Semantically-Based Active Document Collection Templates for Web Information Management Systems

In-Young Ko, Robert Neches, and Ke-Thia Yao *

Information Sciences Institute, University of Southern California, 4676 Admiralty Way, Marina del Rey, CA 90292, USA
E-mail: {iko, rneches, kyao}@isi.edu

Abstract. Representing and processing semantic information regarding individual documents is desirable but not sufficient. To improve the efficiency and reusability of users' work with Web-based information management systems, it is essential to handle document *collections*. We describe techniques for representing semantics both of collections and of information management services that operate upon them. These techniques help users set up complex analyses and structurings of information collections, adapt their work for other analyses or for different collections, and obtain automatic refreshing and updating for collections with content that changes over time. Our semantic representation helps identify and sequence appropriate analysis and visualization services for a given task. *Templates* capture these sequences in terms of active semantic relations between document collections created and manipulated during those tasks. Templates can be dynamically modified and instantiated to generate document collections for similar tasks, or to refresh an information space with time-varying document membership. They can also be exchanged, allowing others to reapply them.

Keywords: document collection semantics, active document collection templates, Web information management systems, component-based architecture

1 Introduction

When searching, analyzing, or structuring information from the Web, users deal with large document collections composed of multiple Web documents retrieved from various Web resources. Although some Web resources (e.g., Web directory services like Yahoo) return structured information such as categorized document collections, document collections retrieved from the Web are usually unorganized and too big to easily browse. Information analysis and visualization functions are needed to operate upon the document collections in order to characterize, sort, partition and filter them. This paper explains how the semantic modeling and reasoning mechanisms at the *document collection* level can help improve the efficiency and reusability of users' work with Web-based information management systems.

Web-based information management systems such as GeoWorlds [3] provide useful tools for doing so, and offer an environment to help users create task-oriented information spaces from raw collections. Fig. 1 illustrates the process of using GeoWorlds to organize an information space on "High-Speed Internet Coverage Areas in the United States." An initial collection obtained by merging string searches is analyzed in various ways (e.g., grouping by frequently occurring phrases, plotting location references on maps) to help identify and populate a topic hierarchy that organizes the collection. Four major types of functions are illustrated: information gathering, information analysis, information visualization, and information organization. Each type operates on a collection of documents and produces a collection. Using the *information gathering* functions, GeoWorlds users can extract relevant documents from sources such as Web search engines, Web directory services, on-line yellow pages, news video archive databases, etc. They can get help characterizing the resulting initial document collection using a rich set of *information analysis* functions such as noun-phrase extraction, document clustering, category comparisons, language translation, etc. *Information visualization* components applied to the analysis results help users make sense of those results and identify important parts of the document collections to assimilate into the users' information collections. *Information organization* tools then help identify or impose structure on the results. These organized information spaces can be maintained in persistent storage. Fully organizing a body of information is an iterative process on collections and sub-collections, which repeats until the information space meets its users' needs.

* Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Material Command, USAF, under agreements F30602-00-2-0610 and F30602-00-2-0576. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the DARPA, the Air Force Research Laboratory, or the U.S. Government.

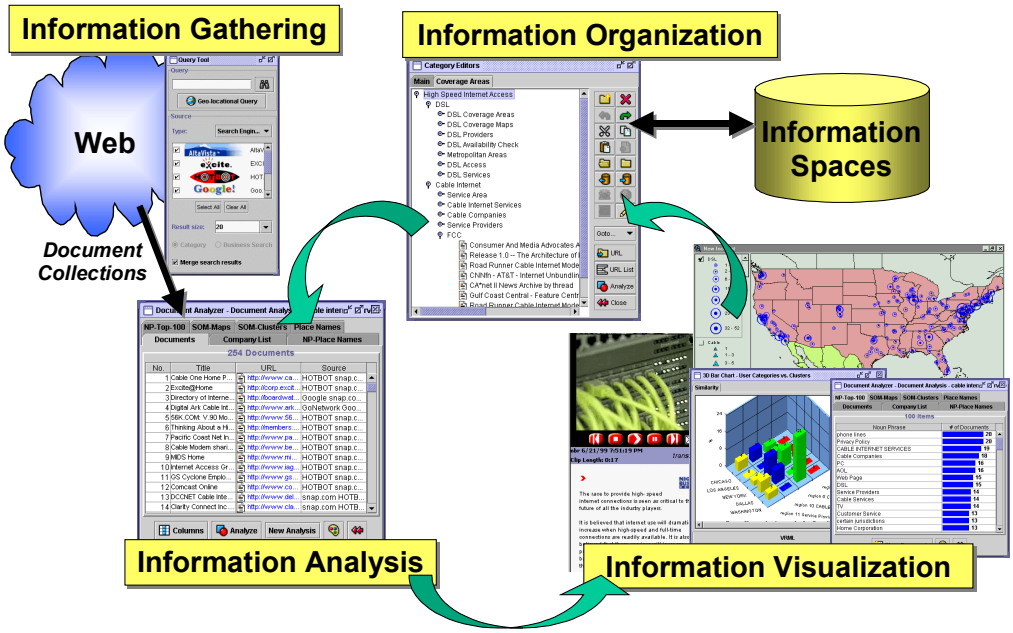


Fig. 1. Overview of the information management cycle for organizing the information space "High-Speed Internet Coverage Areas in US" by using GeoWorlds

Research on the "Semantic Web" [2] has focused on representing machine-processable semantics of Web resources. RDF (Resource Description Framework) [14], SHOE [6] and Ontobroker [4] are among the efforts that provide semantic models, languages, and inference mechanisms for representing and processing Web document semantics. These approaches, however, are limited to individual documents. To make full use of document semantics to augment information management systems like GeoWorlds, it is essential to provide separate models and inference mechanisms for semantics of *document collections*. These are the units of information processing for information management systems.

Based on the document collections' semantics (content types and organization structures), semantically interoperable (not just syntactically matched) analytic and visual services can be selected to perform context-sensitive information analyses. For example, although underlying data structures for all document collections might be the same, only document collections that have been clustered based on geographic location references can be plotted on a map.

Steps taken to organize document collections in an information space can be scripted by describing active semantic relations between document collections. For example, a document collection plotted on a map can be described as the result of plotting a document collection that contains "place-name-based document clusters." That, in turn, is the result of applying a geographic place-name extraction function to an unorganized document collection retrieved from the Web, which itself is the result of some set of actions on objects (e.g., string searches). We call these descriptions *active* because they describe how a document collection is generated from another. We consider them *semantic level* descriptions because they do not include any syntactic details such as the data types within the document collections or the specific analysis functions used.

By providing mechanisms to represent and process the active semantic relations between document collections, we can support *active document collection templates*. The advantage is that these can be composed by users in advance without getting bogged down in the syntactic details. Reasoning about the semantics of available tools and data, the system can dynamically instantiate templates on behalf of a user, and execute them to generate completed document collections based on local resources. Not just results, but the methods used to produce them, can be exchanged among users.

Thus, semantically-based service selection and active document collection templates are features that improve the efficiency of information management systems and make it easier for users to develop large scale, task-oriented information spaces. To enable these features, the following techniques have been developed:

- Explicit semantic representation: provide the semantic model and language to represent semantic information of document collections and services
- Active document collection template composition mechanism: provide the mechanisms and tools to compose information management templates by describing active semantic relations between document collections
- Reasoning mechanism on the semantics: match semantically interoperable services against a document collection or another service by comparing the semantic information of the components

The sections following next (from Section 2 to 4) focus in turn on these three enabling techniques. Section 5 describes the status of the current prototype implementation. Related work is reviewed in Section 6 and planned extensions are discussed in Section 7. The benefits and shortcomings are summarized in Section 8.

2 Explicit semantic representation

Sowa motivates that the same physical entity can be described by different forms to emphasize its content and physical structure [13]. Semantics of a document collection also can be described by using those two independent forms. The *content description* represents the contextual meaning of the collection (e.g., a document collection in which the documents are classified by the major noun phrases). The *structure description* characterizes the organization structure (e.g., a document collection organized in an acyclic graph structure). By dividing the semantic description about a document collection into these two types, the complexity of the semantic representation can be reduced and reasoning performance can be improved [15]. One more component in the document collection semantics is the *active relation* with other document collections, which describes the action of transforming a document collection from one semantics to another (e.g., an initial document collection that is a flat list of documents can be transformed to a hierarchically organized document collection by performing a document clustering service).

Domain-specific ontologies are used to discriminate and classify the document collection content types and organization structures, and service functionalities. Appendix A shows the ontology hierarchies of content types, organization structures, and service types in the current prototype system.

A schema model has been developed to represent the document collection semantics in terms of its content, organization structure, and active relations between other document collections. Fig. 2 illustrates this schema model as an ER (Entity-Relationship) diagram. Content, Organization Structure, and Service are the top-level ontology schemas that describe the top concepts in the ontology hierarchies of the semantic components. Since a service may require multiple inputs and generate multiple outputs, the input or output semantics is specified by a set (Collection Set) of document collections. Therefore, the active relations can be formed between the document collections in the input data set and the document collections in the output data set. *analysisResultOf* relation is explicitly added between the input and output document-collection sets.

A document collection can be a member of multiple I/O data sets and multiple relations with other document collections can be represented. For example, consider a document collection composed of Spanish documents and a document collection categorized based on place names cited in the document contents. These document collections are the results of an English-to-Spanish translation service and a place name extraction service performed on an initial document collection.

The next section explains how this semantic representation is used to provide a model for representing active document collection templates.

3 Active document collection templates composition mechanism

For some types of information analysis tasks, the information spaces resulting from a sequence of information management cycles might be time-dependent. For example, if the user performed the same information management steps several months apart, an information space on "High-Speed Internet Coverage Areas in US" would show more coverage areas on the map, a different coverage ratio between cable and DSL modems, and a different set of highly cited cable and DSL companies. Such time-sensitive information spaces need to be regenerated (refreshed) regularly to maintain the latest information.

Often, a sequence of analysis steps is repeated many times to organize an information space. For example, to get more detailed information about each high-speed internet coverage area, the same set of information management steps (retrieve relevant documents from the Web, extract company names, classify the companies based on

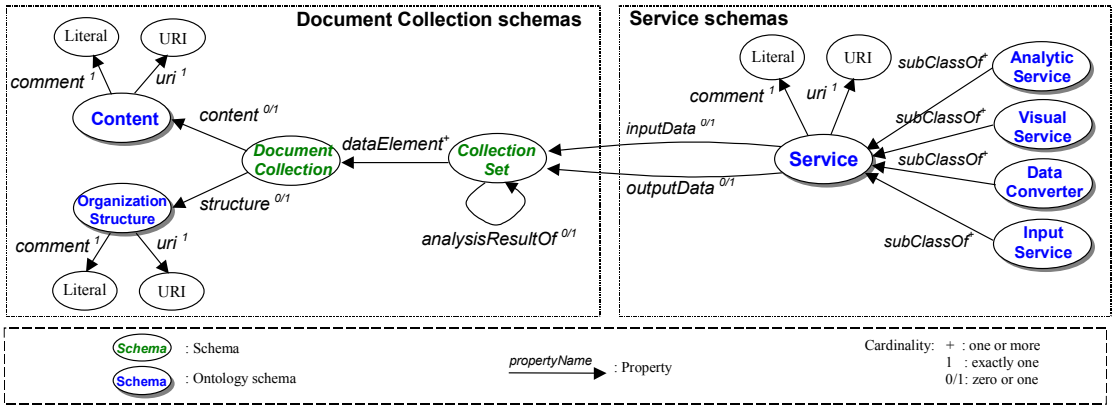


Fig. 2. Schema model for representing document collection semantics

connection types, etc.) need to be repeated for each area.

Without scripting the information management steps, these information space refreshing and detailing processes will be inefficient and may generate inconsistent results. To meet this need, we provide mechanisms to compose and run *active document collection templates*. These are semantic-level scripts of information management steps. In an active document collection template, semantic requirements for document collections and management functions for organizing an information space are described. Also, the active relations between document collections can be described by specifying the analysis functions that transform a document collection into another with different content and/or structure semantics.

Once an active document collection template is composed, it can be modified and used to quickly generate information spaces for other similar tasks. Many information management tasks are similar in their major analysis steps. For example, the steps for organizing the information space on "High-Speed Internet Coverage Areas in US" can be reused to generate similar document collections for other countries. Only the initial query and the place name set need to be modified to include the geographical information for a different country. Templates can also be modified to support other similar business analyses instead of creating new templates.

Active document collection templates enable users to exchange their information management scripts with other users. The templates can be dynamically instantiated based on the locally available resources, and will organize the task-oriented information spaces locally.

The following sub-sections explain the details of mechanisms to compose, instantiate and execute active document collection templates.

3.1 Model for representing active document collection templates

In our approach, an active document collection template can be represented as a graph in which document-collection sets and functional services are the vertices, and relationships between them are the edges. Fig. 3 illustrates a graphic representation of an active document collection template. By performing semantic measurement between the set of initial document collections (t_0) and the input data set semantics of available services, semantically interoperable functions can be matched.[1] When the user selects one of the matched functions (f_1), a new input set (t_1) is created that is composed of document collections (d_2 and d_3) that are required by the function. Then, the output document-collection set (t_2) is formed that points to the semantic description of the output document collections and contains the placeholders for the output objects. The same sequence of composition steps can be repeated on all the available document collections (d_1 ~ d_5) at the stage to add more functions to the template.

When the system records a template, it adds *analysisResultOf* relations between the input and output document-collection sets. This helps it efficiently generate and offer to users extended templates with additional services. For example, consider the following case. Say that the semantics of d_1 and d_2 are the same, i.e., $S(d_1)=S(d_2)=D$ (e.g. both are a flat document list).[2] Say that d_5 (a noun-phrase list) is one of the resulting data elements of the function f_1 (a noun-phrase extraction function), which has been applied to the input data set that contains d_2. Assume the system was considering extending the template of Fig. 3 by inserting a new function such as a document clusterer. Such a function might require a flat document list (such as D) and a noun-phrase list extracted from the document collection (such as the semantics of d_2 or d_5) for its input data. Without additional information in such cases, it would not be clear where to put it. However, the *analysisResultOf* relation enables the system to see that applying the document clusterer to d_2 is preferable because d_5 is the noun-phrase list which is generated by processing d_2, and d_2 (not d_1) should be provided along with d_5 to the document clusterer to get the correct result.

The graphic representation of an active document collection template, shown in Fig. 3 can be serialized (the current prototype generates XML data). This can be stored in a template repository or exchanged with other users.

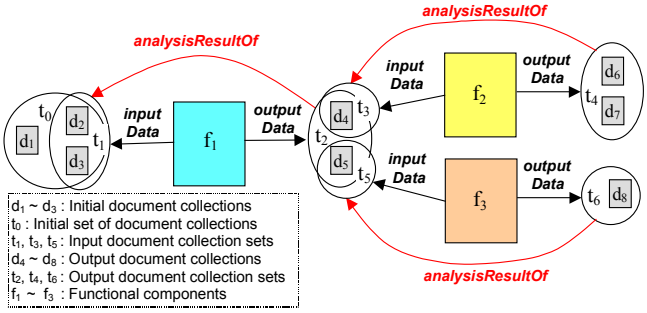


Fig. 3. Graphic representation of an active document collection template

3.2 Instantiation of an active document collection template

An active document collection template can be instantiated by allocating local resources to the template components. In a local system, semantic descriptions about the local resource instances are kept as metadata in a repository. Metadata about a resource instance also includes some syntactic information such as data types, job request entries, and I/O parameter ordering[3]. The semantic compatibility measurements that will be explained at Section 4.4 are performed to select semantically compatible local resources for each component in the template. The process of instantiating a template may require human interaction to

[1] . See Section 4.2 for details.

[2] S is the function that returns the semantic descriptions of a document collection.

[3] In the current prototype, Java class names are used to describe the internal data types of document collections. A job request entry is a directive to request for a service.

resolve multiple matches of resource instances and syntactic mismatches between instantiated components[4].

As the result of an instantiation, component proxies are created to act as clients to invoke the specific services that are selected to instantiate the template, and to receive the results of these service instances. Each *proxy* delegates a resource instance (a document collection or a service) and keeps information for accessing the local resource. Fig. 4 illustrates the proxies and relationships between them instantiated for a template. For each functional semantics, a service proxy is created and for each document-collection set, an ordered-list (ordered based on the I/O parameters of the corresponding service instance) of document collection proxies is created. A service proxy keeps precedence relationship between predecessor and successor proxies and pointer (proxyOf) to the corresponding semantic description in the template. A document collection proxy maintains a pointer to a document collection object.

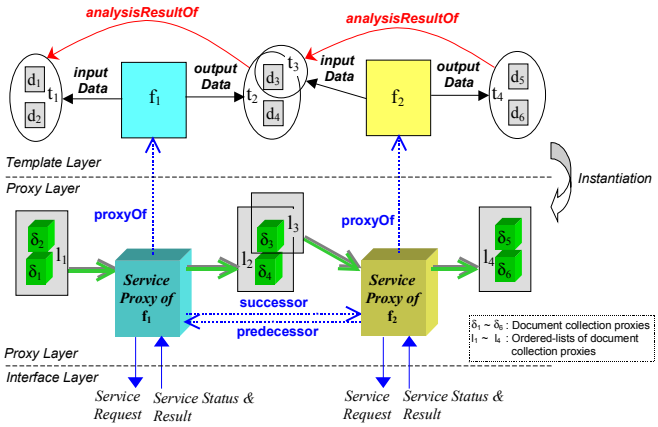


Fig. 4. Instantiation of a template by creating proxies

3.3 Template execution

The instantiated template can be executed by running the service proxies in a sequence governed by the precedence relations. An activated service proxy submits a service request to the system interface, monitors the job status, and receives the result. In the current prototype implementation, this service access mechanism is implemented based on the GeoWorlds' asynchronous service invocation architecture which is described in [15].

Multiple service proxies can be run in parallel and the proxies can be synchronized by using the precedence relations. When a proxy receives the result from its service instance, it updates the object pointer fields in its output document collection proxies to point to the result objects. Then, it invokes all the successor service proxies. A service proxy will not be activated unless it receives signals from all the predecessors. Usually the terminal services are visualization services that have no successor components.

[4] The current prototype resolves some syntactic mismatches automatically by finding and inserting syntactic converters. See Section 5 for details.

3.4 Template update

Active document collection templates for complex information management tasks can be developed *incrementally* by repeating the composition, instantiation and execution steps. After/while executing a template, the user can modify the template by adding new components or removing unnecessary components or replacing certain components with other semantically compatible ones (to specialize, generalize or alter the functionality). When a template component is modified, only the proxies that are affected are regenerated (usually from the modification point to the terminals). Therefore, when the template is re-executed, the unaffected proxies will not be rerun and all the intermediate data can be reused for the newly initiated services.

This incremental development of active document collection templates is especially effective when the information management task has to deal with large document collections and is composed of many analysis steps.

The next section describes how the semantics of document collections can be compared to select and combine semantically interoperable services. This is a key to composing an active document collection template.

4 Semantic reasoning mechanisms

A service can be applied to a document collection if the semantics of the document collection is equivalent to, or subsumed by, that service's input parameter semantics. For example, a category comparison service, which can compare two document collections that are categorized by a set of keywords, can also compare two document collections categorized by a set of place names because the place name-based categorization is a specialization of the keyword-based categorization.

The *semantic inclusion* relation is defined to compare such relationships between document collections or document-collection sets. This relation is the basis for matching services against document collections and measuring semantic interoperability and compatibility among services.

To explain the reasoning mechanism, the following formal representation is defined:

- Document collection semantics: $D = (c, s)$, where c is the content ontology and s is the structure ontology
- Semantics of a document-collection set: $T = \{ D_1, D_2, \ldots, D_n \}$, where $D_1=(c_1,s_1)$, $D_2=(c_2,s_2)$, …, $D_n=(c_n,s_n)$
- Functional semantics of a service: $F=(O,T_I,T_O)$, where O is the functional ontology, T_I is the semantics of the input data set, and T_O is the semantics of the output data set

4.1 Semantic inclusion relation

Semantic comparison between two document collections is done by comparing their content and structure semantics. Let $D_1=(c_1,s_1)$ and $D_2=(c_2,s_2)$ be the document collections to compare. If both the content and structure semantics of D_1 subsumes the content and structure semantics of D_2 (i.e., $c_1 \geq c_2$ and $s_1 \geq s_2$), we say that the document collection D_1 *semantically includes* the document collection D_2 and denote the relation by

$$D_1 \sqsupseteq D_2$$

The semantic inclusion relation is reflexive and transitive:

$$D \sqsupseteq D$$
$$D_1 \sqsupseteq D_2 \wedge D_2 \sqsupseteq D_3 \Rightarrow D_1 \sqsupseteq D_3$$

Semantic inclusion relations between document-collection sets are measured by comparing each pair of related document collections from both sets. Let $T = \{ D_1, D_2, \ldots, D_n \}$ and $T' = \{ D_1', D_2', \ldots, D_m' \}$ be the two document-collection sets to compare. Then,

$$T \sqsupseteq T' \text{ iff } (\forall D_x \in T)\,(\exists D_y \in T')\; D_x \sqsupseteq D_y$$

If there exists any element in T that semantically includes multiple document collection semantics in T' (i.e., the semantic inclusion relation between T and T' is not a function), the semantic inclusion relation is *map ambiguous*.

4.2 Semantically-based service selection

Given a set of document collections with semantics T_d, a service F with input parameter semantics T_I is selectable if T_I semantically includes T_d (i.e., $T_I \sqsupseteq T_d$). If the semantic inclusion relation between T_I and T_d is map ambiguous (i.e., there exist multiple mappings between the data sets), human interaction is required to resolve the ambiguity by explicitly specifying the mapping between T_I and T_d.

4.3 Semantic interoperability

Semantic interoperability characterizes the requirements for two services in an active document collection template to be composable, $F' \bullet F$ (output of F is the input of F'). Let the functional semantics of F be (O,T_I,T_O) and the functional semantics be F' be (O',T_I',T_O'). Then,

F' is *semantically interoperable* with F iff $T_I' \sqsupseteq T_O$

i.e., if the output document-collection set of F can be accepted by F', F' can be combined with F.

4.4 Semantic compatibility

Semantic compatibility in an active document collection template characterizes the requirements for one service component to replace another service component.

Let $F=(O,T_I,T_O)$ be the functional semantics of the original service and $F'=(O',T_I',T_O')$ be the functional semantics of the replacement service. The most conservative way to check the semantic compatibility is to compare the I/O data semantics of the two functions as follows:

F' is *semantically compatible* with F iff $T_I' \sqsupseteq T_I \wedge T_O \sqsupseteq T_O'$

This rule ensures that the replacement function accepts all input data semantics accepted by the original function, and only generate output data semantics generated by the original function.

However, by looking at the neighboring services (predecessors and successors) in the active document collection template we can derive a less restrictive *context-dependent semantic compatibility* rule. Suppose that T_x is the least upper bound (least general generalization) of the semantics that the predecessors of F can generate, and T_y is the greatest lower bound (most general specialization) of the semantics the successors of F can accept, and $F''=(O'',T_I'',T_O'')$ is the replacement service then

F'' is *context-dependent semantically compatible* with F
iff $T_I'' \sqsupseteq T_x \wedge T_y \sqsupseteq T_O''$

Notice with the above definitions, the following is true $T_I' \sqsupseteq T_I'' \sqsupseteq T_x$ and $T_y \sqsupseteq T_O'' \sqsupseteq T_O'$. This implies that context-dependent semantic compatibility is less restrictive than semantic compatibility, i.e., the set of services that accept T_I'' and generate T_O'' is a superset of the set of services that accept T_I' and generate T_O'.

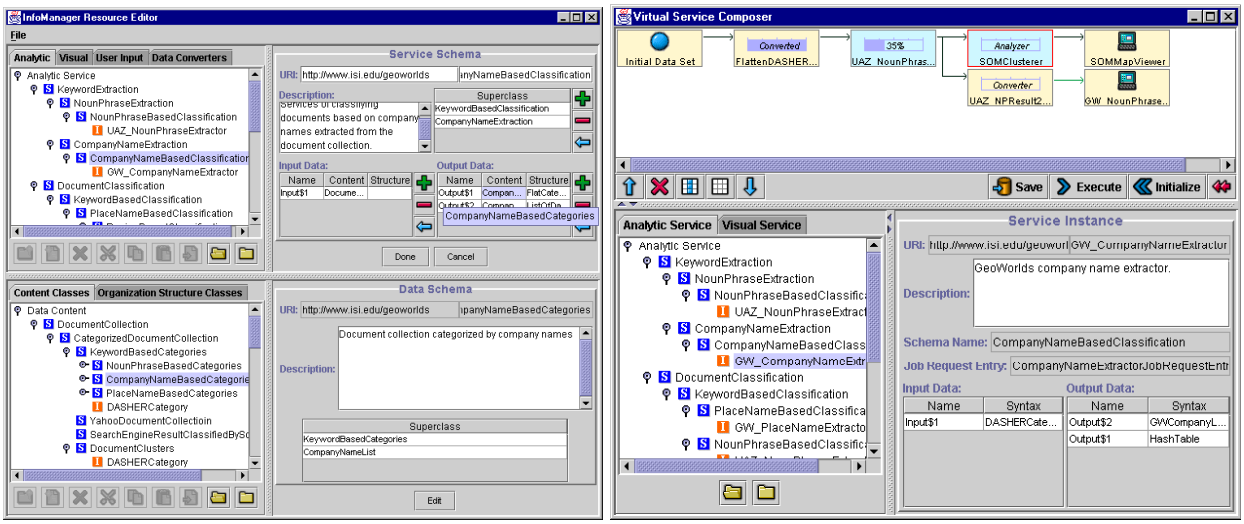
The act of replacing a service in a template with another compatible component can be classified into three cases, based on the relationship between the functional ontologies of the existing and the replacement services:

Generalization:
F' is a *generalization* of F if F' semantically compatible with F and $O' \geq O$.

Specialization:
F' is a *specialization* of F if F' semantically compatible with F and $O \geq O'$.

Alteration:
F' is an *alteration* of F if F' semantically compatible with F and there is no subsumption relations between O and O'. A service component with the functional semantics F can be replaced by a service component with function F' to perform an alternative functionality with preserving the data flow semantics in the template.

Using these relationships, we can build a system that helps in a number of ways. It can create and serialize templates or scripts. It can automatically select services and collections to instantiate a template, and can even make some repairs and substitutions to overcome some mismatches.



(a) (b)

Fig. 5. Prototype implementation - (a) metadata editor, (b) active document collection template composer

5 Prototype

A prototype system that embodies the capabilities outlined above has been implemented in Java. Persistence of resource descriptions and active document collection templates is provided via XML serialization. The major components in the prototype are the *inference engine*, *metadata editor*, and *active document collection template composer*.

The inference engine implements the mechanisms required to retrieve, compare, select and substitute among collections and among services, along the formal lines described in Section 4. It also provides consistency checking functions by which the system can ensure that only valid semantic descriptions can be added and appropriate resource instances can be classified.

When the inference engine is generating or testing candidates for use in a template, it can bridge across some semantic mismatches by adding additional components. For example, it can add a user input component if a data component is required by a service but is not available at the stage of editing the template[5]. Similarly, a structure converter (if available) can be inserted between components to convert between data formats when they are matched in content semantics but not matched in structure semantics.

The metadata editor is the system's GUI for registering and modifying semantic descriptions of document collections and services. By using this tool, local resource instances can be classified under the ontology hierarchies and their syntactic descriptions can be edited. Fig. 5 (a) shows a screen shot of the metadata editor. The upper part of the window is for editing service components (analytic, visual, user input, and data conversion services). The lower part is for editing content and structure semantics of document collections. In both cases, the left side displays the ontology hierarchy and the right side provides forms to edit schema property values[6].

The active document collection template composer provides GUI-based tools to help users set up analysis and structuring activities by composing, instantiating and executing templates. Fig. 5 (b) shows the template composer window. The upper part displays the currently included components and connections between them. The lower part shows the ontology hierarchies and a selected schema of available services that are composable with the ones already included in the template. This lets users see what options are available to them for adding steps to their analysis.

As described in Section 3, the composer creates proxies and proxy connections (a directed acyclic graph of control and data flow between proxies) when a template is instantiated. This enables the system to invoke, monitor and synchronize the services. Also, with the help of the inference engine, it automatically finds and inserts syntactic converters (e.g., data type converters) between syntactically mismatched proxies if appropriate converters are available. When a template is executed, each proxy node in the graph displays the status (progress bar and messages) of the service.

The example template shown in Fig. 5 (b) is a script to extract most frequently cited noun-phrases from a document collection, display the noun-phrases extracted, cluster the documents based on how many noun-phrases they have in common, and display a cluster map that shows the relation between adjacent clusters. This is a very frequently used script when we try to identify the major topics implicit in a

[5] Examples of the data that can be provided by user input services: a query string, a set of place names to be extracted from a document collection.

[6] In the example shown, the upper-left part currently shows the ontology hierarchy of analytic services. The upper-right part shows the schema for one of these services: "Company Name Based Classification." The lower part displays the outputs currently being edited in that service schema, in this case, the ontology hierarchy for the document collection contents, and the schema for "Company Name Based Category."

large number of documents retrieved from the Web, and want to understand the relationships between those topics. By using this template, we avoid the tedious job of repeatedly selecting and invoking the same sequence of services for every information management task. We can quickly extend the script to develop more complex analyses that require extracting noun-phrases and generating document clusters.

6 Related work

As networked resources such as the Web have become available, digital library collections increasingly need to be specified by description in the form of *criteria* for selecting resources or *tools* for resource discovery, rather than being defined by enumeration in physical document collections [9]. Our active document collection templates are an implementation mechanism for such descriptively specified, Web-based digital library collections. The semantic descriptions of document collections and the active relations between them are the criteria and tools to organize and manipulate document collections for information management tasks.

Sheth pointed out in [12] that the focus of information system interoperability research is changing from system/syntactic level to semantic level. As heterogeneity of digital data, operations and computations is increased, users' demands upon information systems are shifting from the mere data level to information and knowledge levels. Sheth's observation is consistent with our experience with GeoWorlds. As more information types are available from the Web and as we add more analytic and visual services to the system, users want to specify their information management requirements at a high level (instead of spending their energy figuring out which services can be applied to a certain type of document collection). Our semantically-based service selection and the active document collection template composition mechanisms were developed to satisfy this need.

A significant discussion of issues of semantic interoperability in large object systems appears in [7]. They argue that explicit representation and run-time manipulation of semantic information can reduce the time and effort to build large software systems composed of COTS and legacy components. Our semantic representation scheme and active document collection templates extend this into the domain of information management, by suggesting that semantics-based mechanisms enable efficient and rapid organization of large-scale, task-oriented information spaces.

The approach of using explicit semantic information for integrating heterogeneous information sources have been used by various information mediation projects such as SIMS [1], InfoSleuth™ [11], and GINF [10], which characterize the information sources by extracting the semantic information (domain ontologies and relationships between components) and expressing it using high-level representation languages. Semantics-based query processing and context-sensitive provision of analysis functions are closely related to our active document collection templates and the semantically-based service selection mechanism. However, their queries are for retrieving initial document collections and cannot specify the full information management cycles. Also, their analysis functions are mostly limited to information integration functions that are tightly bound with particular information sources.

There are similarities between our semantic inclusion relation and the semantic proximity measurement in the Semantics-Based Information Brokering system [8]. The semantic proximity represents semantic similarities between database objects based on their context. Similar to the semantic description of a document collection (or a document-collection set), the context of a database object is represented by referring to pre-existing ontologies. Also, the semantic similarity between database objects is measured by comparing corresponding ontologies.

7 Future work

The current template model has some limitations on its representational power (e.g., looping and dynamic behaviors between components cannot be specified). We are investigating information management cases and analysis functions that require more representation power. We plan to enhance the model based on that investigation's results.

By applying *semantic distance* measurements such as [5] to the semantic inclusion decision, some cases involving semantic ambiguity and multiple matching services can be resolved automatically. Instead of a binary decision, our inference engine can measure the semantic distance between document collections or between document-collection sets, i.e., measure how much a component semantically includes another one. Based on this distance measure, the system can select the most semantically close components.

USC ISI's TBASSCO (Template-Based Assurance of Semantic interoperability in Software COmposition) project addresses concerns about quality in adaptive composition of component-based software. We are developing semantically-based software gauges that measure the level of semantic interoperability between components. These help system engineers evaluate components' functional and data equivalence compatibility, find pertinent data conversion mappings, and predict performance (time, space, network) of a component architecture. The semantic measurement and the active document collection template composition mechanisms described in this paper are being generalized and extended for the TBASSCO semantic gauges.

We are currently developing an automatic template generator that returns active document collection templates based on users' high-level specification. It will generate all possible scripts that can achieve a user's information analysis goal. It will allow users to select and modify the template(s) to be instantiated for their tasks. We believe this will make their information management tasks go much easier and faster.

Persistence both of semantic description about document collections and services, and of active document collection

templates, is being implemented using RDF (Resource Description Framework) [14]. Our schema model for representing semantics maps directly to RDF's graph-based model, and our active document collection templates can be represented in a more structured way using RDF syntax. RDF will also enable remote schema referencing, an attractive alternative to copying entire domain-specific ontology descriptions when active document collection templates are exchanged between users.

8 Conclusion

Our goal is to fully utilize semantic information about collections retrieved from Web resources and to support large-scale, task-oriented information management systems. To this end, we have developed an initial set of semantic representation and processing mechanisms for document collections. We have complemented this with mechanisms to compose and run active document collection templates. These facilitate scripting complex information management steps at a semantic level. Our system is able to instantiate them dynamically and repeatedly based on locally available resources.

Our semantically-based component description and selection mechanism facilitates component-based information management software architectures. It provides an infrastructure in which data and service components can be added dynamically without requiring major changes to the system or to existing components. It improves software component reusability and enhances flexibility in choosing and using information management components. Complexity of component specification and reasoning is reduced by dividing the document collection semantics into two independent types: content and organization structure.

We believe the active document collection template composition and execution mechanisms serve to improve information management performance. By using the templates, expired document collections can be quickly refreshed. Development of new information analysis plans can be speeded up by reusing and modifying existing templates. Also, by exchanging relevant information management templates, users can collaborate with each other on organizing task-oriented information spaces.

The Semantic Web will make task-oriented information management systems more effective by providing infrastructure and tools to retrieve and manipulate more accurate and rich document semantics. Web information management systems that provide ways to manipulate *document collection semantics* and *active relations* between document collections, will make the Semantic Web much more usable and productive for its users.

Information and questions

For more information about GeoWorlds and TBASSCO projects:

http://www.isi.edu/geoworlds
http://www.isi.edu/tbassco

References

1. Yigal Arens, Craig A. Knoblock and Chun-Nan Hsu. Query Processing in the SIMS Information Mediator, Advanced Planning Technology, editor, Austin Tate, AAAI Press, Menlo Park, CA, 1996.
2. Tim Berners-Lee. Semantic Web Roadmap. World Wide Web Consortium (W3C), 1998. http://www.w3.org/DesignIssues/Semantic.html
3. Murilo Coutinho, Robert Neches, Alejandro Bugacov, Ke-Thia Yao, Vished Kumar, In-Young Ko, Ragy Eleish, and Sameer Abhinkar. GeoWorlds: A Geographically Based Information System for Situation Understanding and Management. In Proceedings of the First International Workshop on TeleGeoProcessing (TeleGeo '99), Lyon, France, May 1999.
4. Stefan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman et al. (eds.), Semantic Issues in Multimedia Systems, Proceedings of DS-8, pp. 351-369, Kluwer Academic Publisher, Boston, 1999.
5. Harry S. Delugach. An Exploration Into Semantic Distance. Lecture notes in artificial intelligence, No.754, pp. 119-124, Springer-Verlag, Berlin, 1993.
6. Jeff Heflin, James Hendler, and Sean Luke. SHOE: A Knowledge Representation Language for Internet Applications. Technical Report CS-TR-4078 (UMIACS TR-99-71). 1999.
7. Sandra Heiler, Renée J. Miller, and Vincent Ventrone. Using Metadata to Address Problems of Semantic Interoperability in Large Object Systems. First IEEE Metadata Conference, Silver Spring, Maryland, April, 1996.
8. Vipul Kashyap and Amit Sheth. Semantics-Based Information Brokering. In Proceedings of the Third International Conference on Information and Knowledge Management (CIKM), Gaithersburg, MD, November, 1994.
9. Carl Lagoze and David Fielding. Defining Collections in Distributed Digital Libraries. D-Lib Magazine, November 1998, ISSN 1082-9873. http://www.dlib.org/dlib/november98/lagoze/11lagoze.html
10. Sergey Melnik *et al.* Generic Interoperability Framework, Working Paper, Department of Computer Science, Stanford University. http://www-diglib.stanford.edu/diglib/ginf/WD/ginf-overview/
11. Marian Nodine, William Bohrer, Anne Hee Hiong Ngu. Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth™. 15th International Conference on Data Engineering, March, 1999, Sydney, Australia.
12. Amit P. Sheth. Changing Focus on Interoperability in

Information Systems: From System, Syntax, Structure to Semantics, *Interoperating Geographic Information Systems*. M. F. Goodchild, M. J. Egenhofer, R. Fegeas, and C. A. Kottman (eds.), Kluwer, 1998.

13. John F. Sowa, *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks Cole Publishing Co., Pacific Grove, CA, ©2000.
14. Resource Description Framework (RDF) Model and Syntax, World Wide World Consortium (W3C) Recommendation, February 22, 1999. http://www.w3.org/TR/REC-rdf-syntax/
15. Ke-Thia Yao, In-Young Ko, Ragy Eleish, and Robert Neches. Asynchronous Information Space Analysis Architecture Using Content and Structure Based Service Brokering. In Proceedings of Fifth ACM Conference on Digital Libraries (DL 2000), San Antonio, Texas, June 2000.

Appendix A: Ontology hierarchies of document collection semantics (content and structure types), and service types in the prototype system