WIT: A Tool for Integrating Web-Accessible Data

W. David Reese, Dennis Heimbigner, and Alexander L. Wolf

Department of Computer Science
University of Colorado
Boulder, CO 80309–0430 USA

University of Colorado
Department of Computer Science
Technical Report CU-CS-887-99 October 1999

ABSTRACT

The proliferation of World Wide Web accessible information sources has introduced the problem of integrating those sources into unified views of the data. Web-accessible data of interest can be customized and tied together in any number of ways and using a variety of possible techniques to result in structures known as datawebs. *Collaborators and others with a desire for unified views of multiple datawebs face the challenge of being able to integrate such webs without actually exercising any control or ownership over them. Such integration can be simplified when standard dataweb architectures and construction techniques are used. This paper discusses approaches to simplifying the integration problem and describes a Web Integration Tool (WIT) that has been constructed to aid in the automated integration of Web-accessible data.*

Keywords: World Wide Web, database, integration, federated databases, dataweb, semantic modeling, entity-relationship models, distributed software development

1 Introduction

People and organizations are putting ever increasing amounts and varieties of data on the World Wide Web (WWW). Typically, these data are hyperlinked together in an arbitrary graph topology to highlight relationships among the Web resources. The ability to link information together in this arbitrary manner has made it easy for entities to create and use customized webs of data or *datawebs*. At the same time, the Web's almost total connectivity has made it an ideal infrastructure for supporting collaborative activities. Unfortunately, the lack of structure in these arbitrary topologies has made it difficult to unify or otherwise integrate distinct webs of related data. As a result, collaboration among entities that build or use these webs is constrained. Further, even non-collaborating users of multiple sources of Web-accessible data are challenged in their ability to visualize a unified presentation of distinct datawebs. This leads us to ask if there are ways to take separate datawebs and integrate them easily.

Distributed software development is one domain that could particularly benefit from the ability to share, or more specifically, unify a view over, Web-based resources. It is increasingly common for software development organizations to operate from multiple geographical locations, employing large numbers of people with a need to share resources across multiple sites. Datawebs, because of their nearly total connectivity, are one means to share development information. Perhaps more significantly, as collaboration becomes more common among organizations that are temporarily allied (e.g., *virtual corporations*) to develop software, there are increasing needs for data sharing tools and techniques that support the ability to share only selected data in a specific, *integrated* context while maintaining some degree of autonomy over the data involved.

We have developed a general framework within which we can describe our process for integrating Web-accessible data. Specifically, we create, select, or otherwise identify Web resources to be integrated. Once identified, these resources are mapped into a semantic model capable of expressing relationships among them and relevant semantic information is superimposed onto the Web resources of interest. Essentially, this creates standardized datawebs with which to work and significantly simplifies the subsequent integration process. We have constructed WIT, the Web Integration Tool, to automate the integration of such datawebs. WIT empowers users with the ability to select multiple datawebs for content analysis and integration. An important aspect of our approach is that WIT performs all integration in a totally unobtrusive manner, leaving the original structure of the datawebs involved unaltered. The resulting integrated datawebs are hyperlinked structures that can easily be searched or navigated using an ordinary Web browser.

Certainly, a number of approaches to Web-based integration solutions exist. Internet search engines provide a quick and easy way to view Web-accessible data by topic from an incredibly large number of resources. Unfortunately, massive quantities of Web-accessible data are not indexed by even the most comprehensive search engines. Further, it is difficult to limit searches to specific datawebs and, if the desired data can be located, relationships between the data and other Web resources are not highlighted in any semantically explicit fashion. As a result, search engines often overwhelm users with extensive listings of unusable data instances.

Another approach could involve the construction and use of a centralized index of related Web-accessible data. These centralized indices typically consist of Web pages full of organized hyperlinks to other Web-accessible data. Many search engine companies, for example, operate *portals* where URLs on various subjects are listed. This approach comes closer to solving the challenges collaborators and others face when they need a unified view of multiple datawebs. Unfortunately, such indices can also grow overwhelmingly large, making their maintenance and navigation/search more of a challenge. Moreover, the highlighting of semantically explicit relationships among Web-accessible data is still lacking. Perhaps most importantly, these centralized indexes are not designed to be quickly merged or unified with other, possibly unrelated or overlapping centralized indexes created and owned by others.

It would seem there are two key aspects to the problem at hand and to any approach toward a more optimal solution. First, additional structure needs to be imposed on the resources that comprise the Web-accessible data to be used and shared within specific contexts. Specifically, supplemental structure that is capable of both classifying data and semantically highlighting relationships among Web-based data would be useful. Such structure would permit a uniform and logical way to access, integrate, and store such data in order to make it more useful for collaborating entities such as software developers or anyone needing a unified view of multiple datawebs. Second, the resulting datawebs need to be integrated. With supplemental dataweb structure in place, the integration aspect can be more easily addressed.

Creating the ability to retrieve and integrate information from multiple resources has been the goal of numerous research projects in recent decades. The database community has attempted, with some success, to grapple with the underlying issues of schema integration [4, 26], federated and other multi-database architectures and systems [8, 16, 25, 28], and the particularly challenging issue of semantic heterogeneity commonly found when dealing with multiple sources of data [3, 11, 14]. Simultaneously, the information retrieval and network communities have faced many of the same challenges with Web-based information resources [12, 13, 18]. Further, the advent of the World Wide Web and the massive amounts of easily published information it makes readily accessible, has accelerated research efforts related to the integration and retrieval of network-accessible data in particular [19]. Unfortunately, attempts at solving the various aspects of integrating and retrieving data from multiple sources have led to complex, heavyweight solutions involving special wrappers or translators, sophisticated supplemental knowledge bases, complex mapping functions, and/or the involvement of global database administrators.

The Web, however, enables the embedding of semantic information within its structures of related data by using RDF, XML, or other schemes [6, 7, 9, 21, 30]. By focusing on adding semantic structure to groups of related Web data, the challenges of integrating distinct sources of data (and subsequently retrieving data of interest) become less daunting. In addition, by leveraging the standardization of the Web and markup languages such as HTML, it becomes possible to avoid a number of architectural-boundary related problems all together. By focusing on lightweight solutions that fully utilize embedded semantic information, we believe WIT makes an important original contribution to the field of integrating Web-accessible data.

2 The Integration Process

At a more abstract level, integration of data is a two-phase process. First, the structures to be integrated must be created if they do not already exist. Then, a unified view of all structures to be integrated can be created. In order to simplify the task of integrating Web-accessible data, it makes sense to focus on the structure of the information we desire to integrate. Earlier, the notion of *datawebs*, or customized webs of linked data, was mentioned as one method of organizing such data. If datawebs comprise the building blocks of our integrated structures, it follows that the use of uniform dataweb design and construction techniques could not only simplify their subsequent integration, but also enable more generalized approaches to integration tools and techniques.

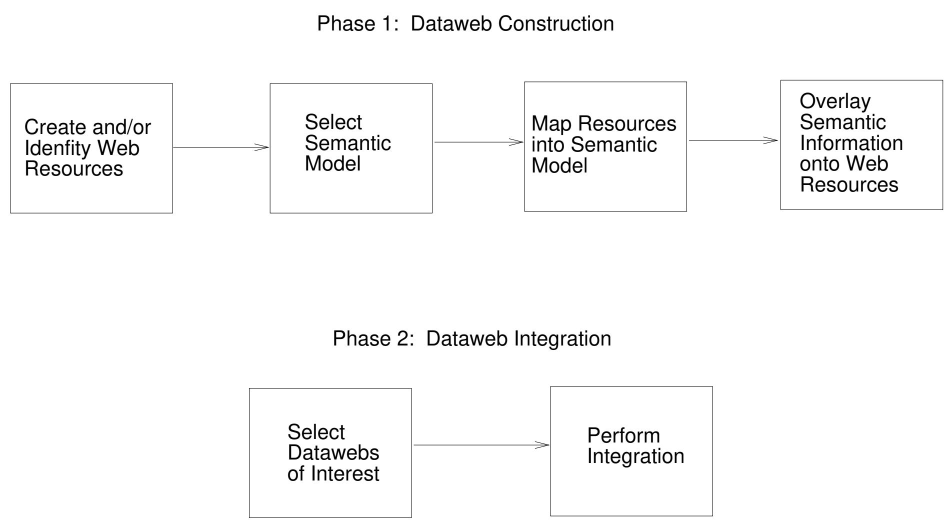

Figure 1: The 2-Phase Integration Process.

2.1 Dataweb Creation

Prior to any actual integration, the Web-accessible data of interest needs to be identified and/or created. Once it exists in Web-accessible form, a semantically rich supplemental structure can be added to tie the data of interest together into one or more datawebs. The specific architectural technique used to construct the dataweb is not critical. However, the uniform use of a technique for constructing all datawebs that could potentially be integrated at some later point in time can greatly simplify the integration process. For example, if the Web-accessible data we wish to integrate is read only, one uniform starting point might be to supplement the data of interest with a surrogate for the read only data—namely, another newly created Web page containing pointers, or hyperlinks, to the data of interest and possibly descriptions of how the data pointed to relates to other data of interest. As a concrete illustration, consider one or more software engineering organizations that desire to build a dataweb of artifacts (e.g., source code, test sets, documentation, etc.) related to a particular project. Once the artifacts were identified and made Web-accessible, supplemental Web pages could be created with hyperlinks to each artifact instance. Each supplemental Web page might also contain additional descriptive information about how the artifact it links to relates to other artifacts, locations (URLs) of related artifacts, descriptions of (or possibly URLs to other Web documents describing) the specific nature of the relationship or other attributes of the artifact.

Obviously, a wide spectrum of dataweb organizing techniques are possible. Considerable latitude also exists in selecting the *granularity* of the Web-accessible data to be included in the datawebs. One could choose to construct datawebs where the data of interest were specific lines of text within specific Web pages. One might also choose the Web page as the level of data granularity within a dataweb. Of course the granularity could also be made quite coarse by using host names and paths to make entire directories of information the lowest level of granularity. Once again, the level of granularity used is not important, but use of an agreed upon granularity level within datawebs can simplify their subsequent integration.

2.2 Dataweb Integration

Once datawebs exist and have been identified as targets for incorporation into larger, integrated datawebs, integration can take place. Our focus during the integration phase is on leveraging both the standardization and the semantic richness of the supplemental structure used to construct each dataweb. If the integration process closely parallels the architecture of the underlying datawebs involved, a number of advantages will result. First, the resulting integrated dataweb will structurally retain much of the same look and feel of the underlying datawebs—a useful feature for dataweb users. More importantly, the resulting dataweb will itself retain properties useful for further integration with other datawebs.

3 Dataweb Architecture

As the previous section indicates, datawebs can be constructed in any number of ways. However, use of a common architecture can be helpful in designing lightweight solutions to the problem of integration and retrieval of information from multiple Web-based resources. A couple of generalized goals should be kept in mind when choosing the architectural technique to tie Web-accessible data together into a dataweb. First, the addition of semantic information about the Web-based data and how each piece relates to others is desirable. This kind of information increases the utility of the dataweb by making its search and navigation easier. It also assists in the organization of the dataweb itself (during the dataweb construction phase) by making the various data types and relationships among them more explicit. Second, since we cannot guarantee that all Web-accessible artifacts of interest will be under the direct control or ownership of their users, the use of a surrogate such as a supplemental Web page containing a hyperlink to the artifact, buys us significant flexibility. We can manipulate or alter the surrogates we create. It also becomes possible to include the same artifact of interest in multiple distinct datawebs, with each dataweb referring to the same artifact in a different context.

3.1 Semantic Modeling

For the organization of Web-based artifacts, semantic models can be extremely useful. As Web-based artifacts are created or identified for inclusion in datawebs, it becomes important to be able to explicitly organize the artifacts along the lines of some sort of conceptual model. This organization is usually accomplished by the dataweb creator's reflection on how, at some level of abstraction, the various artifacts are categorized and/or related to one another. Semantic models facilitate the construction of a *schema*, or overall conceptual model, for artifacts that will comprise a dataweb.

A number of semantic models have arisen since the mid 1970s for use by database designers [17]. Among the more widely used models are the Functional Data Model (FDM) [27], the Semantic Data Model (SDM) [15], and the Entity-Relationship (E-R) [10] semantic data model. While any of these could be used as a basis for organizing dataweb schemata, the E-R model has the advantage of being the most well known. It also seems sufficiently rich for use as a basis for conceptually organizing most types of datawebs. Indeed, we know of at least one dataweb management system that uses this model [9]. For these reasons, we will focus on using the E-R model to describe the schemata of the datawebs in this paper.

In a nutshell, the E-R model uses the term *entity* to describe a particular kind or type of artifact. Entity types are depicted in E-R diagrams as rectangular boxes. The model uses the term *relationship* to describe how entity types interact with, or relate to, one another. E-R diagrams depict relationship types as diamond shaped figures that connect, via lines, the entity types that

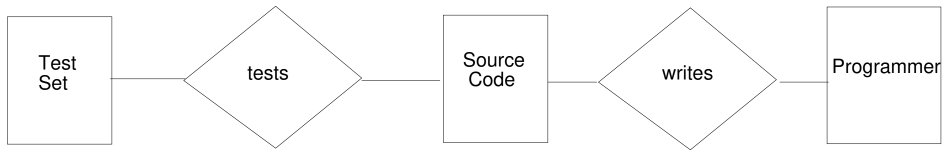

Figure 2: A Simple Entity-Relationship (E-R) Diagram.

can participate in the relationship. E-R diagrams can vary in size and complexity with the level of abstraction used in the conceptual model they strive to represent. Traditional E-R diagrams depict only the *types* of entities or relationships involved in a model and *not* actual instances of artifacts or relationships between them. A simple E-R diagram can be seen in Figure 2.

3.2 Dataweb Structure

Once the selection of a semantic model has been made, Web-based artifacts of interest can be organized along the lines of the resulting schema. The challenge then becomes to efficiently supplement the Web-accessible data with semantically rich Web-based components (additional Web pages, links, anchors, etc.) that will meaningfully connect the various artifacts of interest together into a dataweb. Again, since it is possible that we do not own or control the Web-based artifacts of interest, the structure of the dataweb should not depend on our ability to alter, or write to, the artifacts being organized. Pointing to the artifacts, via hyperlinks, would seem to be an obvious way to overcome this read only constraint. It follows that a logical starting point for structuring the dataweb would be to use the hyperlinks, or the Web pages containing them, as surrogates for the artifacts of interest. These surrogates can then be organized along the lines of the semantic model to conform to the desired schema for the dataweb to be constructed.

One problem we immediately run into by using surrogates is efficiency. The Web-based components comprising our surrogates need to be created, stored, and possibly maintained on an ongoing basis. Further, they add another level of indirection to our datawebs. As we search or navigate the datawebs, the surrogates will always represent yet another link, the next-to-last, to be traversed along the path to each artifact of interest. In addition, we may need to add more than one level of indirection as we organize artifacts into a dataweb conforming to our semantic model. After all, a single Web page filled with hyperlinks to artifacts of interest would closely resemble, and have many of the same shortfalls as, a Web page returned by a search engine! Indeed, for maximum utility and ease of use, several levels of semantically rich, organizing, supplemental Web structure could be needed. Thus, any system we use to supplement Web-accessible data of interest with a semantically-enhanced dataweb structure will likely involve some sort of trade off between utility and overhead. At a very abstract level, our end product should be a dataweb similar to that shown in Figure 3.

4 The Web Integration Tool (WIT)

In this paper we hypothesize that datawebs constructed following a uniform architecture simplify the design and development of lightweight, automated integration tools and techniques. We built the Web Integration Tool (WIT) to validate this hypothesis.

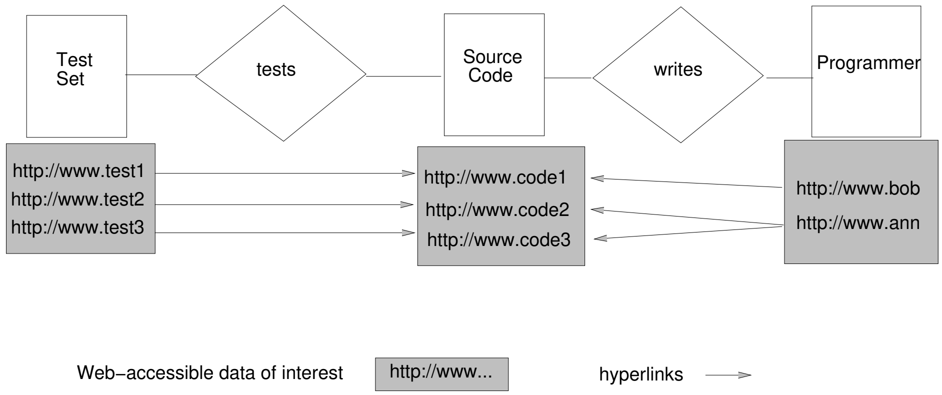

Figure 3: Schema and Associated Dataweb.

WIT, a tool for integrating datawebs, currently exists as a prototype system at the University of Colorado's Software Engineering Research Laboratory. The goal of the tool is to simplify, and to the extent possible, automate the process of selectively integrating multiple datawebs into a unified dataweb. The resulting structure can then be accessed and efficiently navigated by collaborators and others using Web resources (and relationships among them) from the combined datawebs.

4.1 WIT Capabilities

The WIT prototype enables users to rapidly build a unified dataweb by simply supplying the URLs of datawebs to be integrated. The integrated datawebs constructed by WIT closely follow the data model of the datawebs used as inputs to the integration process. As a result, WIT-constructed datawebs can easily be used as input for further integration with other datawebs, allowing arbitrary numbers of datawebs to be integrated into a single unified dataweb. Thus, federations of any size can use WIT to construct unified views of their data.

The WIT integration process does not alter the original datawebs used as input in any way. Instead, WIT creates a new, integrated dataweb based on the contents of the individual datawebs to be integrated, and writes it to the desired location (i.e., the WIT target host machine). However, WIT *can* be made to alter existing datawebs when desired. For example, users of WIT can alternatively elect to *merge* one dataweb into another. The merging operation differs from integration in that one of the datawebs is *supplemented* with information contained in one or more other datawebs. Essentially, existing directories of (hyperlinks to) relations and entities are supplemented as needed to incorporate information from other datawebs. Obviously the merging process does alter the dataweb into which the other(s) are being merged. However, since the current WIT prototype operates on *pairs* of URLs, the merging capability enables odd numbers of datawebs to be integrated.

At the user's option, WIT also allows users to gather statistical information about the datawebs being integrated. WIT can provide users with instance counts for each entity and relationship type contained in the datawebs. This information can be useful for determining not only detailed dataweb inventories, but also how much they have changed since such statistics were last gathered. Finally, users can also elect to obtain a detailed transcript, or log, of the integration activities WIT performs (e.g., the scanning and copying/merging of remote directories and files from the URLs of datawebs involved).

4.2 Architecture and Implementation

Because WIT operates on a standardized dataweb architecture, a very lightweight implementation of this integration tool is possible. WIT utilizes Java servlets, accessible from a browser, to compare contents of remote datawebs and create the necessary files and links needed to provide a unified view of the datawebs of interest. Network connectivity and an HTTP server, supplemented with a servlet engine, are required by sites using WIT. Users can access WIT most easily by using a network-connected browser. WIT operates in a totally unobtrusive manner, disturbing neither the Web-accessible data nor the organizing structure of the datawebs being combined.

4.2.1 WIT Datawebs

An early decision required in the design of WIT related to the selection of a (standardized) data model to use for its dataweb architecture. As discussed earlier, any of a wide variety of possible architectures could be used so long as a single standard was adhered to. Adoption of an existing dataweb management system would serve two purposes. First, it would allow our efforts to remain focused solely on the integration aspects of datawebs. The multitude of possible designs and implementations would be immediately narrowed to a single variant. Secondly, the design and construction of an integration tool for real, pre-existing datawebs would lend credibility and a degree of validation to any integration tool successfully built upon that dataweb model. Our review of existing dataweb construction and manipulation tools resulted in the discovery of a novel system called Labyrinth.

Labyrinth [9], a research system developed at Politecnico di Milano, provides one solution to the lack of structure in related Web-based resources. Without altering original Web resources, Labyrinth supplements each resource with a related HTML *entity shadow file*, a Web page containing links not only to the original Web resource, but to *relationship* shadow files (Web pages) as well. These links enable users of Web-based resources to make relationships among such resources explicit. In turn, it becomes possible to navigate these datawebs much the same as one would traverse an Entity-Relationship diagram by traversing the lines connecting entities to relationships.

Related Web resources are further supplemented under Labyrinth with *directories* (Web pages) containing links to all instances of each shadow file type. At the highest level of Labyrinth structure, a *schema*, in the form of a directory of directories, identifies links to each type of entity and relationship directory. All Labyrinth-specific supplemental dataweb information consists of HTML pages containing hyperlinks to either original Web resources or other Labyrinth HTML pages. Further, the original Web resources remain unaltered and need not be under the control of the organization constructing the Labyrinth dataweb. Users access Labyrinth datawebs just as they would any other Web data—most easily via a browser.

In Figure 4, the logical structure of a Labyrinth-style dataweb can be seen. Arrows in the figure represent hyperlinks and the boxes represent Web pages. The Web page in the left most column represents the top-level schema of the dataweb and contains hyperlinks to each of the entity directories (in the second column) and relation directories (shown in the third column). Labyrinth distinguishes hosts containing this top-level schema as *home* machines. Each of the directories, in turn contain hyperlinks to shadow files (shown in the fourth column). Entity shadow files use hyperlinks to point to the shadow files of relations they participate in as well as to the instance of the entity they represent (instance entities are shown in column 5, the right most column). Conversely, relation shadow files contain hyperlinks to the shadow files representing the entities that participate in the relationship. For example, a *tests* relationship shadow file would contain hyperlinks to a *Test Set* shadow file and to a *Source Code* shadow file. Relationship shadow files

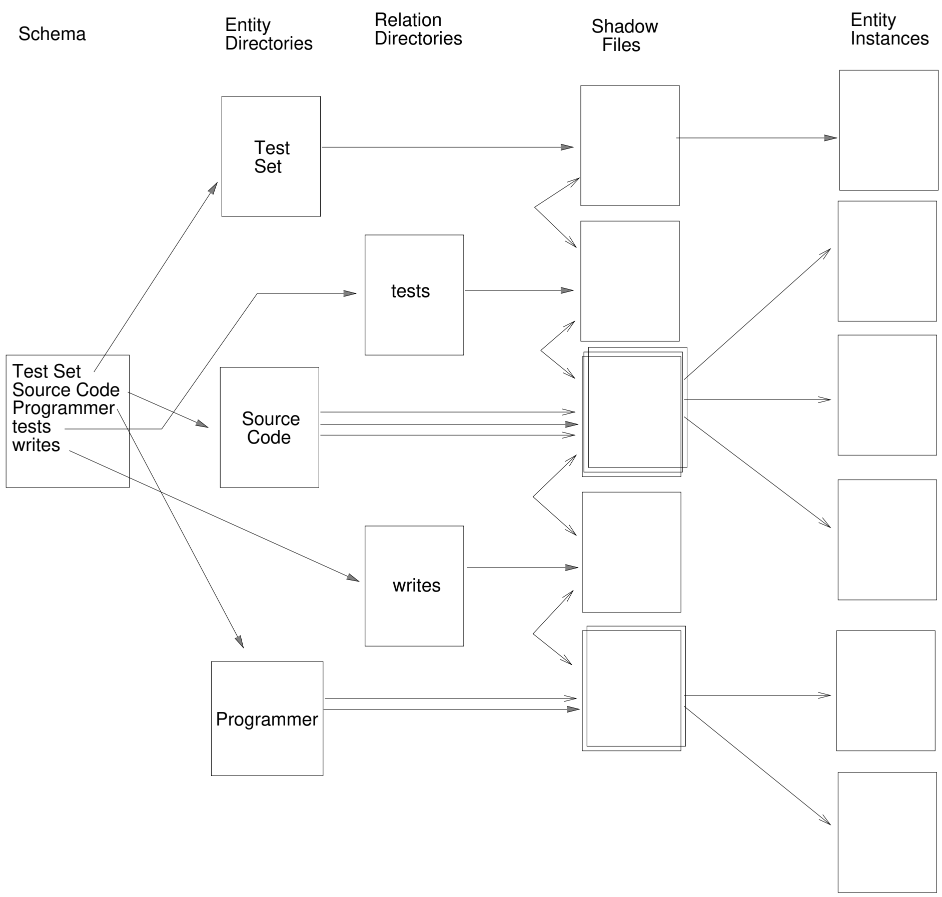

Figure 4: An Example of Labyrinth's Logical Structure.

provide a means of explicitly showing how Web-accessible data relates to other Web-accessible data—a powerful semantic enhancement to the organization of such data. The files shown in the far right of Figure 4 represent the original Web data—data that was present prior to adding any supplemental structure that effectively incorporated it into a dataweb. This original data can reside anywhere on the Web and need not be owned or under the control of the dataweb's constructor. For the remainder of this paper we will use the term *ER-dataweb* to refer to datawebs built using a Labyrinth-style architecture.

4.2.2 Dataweb Manipulation

An additional advantage to selecting ER-dataweb architecture for the WIT system was the Labyrinth system's ability to quickly build, search, and update ER-datawebs. Labyrinth provides a straightforward, browser-based interface for ER-dataweb updates like adding or removing instances of entities and the relationships they participate in. This utility stems largely from a series of template and other supporting files (which, for clarity, are *not* shown in Figure 4) that further supplement the ER-dataweb at the entity and relationship directory and shadow file level. WIT currently integrates most of these template and supporting files during its ER-dataweb integration processing. This allows WIT-produced integrated ER-datawebs to remain compatible with the Labyrinth system—enabling subsequent manipulation of the integrated ER-datawebs by Labyrinth software.

4.2.3 WIT Implementation

The WIT system is comprised of a series of Java servlets that interact across the network space where the ER-datawebs to be integrated reside. Java servlets provide a straightforward means of analyzing and navigating the internal structures of ER-datawebs residing on servers distributed across a network. WIT's servlets are designed to be collocated on the same hosts as Labyrinth for most usage scenarios. Neither WIT nor Labyrinth needs to be located on the Web hosts where the underlying ER-dataweb content resides.

WIT is capable of retrieving and, in some instances writing, remote and local files and directory listings. This capability, along with knowledge of the highly standardized architecture of ER-datawebs, enables the servlet to methodically scan each ER-dataweb's content and supporting structures to create a structured superset of all components encountered. This superset becomes the integrated ER-dataweb and is written to the desired host machine.

WIT servlets can be installed on any host where integrated ER-datawebs are to be stored. This host is also referred to as the *target* host. WIT servlets can also be added to any Labyrinth home machine. Doing so enables the ER-dataweb originating there to participate in a federation of WIT ER-datawebs that can be integrated with each other. By co-locating WIT with home machines, the federation of WIT servlets can easily access the Labyrinth directories of ER-datawebs participating in the federation.

The servlets use the user-provided URLs of ER-datawebs to be integrated as the starting point for integration processing. Armed with the locations of the ER-datawebs, WIT opens network connections between the target host called by the user (where the integrated ER-dataweb will eventually be written) and the host machine(s) of the ER-datawebs to be integrated. The WIT servlets then cooperate to exchange information about the structure and content of the ER-datawebs involved. Names of entity and relationship types, for example, can be surmised by reviewing the relevant Labyrinth entity and relationship directory listings. Identical entity and relationship directories can then be constructed in memory on the target host.

After combining entity and relationship types from the participating ER-datawebs, duplicates are removed and the integrated entity and relationship directories are written to the target host. A WIT servlet on the target host then proceeds down the directory structure, to directory listings of entity or relationship *instances*, performing a similar information exchange with the WIT servlets residing on the hosts of the ER-datawebs to be integrated.

In addition to building an integrated structure of directories representing the union of the ER-datawebs to be integrated, the underlying files, including the Labyrinth template files, need to be copied to the target host. Again, based on directory listing information shared among the WIT servlets, network connections are established between the distributed servlets to enable copying files to the target host's memory. There, a number of the template and other Labyrinth support files are parsed and rebuilt as needed to correctly support future manipulation of the integrated ER-dataweb being built. Once the files are appropriately modified, they too are written to the file system on the target host.

During the integration process, WIT retains information about quantities of entity and relationship types for each host involved. Instance counts, by type, are also made and retained for each host involved. The result is the ability to offer users the option of reviewing statistical information about the diversity and inventory of entities and relationships comprising the individual and integrated ER-datawebs. Such statistics can be valuable for estimating the level of activity or change within ER-datawebs and WIT makes them easy to gather. Similarly, an audit log can be generated at the user's option to show the activity of the WIT servlets. Users can use the log to review the integration steps taken by the servlets including reading, copying, creating, and merging of directories and files to and from the host machines involved.

As indicated above, a significant number of network connections are used during WIT's integration operations. This is partially a result of the limitations of the HyperText Transport Protocol (HTTP) and its one-request, one-response paradigm. By establishing network connections as needed, remote WIT servlets can be queried about directory contents and subsequently queried about the specifics of files or subdirectories all from within the same servlet method.

4.3 The WIT User Interface

Users can access the WIT system via an HTML form. The forms can be customized if desired and placed on each host that can potentially serve as a target host. Users can access WIT's input forms from any location with network connectivity, enabling users to remotely create integrated ER-datawebs. Because the initial WIT prototype was designed to integrate no more that two ER-datawebs at a time, the forms provide fields for users to enter the URLs of two ER-datawebs to be integrated. Users can also select among options for generation of ER-dataweb statistics or audit logs. A sample WIT main menu is shown in Figure 5.

The output of the integration process is also displayed in the form of a top-level Labyrinth style schema. The schema contains a hyperlinked listing of all entity and relationship types in the integrated ER-dataweb and is nearly identical to the Labyrinth system's ER-dataweb representation. When integration is requested using WIT's main menu, the integrated ER-dataweb's representation is written to a Web page on the target host and then replaces WIT's main menu in the browser. In addition, if the option was selected, users will also see statistics related to either of the input ER-datawebs or the integrated ER-dataweb in the same browser-displayed page as well.

Figure 5: WIT's Main Menu.

4.4 Post-integration Operations

The ER-datawebs generated by WIT have a number of interesting properties. As mentioned before, they can be further integrated into even larger integrated ER-datawebs. They can be navigated, searched, added to, or deleted from in largely the same way as the Labyrinth-style ER-datawebs can. (In some cases, minor modifications of Labyrinth support files may be required for subsequent addition and deletion of entity or relationship instances.)

Once constructed, WIT ER-datawebs remain dynamic and under the influence of the *owners* of the Web-accessible data of interest. Entity instances referenced by URLs within shadow files can continue to be changed by those with write permissions at those URLs. As a result, a significant degree of autonomy is retained by the participants in WIT's federated ER-dataweb schemes. Further, because WIT tries to minimize ER-dataweb support structure that is actually copied to the target host, it becomes possible for owners of ER-datawebs involved in integration to retain revision control of significant portions of WIT-constructed integrated ER-datawebs. This occurs in cases when a particular ER-dataweb contains a unique type of entity or relationship. Entity (or relationship) types that appear in only a single ER-dataweb do not need their respective type directories to be unified during integration. Instead, WIT creates a hyperlink from the target host's top-level schema to the directory of interest at the host machine containing the unique entity or relationship type. Here again, this increase in autonomy is a double-edged sword. The integrated ER-dataweb can be dynamically updated after integration time. However, the dynamic updates may not be desired by some users of integrated ER-datawebs. For example, some users may be interested in historical snapshots of an integrated ER-dataweb for some point in time. Others may be concerned about the possibility of dead links developing within the integrated ER-dataweb over time. Of course, re-integrating the ER-datawebs with the WIT system will always result in the latest, most up-to-date ER-dataweb structure at all levels.

4.5 Usage Scenarios and Potential Applications

The field of distributed software development is a particularly good target for WIT and other Web-based integration tools. Consider a large defense contractor that, due to the unpredictable nature of government contract awards, chooses to subcontract substantial portions of its software development (or maintenance) work. On an as-needed basis, subcontractors are selected for the development or maintenance of specific subsystems. The contractor needs a way to unify a view of the artifacts related to the various subsystems in order to control budgets, testing, and other aspects of the integration process. In addition, the integrated view should allow (read only) access to artifacts of interest. Finally, because of the transient nature of the subcontractors, the insertion or deletion of subcontractors and their artifacts needs to be accomplished with minimal disruption to others participating in the federation.

WIT would prove useful under such circumstances. Sub-contractors could organize hyperlinks to their artifacts using a standardized ER-dataweb structure desired by the prime contractor. With all subcontractors organizing ER-datawebs in the same manner, WIT could be used to provide a unified view (and access) to all subcomponents under development by the federation members for purposes such as integration testing. The prime contractor or entity responsible for integration testing could use WIT to easily select and integrate relevant ER-datawebs to generate a customized version of a federated ER-dataweb of interest. The hyperlinks in the WIT-generated ER-dataweb could then be followed to rapidly locate and access like entities (and relationships they participate in) from any number of ER-datawebs within the federation. One could, for example, locate relevant subsystems, their test suites, and related budget or staff-hour data for each. Further, so long as members of the federation adhere to original ER-dataweb organizing schemes when updating artifacts, the WIT-generated integrated ER-datawebs will remain dynamic and up-to-date. However, the departure or arrival of new federation members could necessitate the reuse of WIT to generate an integrated ER-dataweb reflecting such additions or deletions. During the integration process, WIT does not automatically identify *new* relationships between entities from distinct ER-datawebs that are integrated. In cases where logical relationships are created as a result of combining diverse ER-datawebs, new relationship instances (and hyperlinks to the entities that participate in them) would not be created by WIT.

Because ER-datawebs are navigated by following links between related entities and the relationships they participate in, they are best suited for organizing unique entities such as software development artifacts. Such artifacts are often essential to keep up-to-date yet need to be continuously shared by sometimes geographically distributed groups for diverse reasons. The ability to integrate multiple ER-datawebs into a logically unified structure can significantly reduce the time and complexity involved in searching for and accessing such data. Moreover, such unified views can serve to highlight otherwise difficult to spot relationships, trends, or problems that exist in the data being integrated.

Recent demonstrations of WIT have shown its applicability in domains ranging from software engineering to emergency preparedness data used by cooperating local governments within a region to prepare for and recover from natural and man-made disasters. One could further imagine WIT's applicability to distributed groups of architectural or electrical engineers who need to collaborate in order to produce complex designs involving the integration of many parts and related information. These groups can use WIT to easily share all aspects of their work as needed without surrendering any autonomy over the artifacts involved. With its light-weight infrastructure, WIT is an extremely flexible tool that can be configured quickly for projects of limited duration.

5 Related Work

Obviously the phenomenal popularity and growth of the Web have spurred an ever-increasing number of efforts to resolve integration of, and retrieval from, distinct data resources. Bouguettaya et al. provide an excellent overview of the challenges involved and researchers' attempts to overcome them [5]. The Labyrinth organizational scheme is only one of many possible ways to add supplemental structure to (read-only) Web-based resources. Similar approaches could be taken with Semantic Database Models other than the E-R model Labyrinth uses. Hull and King have surveyed a number of semantic models [17], any of which could have been similarly used.

ARIADNE [1], is another research system, developed at the University of Southern California, that strives to enable the retrieval and integration of data from multiple Web sites. It uses a *mediation* architecture that relies on customized wrappers around Web data resources. ARIADNE is designed for domain-specific usage and requires the construction of a domain model, using the LOOM [23] knowledge representation system, that can be mapped to data resources and user queries. Recognizing the labor-intensive nature of generating wrappers for various Web data resources, ARIADNE's designers incorporated support for semi-automatic wrapper generation.

Infomaster [13], an information integration system at Stanford University, also uses a variety of wrappers to gain access to and integrate information from various resources (including those not Web-based). Infomaster's wrappers for WWW-accessible information are custom written for each application and enable interaction with a *facilitator*, that in turn, relies on a knowledge base to correctly map translation rules and schemas needed for query processing. A *reference schema* is used as a baseline for describing the translation rules for each data source.

The TSIMMIS [22] system, also developed at Stanford, employs wrappers that use the *Object Exchange Model* (OEM) to provide a uniform interface to its mediator component. The uniform use of the OEM data model eases the translation and integration functions of TSIMMIS needed to integrate multiple sources of heterogeneous information.

WIT differs from these three systems in a number of significant ways. The functionality of wrappers, facilitators, translators, and mediators is achieved by WIT's use of structured and semantically rich representation imposed over existing Web data. It also relies heavily on the E-R semantic model and requires users to have modest familiarity of the concept in order to conduct their queries (via manual hyperlink navigation). WIT also requires pre-existing ER-datawebs as input data resources. The construction of these ER-datawebs is akin to writing custom wrappers in terms of effort involved. However once constructed, these ER-datawebs can be integrated together in arbitrary ways—ways not possible using other systems.

Like the reference schema used by Infomaster, WIT generates an integrated schema that can be mapped onto the individual schemata of various data resources. WIT, however, generates this schema automatically—based on the schemata selected for participation in the integration process.

Moving toward more database-oriented approaches, Spertus and Stein introduced the concept of *just-in-time* or JIT databases [29]. The JIT prototype was influenced by the Web query languages WebSQL [24] and W3QL [20]. Essentially, JIT extensively parses Web-accessible artifacts, associated with one or more URLs, and organizes information about their internal structure (e.g., tags, anchors, links, headings, lists, strings, offsets, etc.) and the components of their URLs (e.g., host, port, path, etc.) into relational database tables—one set for each Web page. SQL queries can then be made on the resulting relational database. Obviously, JIT differs from WIT in its much finer granularity and ability to highlight detailed information about structure within Web pages. WIT, on the other hand, uses the Web page as the lowest level of granularity. Both systems, however, use sufficient structure to enable straightforward integration of multiple databases or datawebs.

The Open Hypermedia Systems community has, for a number of years, been augmenting datawebs with tools that increase their functionality. Some of these tools address strategies for augmenting datawebs with supplemental structure in order to highlight relationships among data in the datawebs involved. The Chimera [2] system, for example, is a link server capable of storing supplemental link and anchor information about datawebs. Chimera works in conjunction with HTTP servers—translating, on the fly, its supplemental Chimera structural information into HTML and affording users a unified view of original and supplemental Web-based data resources via an ordinary Web browser. While Chimera's ability to integrate Web-accessible data with supplemental structure is superb, the system is not focused on any notion of schema or other rigid organization structure for adding semantics to the data.

6 Open Issues and Future Directions

The development of the first WIT prototype highlighted a number of areas where additional functionality could prove useful. Increased selectivity of integration and the ability to verify semantic uniformity across integrating ER-datawebs are perhaps the most obvious extensions. Implementing a mechanism for describing *import* and *export* schemata as described by Heimbigner [16] is one way selectivity of integration could be improved. Improving the accuracy of semantic uniformity will be much more difficult. Over the past few decades, resolving semantic heterogeneity issues among cooperating databases has proven to be the holy grail of integration research. WIT is unlikely to completely solve this problem in the future. Lastly, the ability to select the extent to which the organizing structures of the ER-datawebs are to be copied (and subsequently *owned*) by the integrator could also increase the utility of the tool. Copying desired portions of organizing structure would effectively enable the notion of *autonomy by degree* within federated ER-datawebs.

The focus of our next-generation WIT prototype is on increasing users' ability to select which *portions* of ER-datawebs are of interest and, as a result, should be included in integration operations. Conversely, we would like to increase the ability of participants in such federations to more closely control which portions of their ER-datawebs can be accessed and potentially included in the construction of integrated ER-datawebs by other WIT users.

7 Summary

The integration of traditional databases has always been challenging or, for those who don't control the databases involved, impossible. We believe that Web-based approaches to organizing data make new solutions possible for problems related to control, ownership, autonomy, selective integration, and collaborative use. They also promise to be particularly useful in fulfilling the specific data sharing needs of collaborating software engineers.

We have described a framework for organizing Web-accessible data into *datawebs* that, in turn, simplify the subsequent integration, or ability to present a unified view, of such data. After selecting one format for organizing datawebs from those currently in use, we built a tool that automates the process of (ER-)dataweb integration. Our Web Integration Tool, WIT, has been successfully demonstrated as a lightweight solution to the problem of integrating distinct subsets of Web-accessible data.

The WWW, with its nearly total connectivity, has been a natural platform for collaboration by owners and users of various types of databases. The merging of database and WWW technology has made a number of novel approaches to distributed database collaboration for integration and retrieval possible. In this paper we have discussed some of these approaches and their supporting tools.

Acknowledgments

We would like to thank Alfonso Fuggetta and Guiseppe Valetto of the CEFRIEL research center at Politecnico di Milano in Italy for bringing their Labyrinth project to our attention as well as for their technical assistance and access to the Labyrinth source code.

The work of Dennis Heimbigner and Alexander Wolf was supported in part by the Air Force Materiel Command, Rome Laboratory, and the Defense Advanced Research Projects Agency under Contract Numbers F30602-94-C-0253 and F30602-98-2-0163. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

References

[1] Jose Luis Ambite, Naveen Ashish, et al. ARIADNE: A System for Constructing Mediators for Internet Sources. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD '98)*, pages 561–563, 1998.

[2] Ken Anderson. An Overview of Chimera's Hypermedia Concepts. Available on the World Wide Web at *http://www.ics.uci.edu/pub/chimera/*, 1996.

[3] Yigal Arens and Craig Knoblock. SIMS: Retrieving and Integrating Information From Multiple Sources. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, ACM SIGMOD Record, pages 562–563, June 1993.

[4] C. Batini and M. Lenzerini. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4), December 1986.

[5] Athman Bouguettaya, Boualem Benatallah, and Ahmed Elmagarmid. *Interconnecting Heterogeneous Information Systems*. Advances In Database Systems. Kluwer Academic Publishers, Boston, 1998.

[6] Tim Bray and Steve DeRose. Extensible Markup Language (XML): Part 2. Linking. Available on the World Wide Web at *http://www.textuality.com/sgml-erb/WD-xml-link.html*, 1997.

[7] Tim Bray and C.M. Sperberg-McQueen. Extensible Markup Language (XML): Part 1. Syntax. Available on the World Wide Web at *http://www.textuality.com/sgml-erb/WD-xml-lang.html*, 1997.

[8] M.W. Bright, A.R. Hurson, and Simin H. Pakzad. A Taxonomy and Current Issues in Multidatabase Systems. *IEEE Computer*, 25(3):50–59, March 1992.

[9] Fabiano Cattaneo, Alfonso Fuggetta, Luigi Lavazza, and Giuseppe Valetto. Labyrinth: Schema-based Distributed Document Management on the Web. Technical Report RT 99002, CEFRIEL - Politecnico di Milano, Milan, Italy, February 1999. Available on the World Wide Web at *http://www.cefriel.it/se/projects/Labyrinth/*.

[10] P.P. Chen. The Entity-Relationship Model–Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[11] William W. Cohen. Providing Database-like Access to the Web Using Queries Based on Textual Similarity. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, ACM SIGMOD Record, pages 201–212, June 1998.

[12] William W. Cohen. Providing Database-like Access to the Web Using Queries Based on Textual Similarity. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, ACM SIGMOD Record, pages 558–560, June 1998.

[13] Michael R. Genesereth, Arthur M. Keller, and Oliver M. Duschka. Infomaster: An Information Integration System. *SIGMOD '97*, pages 539–542, 1997.

[14] Joachim Hammer and Dennis McLeod. An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems. *International Journal of Intelligent and Cooperative Information Systems*, 2(1):51–83, March 1993.

[15] M. Hammer and D. McLeod. Database Description with SDM: A Semantic Database Model. *ACM Transactions on Database Systems*, 6(3):351–386, September 1981.

[16] Dennis Heimbigner and Dennis McLeod. A Federated Architecture for Information Management. *ACM Transactions on Office Information Systems*, 3(3), July 1985.

[17] Richard Hull and Roger King. Semantic Database Modeling: Survey, Applications, and Research Issues. *ACM Computing Surveys*, 19(3):201–260, September 1987.

[18] Ali Hussam, Terry Anderson, et al. Semantic Highlighting. In *Proceedings of the CHI 98 Summary Conference on Human Factors in Computing Systems*, pages 191–192, 1998.

[19] D.D. Karunaratna, W.A. Gray, and N.J. Fiddian. Establishing a Knowledge Base to Assist Integration of Heterogeneous Databases. In Suzanne M. Embury, Nicholas J. Fiddian, W. Alex Gray, and Andrew C. Jones, editors, *16th British National Conference on Databases, BNCOD 16*, LNCS, pages 103–118, Berlin, July 1998. Springer-Verlag.

[20] David Konopnicki and Oded Shmueli. Information Gathering in the World-Wide Web: The W3QL Query Language and the W3QS System. *ACM Transactions on Database Systems*, 23(4):369–410, December 1998.

[21] Ora Lassila and Ralph R. Swick, editors. Resource Description Framework (RDF) Model and Syntax Specification. Available on the World Wide Web at *http://www.w3.org/TR/WD-rdf-syntax/*, 1998.

[22] Chen Li, Ramana Yerneni, et al. Capability Based Mediation in TSIMMIS. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD '98)*, pages 564–566, 1998.

[23] Robert MacGregor. A Deductive Pattern Matcher. In *Proceedings of AAAI-88, The National Conference on Artificial Intelligence*, 1988.

[24] Alberto O. Mendelzon and Tova Milo. Formal Models of Web Queries. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 134–143, 1997.

[25] E. Radeke, R. Böttger, et al. Efendi: Federated Database System of Cadlab. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, page 481, 1995.

[26] Ingo Schmitt and Can Türker. An Incremental Approach to Schema Integration by Refining Extensional Relationships. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM 98)*, pages 322–330, 1998.

[27] D. Shipman. The Functional Data Model and the Data Language DAPLEX. *ACM Transactions on Database Systems*, 6(1):140–173, March 1981.

[28] Munindar P. Singh, Philip E. Cannata, et al. The Carnot Heterogeneous Database Project: Implemented Applications. *Distributed and Parallel Databases*, 5(2):207–225, April 1997.

[29] Ellen Spertus and Lynn Andrea Stein. Just-In-Time Databases and the World-Wide-Web. *CICM '98*, pages 30–37, 1998.

[30] Kenji Takahashi. Metalevel Links: More Power to Your Links. *Communications of the ACM*, 41(7):103–105, July 1998.