October 2006.

[10] Y. Saito and M. Shapiro, “Optimistic replication”, ACM Computing Survey, 37(1), 2005.

[11] Demers, A. J., Greene, D. H., Hauser, C., Irish, W., and Larson, J., “Epidemic algorithms for replicated database maintenance”, In 6th Symp. on Princ. of Distr. Comp. (PODC), . Vancouver, BC, Canada, 1987, pp. 1–12.

[12] RSS-DEV Working Group: “RDF Site Summary (RSS) 1.0”, http://web.resource.org/rss/1.0/, May 5, 2001

[13] Manjunath, G. et al, “Implementing Views for Controlled Access to the Semantic Web”, Workshop on Semantic Web for Collaborative Knowledge Acquisition (SWeCKa), 2007

[14] Close, Tyler “Web Calculus”, http://www.waterken.com/dev/Web/

[15] Adobe Flex: http://www.adobe.com/products/flex/

[16] Jetty web server: http://www.mortbay.org/

[17] Seaborne, Andy and Manjunath, Geetha, “SPARQL/Update: A language for updating RDF graphs”, Version 2: 2007-08-09, http://jena.hpl.hp.com/~afs/SPARQL-Update.html

[18] S. E. Campanini, P. Castagna, and R. Tazzoli, “Platypus wiki: a semantic wiki wiki web”, Proc. of 1st Italian Semantic Web Workshop, Dec 2004.

[19] Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H., Studer, R., “Semantic Wikipedia”, Proc. of the 15th International WWW Conference, Edinburgh, Scotland. (2006)

[20] Vrandecic, D., Krötzsch, M., “Reusing ontological background knowledge in semantic wikis”, Proc. of the 1st Workshop on Semantic Wikis, Budva, Montenegro. (2006)

[21] Kawamoto K, Kitamura Y, Tijerino Y, “KawaWiki: A Semantic Wiki Based on RDF Templates”, Web Intelligence and International Agent Technology Workshops, IEEE/WIC/ACM International Conference, 2006, pp. 425-432.

[22] S. Decker and M. Frank. “The social semantic desktop”. Tech. Rep. 2004-05-02, DERI, 2004

[23] Wolfgang Nejdl, et al, “EDUTELLA: a P2P networking infrastructure based on RDF”. In 11th World Wide Web Conference, May 2002: pp 604-615

[24] Sharma, Vikram, “Summary Hash History for Optimistic Replication”, USENIX ATC 2007

[25] Stéphane Weiss, Pascal Urso and Pascal Molli, “Wooki: a P2P Wiki-based Collaborative Writing Tool”, In Web Information Systems Engineering, Nancy, France, December 2007.

[26] Git version control system: http://git.or.cz/

[27] M. Mesarina, Venugopal K.S, N. Lyons and C. Sayers, “A Management and Performance Framework for Semantic Web Servers”, WWW 2007, Banff, Canada