INTEGRATED COLLABORATIVE INFORMATION SYSTEMS

Ahmet E. Topcu

Submitted to the faculty of the University Graduate School
in partial fulfillment of the requirements
for the degree
Doctor of Philosophy
in the Department of Computer Science,
Indiana University
March 2009

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

Prof. Geoffrey C. Fox (Principal Advisor)

Prof. Donald F. (Rick) McMullen

Prof. Luis M. Rocha

Prof. Jee H. Koh

Abstract

The evaluation of the Web shows that people want to access information easily, store them in a personal way, and share them with the others. There are numerous tools and services built in recent years in different categories having Web 2.0 capability. Examples include Social Bookmarking Tools (YouTube, del.icio.us, Flickr,), Blogs (blogger.com, Google Blog), Social Networking Tools (MySpace, LinkedIn) and other related tools. New tools and services are built and open to the Web community continuously. New blogs and data are published every second. The users of these tools have the opportunity to use different tools and decide the best ones in their perspective. Users don't need to know about the version of the tools and services. However, having many tools in similar areas is a problem. If a user wants to use some other tools, how can the user move the data from the previous tool to the new tool? What if the user decides to use similar tools in the same environment and compare information at the same time? In other words, users should have a flexible environment to use multiple tools at the same time. In the current Web 2.0 domain, it is not easy to say that which tools and services are the best because of the large number of existing tools and the continuous development of new tools.

In this dissertation, we present integration model and its components using web-accessible data and services and test application based on our integration infrastructure to evaluate our solution. The architecture have the following capabilities: (i) Tagging and linking of people through uploading and downloading of information; (ii) Sharing information; (iii) Supporting scientific research community; (iii) Integrating the new tools as they are generated in a specific area; (iv) Providing a dynamic environment in which

the user can benefit from the capabilities of different tools; (v) Allowing rich content. This architecture also provides interchange standards for common metadata format and provides structure for lowering the risk for user. This model is motivated by the above concerns to provide flexible mechanism to integrate similar Web 2.0 tools which have similar data model. We also present evaluation of the system to demonstrate applicability of this architecture integrating various search and annotation tools for scholarly publications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The evaluation of the Web shows that people want to access information easily, store them in a personal way, and share them with the others. There are numerous tools and services built in recent years to provide online collaboration and sharing information in different categories having Web 2.0 capability. Examples include Social Bookmarking Tools (YouTube, del.icio.us, Flickr), Blogs (blogger.com, Google Blog), Wikis (Wikipedia, WikiWikiWeb, Wikitravel), Syndication Feed Aggregators (Netvibes, YourLiveWire), Social Networking Tools (MySpace, LinkedIn) and other related tools. Web 2.0 [1] community applications indicated that new Web-based tools can gain millions of members and change the way of today's Web. This change can also be observed in the domain of scientific research communities enabling and sharing the scientific content, such as bookmarking tools CiteULike [2], Connotea [3], and

Bibsonomy [4], Domain specific academic search tools, such as CiteSeer [5], or general ones, such as Google Scholar [6].

These developments overlap with ongoing efforts to exploit Grid architectures based on Web services [7] for supporting international scientific and engineering research teams by enabling the sharing of large data and compute resources (i.e., creating a Cyberinfrastructure for e-Science [8, 9]).

The classic way of collaboration is posting information by individual or organization. The posted information can be accessed by anyone or selected users selected through by authorization tokens defined in the system. The main purpose of the collaboration is sharing information easily by any users or selected users or groups in web domain. For example Grid application provides sharing computer, resources, and network around the communities. The web is become popular because people want to share more information easily and they want to tag, comment, and rate the information by using the community tools.

Another way to access information is through digital libraries and academic search. Significant advances have also taken place in the areas of digital libraries and academic search. Domain specific academic search tools, such as CiteSeer [5], or general ones, such as Google Scholar [6], have enabled open, fast and easy access to vast online repositories of linked scientific documents.

Despite such important developments, there remains a great need for research tools geared toward niche communities of researchers. For example, currently there is no fast and reliable way to collect and analyze all the papers of a research group; a search in Google Scholar for the publications of our research lab (Community Grids Lab) will

return only about 20% of the desired content [10]. Similarly, there is no easy way to find all publications that focus on a very narrow topic, say all or almost all the papers discussing a particular chemical compound. Moreover, the new tools for annotating scholarly papers (CiteULike, Connotea) are currently detached from the capabilities provided by other research tools, such as academic search tools.

The following figure is depicted below shows a system model for using existing research systems such as academic search tools (Google Scholar, Windows Live Academic, CiteSeer) and social bookmarking tools (CiteULike, del.ici.us, Connotea). These tools can be used by adding new capabilities by building wrapper (constructed as Web services) that allow us to extract information and store information for scientific documents. For example del.icio.us tool can be used to manage tags and comments on the document while allowing custom tools manage the other information needed by user.

1.1 Motivation

In the Web community new tools and services are built and opened to users by providing publishing and sharing information, tagging, and storing them. New blogs and data are published every second. The users of these tools have the opportunity to use different tools and decide the best ones in their perspective. Users don't need to know about the version of the tools and services [11]. However, having many tools in similar areas is a problem. What if a user wants to use some other tools, how can they move the data from the previous tool to the new tool? What if the user decides to use similar tools in the same environment and compares information at the same time? In other words,

users should have a flexible environment to use multiple tools at the same time. In the current Web 2.0 domain, it is not easy to identify which tools and services are the best because of the large number of existing tools and the continuous development of new tools. In this thesis, we are particularly interested in integration of different tools, sharing information in this integrated community, and having additional functionalities for scientific research in a Service Oriented Architecture in order to use the different capabilities of the tools.

We identify the following limitations of the current approaches in Web 2.0 domain using tools for sharing and managing the information.

i. In the web domain, there are numerous annotation and search tools. Each of them has different capability and not complete defined metadata. Tools may provide services to the users such as tagging, posting, sharing information. However, in their category they do not provide a complete metadata definition to support the whole document.

ii. Each tool has different advantages over other tools and is currently detached from the capabilities provided by other research tools.

iii. The wealth of information contained in numerous fields remains largely outside the scope of tools. For example a search in Web Search Tool for specific group or lab publications may not return whole publications.

iv. If the tool you choose is not adopted or worse just disappears, what will happen? Users may use the one specific tool for storing metadata. When the tool disappears, users may lose personal stored information. There

should be an easy way not to depend on one tool. In this dynamic collaboration environment tools can easily disappear.

v. The information in current web domain is spread to web with different tools and there is no easy way to share the information using different tools, and users need to duplicate the same information in the different domains.

vi. Tools may have many features that other tools do not have. Users need to switch between tools and store the same information in them. Also these collaboration systems may not have the best performance or give full access to their systems that cause another limitation for tools.

1.2 Statement of research problems

In this thesis, we provide some assessment about recent developments, the problems defined previously, and investigate a novel approach of building Integrated Collaborative Information Systems Infrastructure to integrate various tools having common metadata that we think will have a model for the scientific research community. We particularly identify the following research questions for building such framework.

1. What are the architectural and implementation principles to use the Integrated Collaborative Information Systems (ICIS) Architecture? How easy is it to integrate new tool to the ICIS? How does Integrated Collaborative Information Systems (ICIS) depend on the tool?

2. Can we implement a framework by integrating tools that handle data and metadata from various tools and resources in Service Oriented Architecture?
3. How does Integrated Collaborative Information Systems (ICIS) use the tools? How does ICIS architecture provide a mechanism to link data between the tools?
4. How does ICIS architecture provide data sharing between users and protect data from other user?
5. Does system architecture provide to use the benefit of the integrated tools without developing new tool?
6. Does the integration architecture approach scales well?
7. What is the performance of the Integrated Collaborative Information Systems (ICIS) and what factors influence the performance?
8. Can we support data sharing and uploading and downloading for scholarly publications using our integrated architecture?
9. How does our integration model support the consistent data for scholarly publications?

We have answer to these questions in CHAPTER 7

.

1.3 Why Integrated Collaborative Information Systems Architecture

The web technologies such as RSS (Really Simple Syndication)[12], ATOM[13] AJAX (Asynchronous JavaScript and XML)[14], microformats[15], and REST (Representational State Transfer)[16] provide flexible Web-accessible data and services for Web 2.0 applications. However, although the current systems are for the most part good, they are independent of each other. Huge amount of data distributed over different tools and services exists in the Web. A large fraction of this data is duplicated. What is needed is an integration model that would bridge the different tools and services. In the 90s the software and system releases were not frequent. Now, people do not careen to know about version of the software and systems. That is not really needed because today's tools provide services that always improve [11]. There are many tools in Web 2.0 but we are not sure which tools will improve and will be embraced by the web communities. So, in this rapid development cycle one tool might have an advantage to the other tool and vice versa. For example, the annotation tools for scholarly papers are currently detached from the capabilities provided by other research tools.

One of the features of Web 2.0 is the focus on the people. The platform is motivated by questioning how people should interact with each other and easily share data in the Web. The resulting tools are easy to use, and allow people to put information and download them easily. However, there is no such a mechanism to combine them and have richer data or metadata integrated services. For example, one metadata captured from one resource may be needed to be stored, shared and uploaded to other tools. This can be achieved using Web services, or Web 2.0 technologies defined earlier such as

AJAX and REST. This model is created using native tools and wrappers around them without re-building the tools. Also local capabilities for example local search capabilities can be added and embedded in different client models, such as gadgets. So, the model has the capability to upload information to the tools and download information from them. The model should also provide sharing of logging of users. Our proposed Integrated Collaborative Information Systems Infrastructure (ICISI) offer a solution to build a framework by integrating tools by supporting distributed and centralized paradigms and managing metadata.

ICISI provide integration of community tools into Web 2.0 environment by building gateways and wrappers to existing community services by using Web services. In ICISI we have defined an architecture that is a model for integration to combine similar tools and use multiple services to user community. This architecture also provides flexibility allowing integration tools having common metadata. One major drawback of the tool integration is to performance of the integrated tools. For example gateway, provide communication between the tool and ICISI, depend on the tool to extract information from the tool, and tool itself may not have good performance response to the request from ICISI. However, ICISI does not depend on one tool and ICISI itself provides centralized repository to eliminate dependency. Another drawback of storing metadata information coming from various tools and processing time of the extracting information. However, by having faster and powerful computer, and cheap memory and hard drive, these limitations can be handled easily.

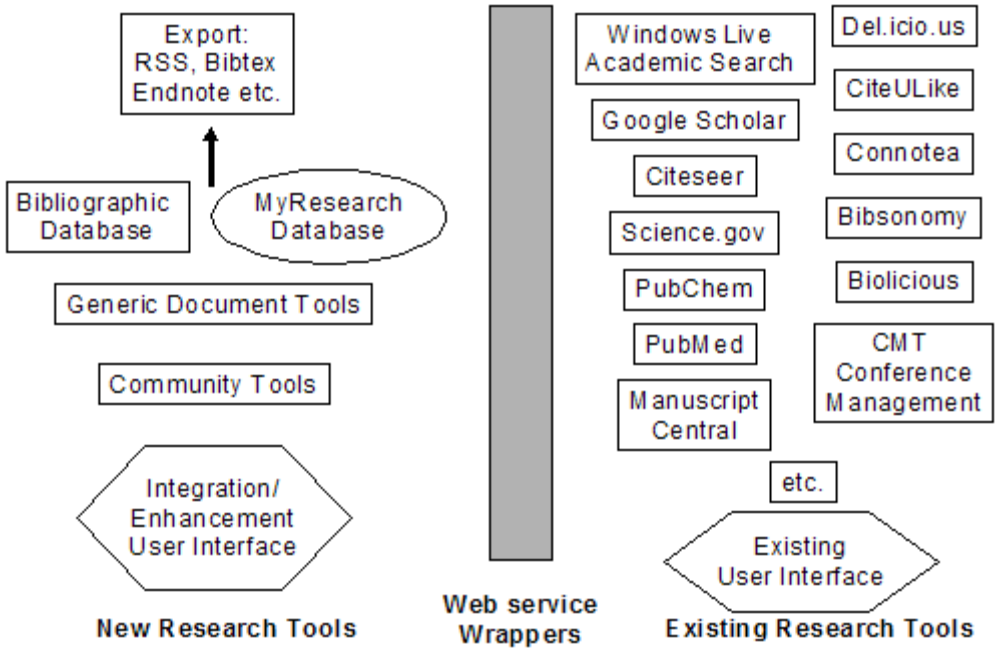


Figure 1-1 : Existing and New Research Tools having new capabilities for Scientific Documents

ICISI provides easily integration of the tools. We can easily add and remove service mechanism to collect information and upload information to the tools. ICISI also provides a protection for user data. Because it is possible that one tool might disappear. ICISI lowers the risk of losing data by protecting user data.

ICISI also provides a model that community building systems consist of mechanism to collect information stored in "central" location that offers input/output services. The model provides tagging and linking of people through uploading and downloading of information by supporting scientific research community. In ICISI, users have dynamic environment and share information to use the benefit of the different tools. Users also collect data from the various resources in order to enrich metadata and push collected metadata to other tools and users. Our aim is to build new tools and services

that add new capabilities by using wrappers to the major tools. These services allow one to both extract information and to store information in them. The tools and model are depicted in Figure 1-1.

ICISI provides following important features: (i) flexibility- providing add and extend service mechanism; (ii) easy integration; (iii) use benefits of different tools; (iv) build combined tools to share data with other users; (v) protect data from users.

1.4 Contributions

The main contribution of this thesis is to propose an architecture for Integrated Collaborative Information Systems (ICIS) supporting both distributed and centralized paradigms and managing metadata coming from various resources.

The implications of this thesis are seven-fold.

- Proposing novel architecture for integration of different Web 2.0 tools under one Collaboration Systems. This approach introduces the Integrated Collaboration Information Systems that link virtual organizations of tagging systems with other Cyberinfrastructure/Web 2.0 subsystems. Essentially, users have dynamic environment and share information to use the benefit of the different tools.
- Proposing a novel framework supporting a methodology to protect user data and allow users to manage information stored in repository and to share them with other users. Details of this methodology used in architecture are discussed in CHAPTER 4.

- Identifying and analyzing the key factors that affect both performance and scalability of the Integrated Collaboration Information Systems.
- Implementing the proposed Integrated Collaboration Information Systems Framework software and showing the integrated tools by using capabilities of different tools.
- Demonstrating an approach building wrappers using Web services around the tools to integrate them and add value to existing systems.
- Integrating scientific research tools by defining metadata and using various URL, and map them. This integration model used to support different environments where communities can take advantage of the tools in Web 2.0 integrated environments.
- Building services that aggregate information from a variety of sources (i.e., "mash-up" tools) and provide added value to communities of researchers. These services use metadata linking and using various URL, and map them.

1.5 Organization of the Thesis

This dissertation is arranged in seven chapters. An overview, which consists of the Introduction of CHAPTER 1 and Conclusions of CHAPTER 7, covers the general context of the research. CHAPTER 2 reviews the background information and review of the core technologies. Architecture and design approach of the Integrated Collaboration Information Systems Framework are described in CHAPTER 3 which provides description of the system structure. CHAPTER 4 describes access control model for protecting user data from other users and explain the methodology to share information

between users in the Integrated Collaboration Information Systems Framework. The prototype system is discussed in CHAPTER 5 which demonstrates the applicability of the Integrated Collaboration Information Systems Framework. The system is CHAPTER 6 presents performance measurement and analysis of the Integrated Collaboration Information Systems Framework. Finally, we conclude in CHAPTER 7 with a discussion of future work by outlining areas for future research directions.

CHAPTER 2

BACKGROUND AND SURVEY OF TECHNOLOGIES

2.1 Academic Web Search Tools

The advent of the World Wide Web (WWW) has led to the creation of a number of digital databases of scientific content. These databases are created using one of two main methods: (i) manual insertion by volunteers (e.g., DBLP) (ii) automated harvesting of content by crawling open-access databases, home pages of authors, web sites of the publication venues, and so on (e.g., CiteSeer). Both methods may be complemented with user submissions.

In this section, we explain the major open-access academic search tools and technologies that use various methods of acquiring and analyzing scientific documents. These tools are discussed next.

2.1.1 Google Scholar

The methods for collecting and analyzing documents used by Google Scholar (GS) are similar to those of CiteSeer. The Google Scholar on About Page states that "Google Scholar provides a simple way to broadly search for scholarly literature. From one place, you can search across many disciplines and sources: peer-reviewed papers, theses, books, abstracts and articles, from academic publishers, professional societies, preprint repositories, universities and other scholarly organizations. Google Scholar helps you identify the most relevant research across the world of scholarly research." [17]. GS provides following capabilities: (i) searching resources from one web platform; (ii) finding papers, abstracts, and citations;(iii) locating the full article through libraries or on the web; (iv) learning about articles in any area of research using advanced search. Note that CiteSeer is both a search system and a digital library having currently more than 800,000 full-text documents in its repository, while GS is a search system which attempts to find and display the URLs that point to the full-text versions of the query results. Unlike CiteSeer, GS aspires to be a "single place to find scholarly materials" covering "all research areas, and all sources" [18]. GS has been generally lauded for the open, fast and easy access it provides to vast collections of digital academic documents. There has also been significant criticism towards GS, especially from librarians. The major criticism has to do with: (j) *scope* (GS does not declare which publishers it currently

covers; at the same time it is known it does not cover some major publishers, such as Elsevier, American Chemical Society, and Emerald [18, 19]); although GS has access to digital archives of the largest academic publishers, the crawlers of GS have not indexed millions of publications;[19] (jj) *coverage* (GS does not provide full coverage of the articles from the publishers that seem to be covered [18, 19]); (jjj) *accuracy* (its metadata extraction algorithms are not very precise, leading to duplicate records, unreliable citation counts, etc. [19]).

GS crawlers were allowed to access largest and most well-known scholarly publishers and university presses (such as IEEE, ACM, Wiley); digital hosts(such as HighWire Press, MetaPress, Igenta); societies; government agencies (such as the American Physical Society, National Institute of Health); other scholarly organizations and preprint servers(such as arXiv.org, CiteBase)[19]. GS provides digital full text versions of the publications by searching from one web location having option to search multiple databases. This provides great beneficiaries to patron of libraries without repeating search queries for different databases [19].

Google Scholar also provides the number of citations for publications. It is very important feature because number of the citations is valuable information for scholarly publications.

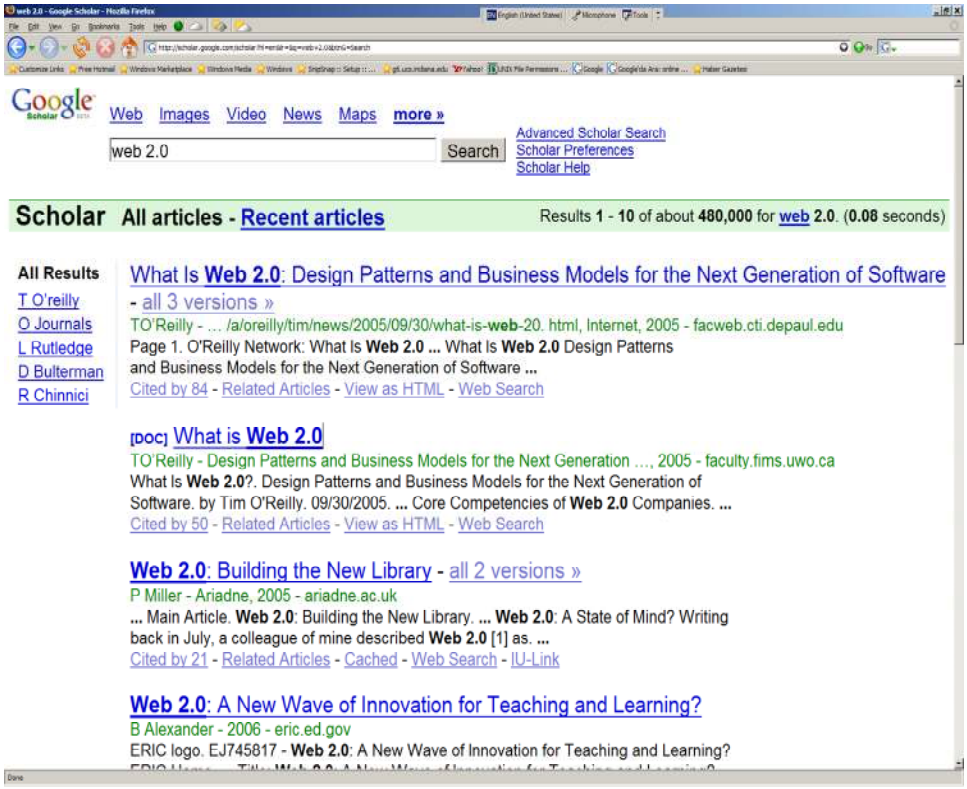


Figure 2-1 : Google Scholar Web Search Tool

Another feature of Google Scholar is to provide "versioning" .These versions may come from publication databases, university resources, and publication web pages. However, there is no clear information indicating "official" version of the document. This confuses the researchers and may be problem for accessing to the final publications[20]. Since GS provides services and update versions are available immediately on web page. So, user does not need to update the GS to have latest version of the service.

2.1.2 CiteSeer

CiteSeer was introduced in 1997 by Giles et al. [5]. As the first tool in this category, CiteSeer is probably also the best known, especially in the field of Computer Science, which is its specialization domain. The core feature of CiteSeer is Automated

Citation Indexing, a method for the automated extraction, parsing and indexing of the citations contained in a paper and of the context of these citations in the paper's body. CiteSeer has pioneered a number of techniques for the automated extraction of document metadata, including front-end metadata such as title, author names, author affiliations, abstract, and back-end metadata, such as acknowledgements, and citations to other papers. The algorithms used by CiteSeer are generally based on carefully crafted heuristics and/or machine learning techniques. Giles et al. [5] states that automatic citation indexing operation is constructed using following steps:

1. Papers available from World Wide Web are downloaded.
2. CiteSeer converts the papers to text.
3. Papers in text format are parsed to extract the citations and it's context in the body of the paper
4. All the extracted information is stored in the database.

CiteSeer extracts the following information by using PreScript from the New Zealand Digital Library project[21]:

- *URL:* The URL of the downloaded Postscript file is stored.
- *Header:* The title and author block of the paper is extracted.
- *Abstract:* If it exists, the abstract text is extracted.
- *Introduction:* If it exists, the introduction section is extracted.
- *Citations:* The list of references made by the document are extracted and parsed further as described below.
- *Citation context:* The context in which the document makes the citations is extracted from the body of the document.

- *Full text:* The full text of the document and citations is indexed [5].

Recently, it was estimated that CiteSeer covers about 24% of papers in Computer Science and it was pointed out that the use of automated methods for harvesting documents has led to a bias toward papers with three or more authors [22]. The CiteSeer also have been shown to contain only a small fraction of such collections in their repositories. For example, Zhuang et al. [23] found that in 2005 CiteSeer contained 25.42% of papers from the International Workshop on the Web and Databases (WebDB) and 26.9% of papers from the Journal of Artificial Intelligence Research (JAIR) for the period 1998-2004. Using focused crawling techniques [24] these authors were able to automatically collect about 81% of papers from these two venues. To deal with issues such as increasing query latency and degradation of system stability, as well as to improve the interoperability of the system, CiteSeer has recently announced the design of a new version of the system, called CiteSeer^{X} [25].

Figure 2-2 : CiteSeer Web Search Tool

2.1.3 Windows Live Academic

Windows Live Academic (WLA) is the latest addition in the area of open-access academic search tools. It became public in 2006. However , Microsoft announced the end of Windows Live Academic indexed 80 million journal articles on May 23, 2008 [26]. Its objectives are similar to those of GS, but unlike GS it has revealed the list of the covered publishers and venues. The initial version of this tool, has been shown to suffer from the same issues of coverage and accuracy discussed above for GS [27]. Another drawback of WLA is that, unlike CiteSeer and GS, it does not yet provide citation indexing.GS has also another advantage over WLA having much faster search response time. These performance drawback of WLA is pointed by some reviewers[28].

WLA has very promising features for users to control of their searches. For example, Quint [29] state following features for WLA:

- *Sort and limit results by author, date (forward or reverse chronological order), journal, conference, or back to the default, relevance.*
- *Use the "Richness Slider" to expand or contract the relevance ranked display of search results.*
- *Click on author names for author bibliographies (with no limits on the numbers of authors "clickable"; no "et al.").*
- *View an abstract of each search citation in a preview pane section on the right of the screen or "hide abstract" if you just want to scan brief citations as quickly as possible using the full screen.*
- *Export citation in basic text, RIS' EndNote, or BibTeX bibliographic formats.*
- *Find in a library using the connection to OCLC's Open WorldCat service (on its way).*
- *Save search strategies in macros using "Search Builder," which will also support RSS feeds ("Feeds" at the top of the screen tied to a "+live.com" option) to alert searchers as new results appear on their Live.com page.*

.

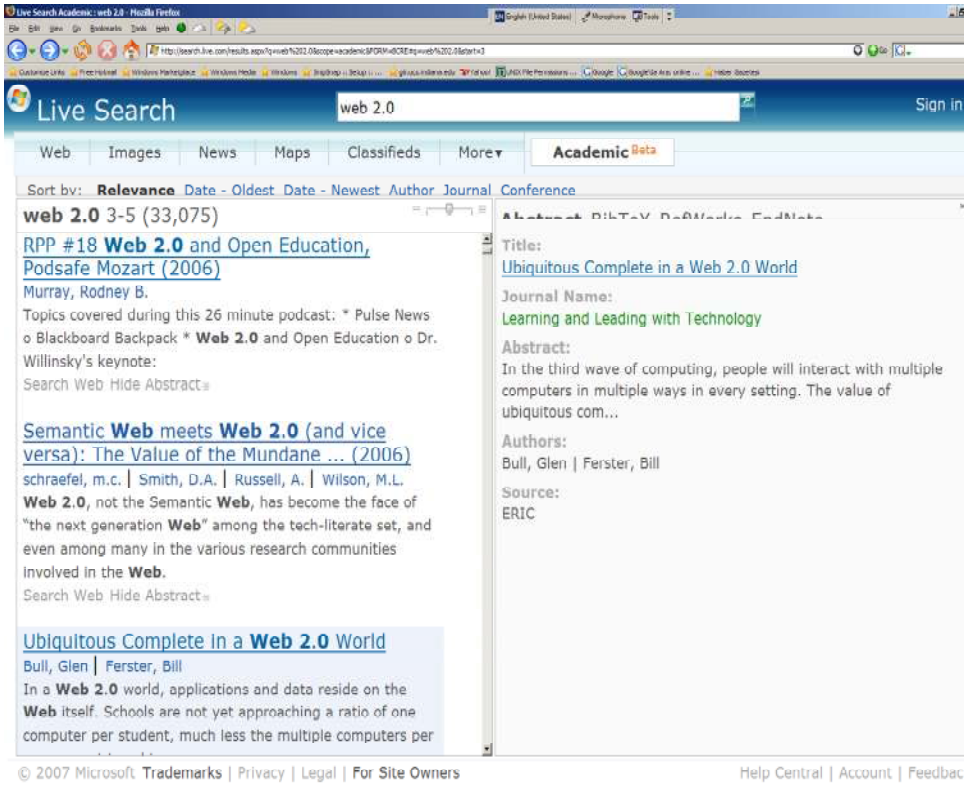


Figure 2-3 : Windows Live Academic Web Search Tool

2.2 Annotation Tools

2.2.1 CiteULike

CiteULike [2] is a online community bookmarking tool which provides service for managing and finding scholarly publications. This tool provides the following features: (i) sharing references with your community; (ii) storing references; (iii) discovering new publications; (iv) tagging services for papers.

CiteULike extracts the citations automatically so users do not need to do some other operations to build citation metadata objects.

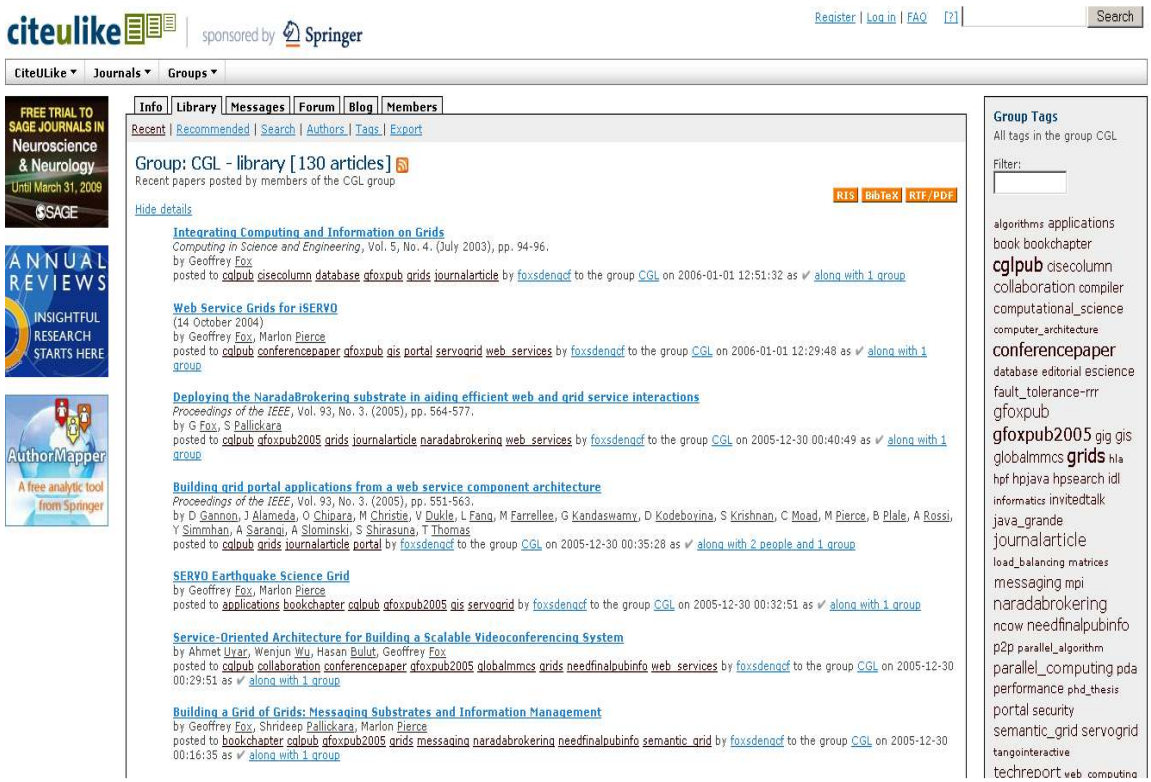


Figure 2-4 : CiteULike Annotation Tool

2.2.2 Connotea

Connotea [3] inspired by del.icio.us provides social bookmarking services and reference management. Connotea is developed by Nature Publishing Group [30] is become public in 2004.The key concepts of Connotea can be listed as: (i) Online storage of references and bookmarks; (ii) Simple, non-hierarchical organizing (iii) Opening the list to others; (iv) Auto-discovery of bibliographic information [31].

Connotea provides bookmark manager allowing user to add to Connotea the web page they are currently viewing. Therefore, user can add relevant information automatically by Connotea from the web site. Another important information related to the article is a tag that provides valuable data about publication data. Tags in Connotea

can be word or phrases. Besides adding tags, users have option to add a comment to the article. The comments which are displayed chronological and conversational thread can be read by other users or viewers. Connotea also provides service for recognizing URL's from common archives and importing bibliographic data. When user add a URL to Connotea, the link is analyzed to decide whether it can be processed by Connotea to extract the publication name, volume, issue number, and other its related metadata. Another feature of the Connotea is providing user subscription to the RSS [32] feeds in order them to see the list of the bookmark or alert new items that are added.

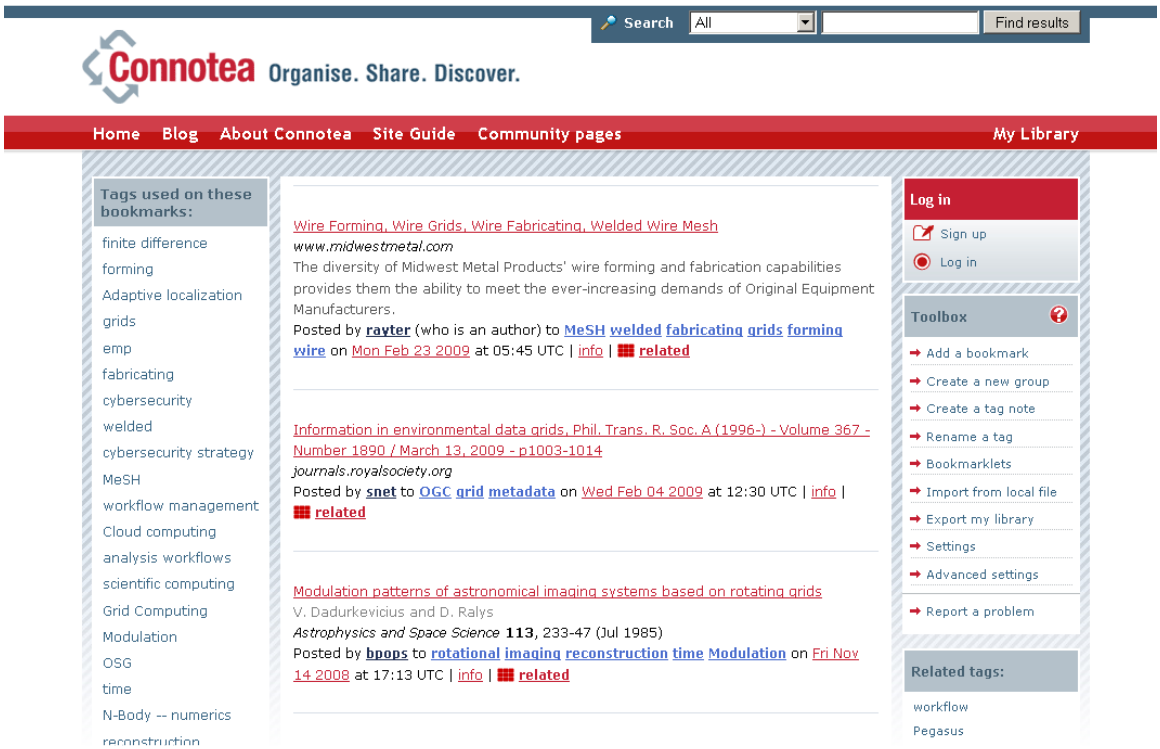


Figure 2-5 : Connotea Annotation Tool

2.2.3 del.icio.us

del.icio.us [33] is a online bookmarking tool to allow users to post their links, tags and other metadata, share them with other people. This tool also provides the following features: (i) access more popular bookmarks; (ii) search the contents; (iii) tagging services. Biddulph [34] states that the del.icio.us has important three major axes: users, tags, and URLs. For example;

- http://del.icio.us/mike is mike's URLs.
- http://del.icio.us/tag/xml is URLs with the tag xml.
- http://del.icio.us/url/8b7fec48fcb35763c9f8e1a8061eb124 is a reference to the URL http://www.xml.com/ (8b7fec48fcb35763c9f8e1a8061eb124 is the md5sum digest of the URL).

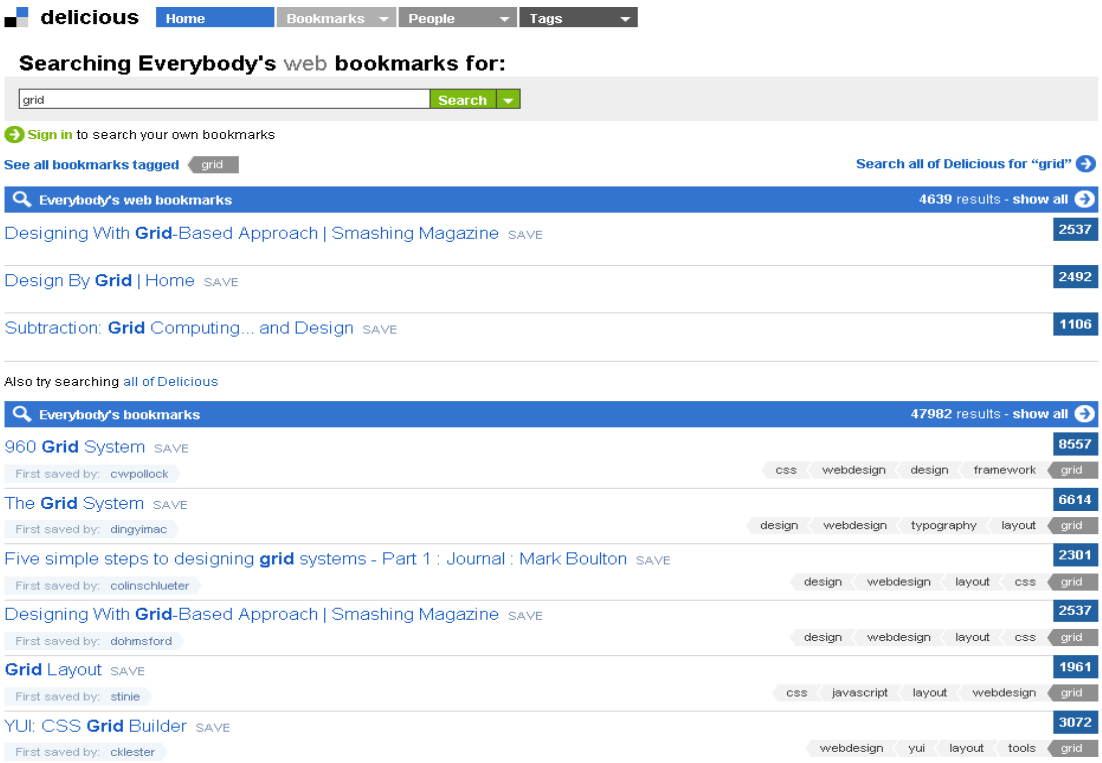


Figure 2-6 : del.icio.us Annotation Tool

These annotation tools are very useful and superior to custom databases, and use similar approaches to bibtex files for managing information. They provide valuable direction in information systems. However, they cannot replace conventional web pages like CGL publications web page recording publications. User can not generate web pages using CiteUlike and Connotea annotation tools as the RSS/bibtex exports from CiteULike does not display full information while Connotea has only small subset of needed information[10].

Connotea, del.icio.us, and CiteULike all use "bookmarklets" to analyze the current page by invoking JavaScript. There might be problem if users try to add a document specified by a DOI. One should record only DOI itself not the particular baseurl. CiteULike recognizes all DOIs whereas Connotea does not recognize all DOIs.

Edit and tag addition date are important but these dates are not recorded by Connotea and CiteULike only dates are stored from initial submission.

Connotea and CiteULike have similar capabilities such as:

- View, search, discover, and add/edit user interfaces.
- Add tags to your or other entries.
- Add URLs/DOIs and bibliographic metadata.
- Extract information from current page using bookmarklets.
- Allow various search criteria on title, authors, abstract, journal, tags, and user.

2.3 Grids and Web 2.0

Grid computing introduces solution to problems defined in [35] can be summarized as: (i) highly controlled research sharing; (ii) problem solving in Virtual Organization (VO) which is set of individuals and institutions such as The NSF TeraGrid [36] and NSF/DOE Open Science Grid [37]. Grids support Web Services architecturally [38] and XML is important for Web Service-based SOA systems [39]. Open Grid Service Architecture (OGSA) [40] and Semantic Web [41] both depend on XML metadata, and Web Services invokes remote methods by using XML-based protocol and interface definitions. This message protocol, usually SOAP [42] is bound to a lower level transport protocol such as HTTP. The method interface expressed in WSDL [43] describes agreed-up set of methods, parameters and return type of services.

Web 2.0 bring new ideas and directions to the scientific community similar to impact of Grid [35, 44] which provides resource sharing, innovative applications, high performance data orientation, and collaboration. One part of Web 2.0 is mash-up, workflow, and collaboration which widely adopted by web communities using tools such as Social Bookmarking (YouTube [45], del.icio.us [33], Flickr [46]), Blogs (blogger.com, Google Blog), Social Networking (MySpace, LinkedIn), Maps (Google Map) and other related tools.

Web 2.0 is a web platform having community applications demonstrated that millions of members can use these systems easily by sending receiving content using programming interfaces. O'Reilly [1] summarized the principal features of Web 2.0 below:

- *Services, not packaged software, with cost-effective scalability*
- *Control over unique, hard-to-recreate data sources that get richer as more people use them*
- *Trusting users as co-developers*
- *Harnessing collective intelligence*
- *Leveraging the long tail through customer self-service*
- *Software above the level of a single device*
- *Lightweight user interfaces, development models, AND business models*

Fox et al.[47] make the comparison of Web 2.0 and Grids in the following tables that are copied from Table I and II of Fox et al.[47].

Table 2-1: Grid view of e-Science features

	Feature	Grid approach
1.	Community building	Designed to enable virtual organizations based on collaborations between existing organizations such as research groups and supercomputing centers. Top-down approach, closely tied to PKI-based security infrastructure
2.	Collaboration	Focused on real-time audio/video collaborations such as access Grid. Virtual organizations provide a framework but typically no interesting functions for asynchronous collaboration
3.	Semantic and ontological representation of metadata	Semantic Grid efforts follow closely the semantic Web and use RDF, OWL for information representation. These can be used for both describing metadata and the contents of digital libraries as well as workflows

Table 2-2 : Web 2.0 view of e-Science features

Feature	Web 2.0 approach
1. Community building	Web 2.0 communities are typically networks of emergent groups of individuals with shared interests. Facebook, MySpaces, and Flickr are prominent examples
2. Collaboration	Dominated by asynchronous collaboration: group-edited content Wikis), shared commenting/rating/tagging of online content. Collaboration and community building are intertwined
3. Semantic and ontological representation of metadata	Metadata described by Microformats (semantic XHTML extensions) that represent community consensus and convention. Ontologies are replaced by "folksonomies" of conventional tags used to describe a network entity

2.4 Collaboration Tools

Sakai [48] is an academic open source online Collaboration and Learning Environment (CLE) project that support learning, research and collaboration. It has been used over 100 institutions. First version of the project was released in March 2005. The Sakai provides features are common to course management systems such as sharing course material, gradebook, chat, uploading/downloading assignments, and online testing. Another feature of Sakai is to provide collaborative environment for researchers.

Drupal [49] is another open source Content Management System supports features such as file uploads and downloads, blogs, collaborative authoring environments, forums, peer-to-peer networking, newsletters, podcasting, and picture galleries. It allows

users to publish resources and manage them. Drupal aim is defined in [50] as “Drupal is designed to be flexible and powerful enough to meet a broad range of web technology needs, from simple informational postings to large organizational sites and collaborative projects. This said, there is acentral interest in and focus on communities and collaboration. Drupal aims to enable the collaborative production of online information systems and communities.”

Microsoft SharePoint [51] is one of the community tool developed by Microsoft. It is explained in [52] saying that “Microsoft SharePoint enables groups to configure portals and hierarchies of websites without specifically requiring web-development. This allows groups of end users, as participants, to have much greater control in finding, creating, collecting, organizing, and collaborating on relevant information, in a browser-based environment. It also allows views of the different collections of information to be easily filtered, grouped, and/or sorted by each consumer according to their current desire. It has a robust permissions structure, allowing organizations to target users' access and capabilities based on their organizational role, team membership, interest, security group, or any other membership criteria that can be defined.”

The other platforms explained in [53] for building online communities using Web 2.0 features that enable sharing information such as Joomla(one of the most widely used content management systems and community platforms based on PHP) [54] , PHP-NUKE (One of the older CMS/community platforms) [55], Community Server (One of the few .NET blog platforms has evolved into a full-blown community product) [56], and jive (popular in the enterprise space) [57].

2.5 Technologies

We described the main technologies that are important to our proposed research thesis design implementation in the following sections.

2.5.1 Web Services

WSDL is an XML (provides a language which can be used between different platforms and programming languages and still express complex messages and functions) used to describe Web Services. It provides information about location of the services, how to locate them, on what the service is about, where it resides and how it can be invoked. One of the important benefits of the Web service is to provide interoperability. Therefore, different distributed Web services run on different platforms and architecture[58]. Following elements used in defining the WSDL document described in [59]: (i) Types– a container for data type definitions using some type system (such as XSD); (ii) Message– an abstract, typed definition of the data being communicated; (iii) Operation– an abstract description of an action supported by the service; (iv) Port Type– an abstract set of operations supported by one or more endpoints; (v) Binding– a concrete protocol and data format specification for a particular port type; (vi) Port– a single endpoint defined as a combination of a binding and a network address; (vii) Service– a collection of related endpoints[59].

SOAP stands for Simple Object Access Protocol and used for communication between client and server applications, and it is platform independent. SOAP is simple and extensible and language independent. SOAP request sample shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <ns1:echoString xmlns:ns1="http://soapinterop.org/">
      <arg0 xsi:type="xsd:string">Hello!</arg0>
    </ns1:echoString>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Web services have many advantages over others services; (i) protocol independent services; (ii) well-defined interfaces for distributed services; (iii) separation of interface from implementation (transparency); (iv) communicate using open protocols; (v) used by other applications easily; (vi) can be discovered using Universal Description Discovery and Integration(UDDI) [60].

In order to discover the services, UDDI is used as a platform-independent framework for describing services used by Web services on the web domain. The clients search the UDDI repository to find out the desired services.

2.5.2 Apache Axis 1.x

Apache Axis [61] is an implementation of the SOAP ("Simple Object Access Protocol") which is defined in W3C speciation: SOAP is a lightweight protocol in a decentralized, distributed environment and provides exchanging structured information in this environment [62]. But Axis isn't just a SOAP engine. It also includes specified in [61]: (i) a simple stand-alone server; (ii) a server which plugs into servlet engines such as Tomcat; (iii) extensive support for the *Web Service Description Language (WSDL)*; (iv) emitter tooling that generates Java classes from WSDL; (v) some sample programs, and a tool for monitoring TCP/IP packets. Axis 1.x also delivers the following key features explained in [61]:

- **Speed:** Axis uses SAX (event-based) parsing to achieve significantly greater speed than earlier versions of Apache SOAP.
- **Flexibility:** The Axis architecture gives the developer complete freedom to insert extensions into the engine for custom header processing, system management, or anything else you can imagine.
- **Stability:** Axis defines a set of **published interfaces** which change relatively slowly compared to the rest of Axis.
- **Component-oriented deployment:** You can easily define reusable networks of Handlers to implement common patterns of processing for your applications, or to distribute to partners.
- **Transport framework:** Clean and simple abstraction for designing transports is defined (i.e., senders and listeners for SOAP over various protocols such as SMTP, FTP, message-oriented middleware, etc), and the core of the engine is completely transport-independent.
- **WSDL support:** Axis supports the Web Service Description Language, version 1.1, which allows you to easily build stubs to access remote services, and also to automatically export machine-readable descriptions of your deployed services from Axis.

There are also four services defined in Axis indicated in [61].

- **RPC** services use the SOAP RPC conventions, and also the SOAP encoding.
- **Document** services do not use any encoding but DO still do XML to Java or Java to XML databinding.

- **Wrapped** services are just like document services, except that rather than binding the entire SOAP body into one big structure, they "unwrap" it into individual parameters.
- **Message** services receive and return arbitrary XML in the SOAP Envelope without any type mapping / data binding. If users want to work with the raw XML of the incoming and outgoing SOAP Envelopes, they write a message service.

2.5.3 Parsers

The DOM (Document Object Model) [63] developed by the World Wide Web Consortium defines a standard way for accessing and manipulating documents. The nodes in documents may be elements, their text, and their attributes. In DOM all the information from the document needs to be read into memory and stored in a tree a structure. Information stored in the memory as a tree structure has been parsed and client applications access to the structured information. Therefore, client can use the following features:

i. XML documents can be viewed as a tree structure.
ii. All elements can be extracted through the DOM tree.
iii. Element contents can be modified or deleted.
iv. New element or node can be added to the XML documents.

The node types, and their node types they may have as children, are explained [64]:

- **Document** -- Element (maximum of one), ProcessingInstruction, Comment, DocumentType

- **DocumentFragment** -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- **DocumentType** -- no children
- **EntityReference** -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- **Element** -- Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
- **Attr** -- Text, EntityReference
- **ProcessingInstruction** -- no children
- **Comment** -- no children
- **Text** -- no children
- **CDATASection** -- no children
- **Entity** -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference

This DOM approach has advantage when users have small size of data, and information will be taken directly from memory. However, it has some disadvantage for example bigger size of the document affects the performance having larger memory consumption. Moreover, it may not keep whole document in a memory at one time, and whole document must be successfully parsed before client program access to the information [65].

Simple API for XML (SAX) [66] processes XML by using the least amount of system resources. Also, simple structure and the lightweight nature of the SAX provided

easy adaptation in many communities. SAX has event based API instead of storing the document in a memory and passing it to the client program. Then, client registers to receive notifications for various part of the document. The performance will be stable even for larger documents. The client program does not wait until entire document has been read. It can start to process the document earlier. This provides advantages when network bandwidth is an issue. However, it has some disadvantages for example when client need the same part of the document later. Because, there is comprehensive model of the document is available in memory in SAX. In this case using DOM model would be more appropriate [65].

2.5.4 HTTP Client

The Hypertext Transfer Protocol (HTTP) is defined in [67] as "an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers." Three commonly methods are used by HTTP: (i) GET- retrieve information from the server; (ii) POST – Data is sent to server by the client as a part of the request; (iii) HEAD - find out whether information is available on the server.

Jakarta Commons HTTP Client is an open source project and implementation of HTTP using Java Programming Language. It provides following features explained in [68]:

- Standards based implementation of HTTP versions 1.0 and 1.1 using Java.

- Full implementation of all HTTP methods (GET, POST, PUT, DELETE, HEAD, OPTIONS, and TRACE) in an extensible OO framework.
- Supports encryption with HTTPS (HTTP over SSL) protocol.
- Granular non-standards configuration and tracking.
- Transparent connections through HTTP proxies.
- Tunneled HTTPS connections through HTTP proxies, via the CONNECT method.
- Transparent connections through SOCKS proxies (version 4 & 5) using native Java socket support.
- Authentication using Basic, Digest and the encrypting NTLM (NT Lan Manager) methods.
- Plug-in mechanism for custom authentication methods.
- Multi-Part form POST for uploading large files.
- Pluggable secure sockets implementations, making it easier to use third party solutions
- Connection management support for use in multi-threaded applications. Supports setting the maximum total connections as well as the maximum connections per host. Detects and closes stale connections.
- Automatic Cookie handling for reading Set-Cookie: headers from the server and sending them back out in a Cookie: header when appropriate.
- Plug-in mechanism for custom cookie policies.
- Request output streams to avoid buffering any content body by streaming directly to the socket to the server.

- Response input streams to efficiently read the response body by streaming directly from the socket to the server.
- Persistent connections using KeepAlive in HTTP/1.0 and persistence in HTTP/1.1
- Direct access to the response code and headers sent by the server.
- The ability to set connection timeouts.
- HttpMethods implement the Command Pattern to allow for parallel requests and efficient re-use of connections.
- Source code is freely available under the Apache Software License.

2.5.5 AJAX and JSON

Asynchronous JavaScript and XML (AJAX) is a combination of the web client technologies for building interactive web applications. It provides new approach for web application providing simplicity and independent of communication with the server using JavaScript's XMLHttpRequest class methods. XMLHttpRequest returns an XML messages that can be parsed using JavaScript[69].

Ajax incorporates the followings explained in [69]:

- Standards-based presentation using XHTML and CSS.
- Dynamic display and interaction using the Document Object Model.
- Data interchange and manipulation using XML and XSLT.
- Asynchronous data retrieval using XMLHttpRequest

- JavaScript binding everything together.

JavaScript Object Notation (JSON) is an alternative to XML for data transmission in the web. It has simple data structure and no parsing is necessary for extracting information in Ajax applications. This has advantage if one wants to use pass simple datatype such as string, number, and array over the internet [70].

2.5.6 XPATH

XML document has tree of nodes, and XPath [71] models these nodes which have different types of nodes such as element nodes, attribute nodes and text nodes. XSLT [72], XPointer [73] and other XML parsing software uses XPath to extract specific information from structured objects. It also provides mechanism to manipulate strings, numbers and booleans. XPath addresses the parts of an XML document which has been modeled as tree of nodes. In the following example, XPath points to the author node in a structured XML document.

XPath authorQuery =node.createXPath ("//item/livesearch:academic/author")

2.5.7 Simple Message Formats: RSS and Atom Feed

Really Simple Syndication (RSS) provides syndication of Web site content. Since RSS data size small, it can be loaded fast on the wire [32]. There are three required channel elements for defining object in RSS file:

- title - defines the title of the item
- link - defines the hyperlink to the item

- description - describes the item

Atom is another way of syndication of information. It is supported by many news reader clients. Possible use of Atom Feeds explained in [74].

- **Feed:** Use for computer-friendly syndication -- similar to RSS in functionality.
- **Editing:** Use between atom-enabled clients and servers to create and update entries, comments, and either annotate or edit other resources.
- **Archiving:** Use for internal storage, import, and export of entries, comments, and other resources.
- **Commenting:** Use to post comments to an entry.
- **Cross-linking:** Use to notify one entry about another entry.

CHAPTER 3

INTEGRATED COLLABORATION INFORMATION SYSTEMS ARCHITECTURE

There is no precise definition of integration in the literature. It is not a property of a single tool but it should have relationship with other tools in the environment [75]. Integration can be categorized into five kinds: (i) *platform*, related with framework services; (ii) *presentation,* concerned with user interaction; (iii) *data*, using information in the tools; (iv) *control*, mechanism for tool communication and interoperation ; (v) *process*, related to roles of tools in the systems [76]. The aim of integration is to transform multiple tools into one useful and flexible environment for building communities and to provide multi-functional services to the users. We aim to build such a flexible mechanism by using an integration model to use the benefits of different tools to build collaborative environment.

The model should have the following capabilities: (i) Tagging and linking of people through uploading and downloading of information; (ii) Sharing information; (iii) Supporting scientific research community; (iii) Integrating the new tools as they are generated in a specific area; (iv) Providing a dynamic environment in which the user can benefit from the capabilities of different tools; (v) Allowing rich content.

The integration model itself doesn't build new tools. It uses the existing tools. One of the application areas is in academic search. In the following section we will define the integration model of similar tools. The key feature is to reuse the tools so that there is no need to rewrite a new tool for specific domain. So, the proposed model should be easier to link together all relating information.

Interoperability for integration is to decide how much work needs to be done for getting data from one platform and use it in other system. Successful integration can be done with respect to interoperability if a system requires having little work to reach data or metadata used in the tools. We define a model that community building systems consist of mechanism to collect information stored in "central" location that offers input/output services. These services should be complete with WSDL (Web Service Definition Language) interfaces to provide wrapper services [77]. These systems should also provide mechanism to have simple internet-scale programming approach such as asynchronous JavaScript and JavaScript Object Notification (JSON), gadgets to make integration powerful and flexible for different systems.

3.1 General picture of the Integrated Collaboration Information Systems (ICIS) Architecture

Integrated Collaboration Information Systems (ICIS) architecture has communication with tools and clients through gateways defined in this model. The different gateways is defined and implemented for each tool to have successful interaction between them to extract information from them or store information in them. ICIS architecture also provides services to build community-centric platform for users. The simplified view of this architecture is depicted below in Figure 3-1.

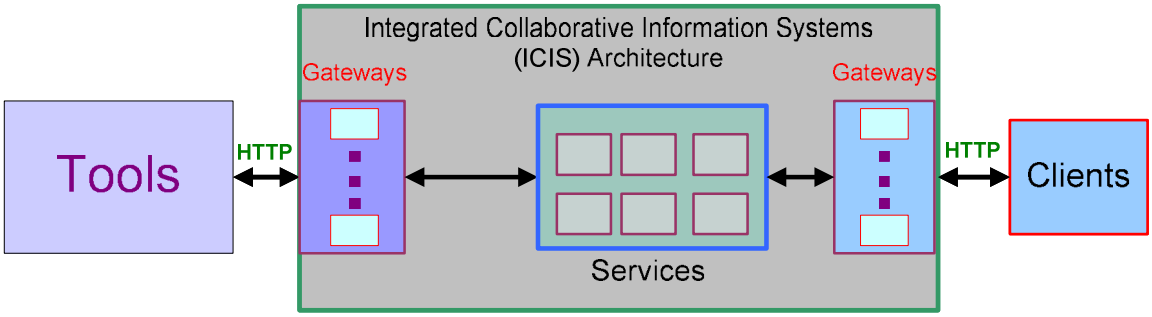


Figure 3-1 : Simplified view of the ICIS Architecture

Figure 3-2 shows the detailed view of the overall architecture of the integration model. This system consists of six components: (a) *Tools*, external web tools to provide a service to the clients; (b) *Integration Manager*, have information service and provide a communication between the tools, and clients. It is responsible for integration operation in the system; (c) *Filter*, operate two-way data filtering; (d) *Permission Handler*, checks existing Digital Entity (DE) permission or build a new permission token for new DEs; (e)

Data Manager, provides a mechanisms to extract data from the repository and insert data into the repository; and (f) *Storage*, maintains user data and permissions in the database.

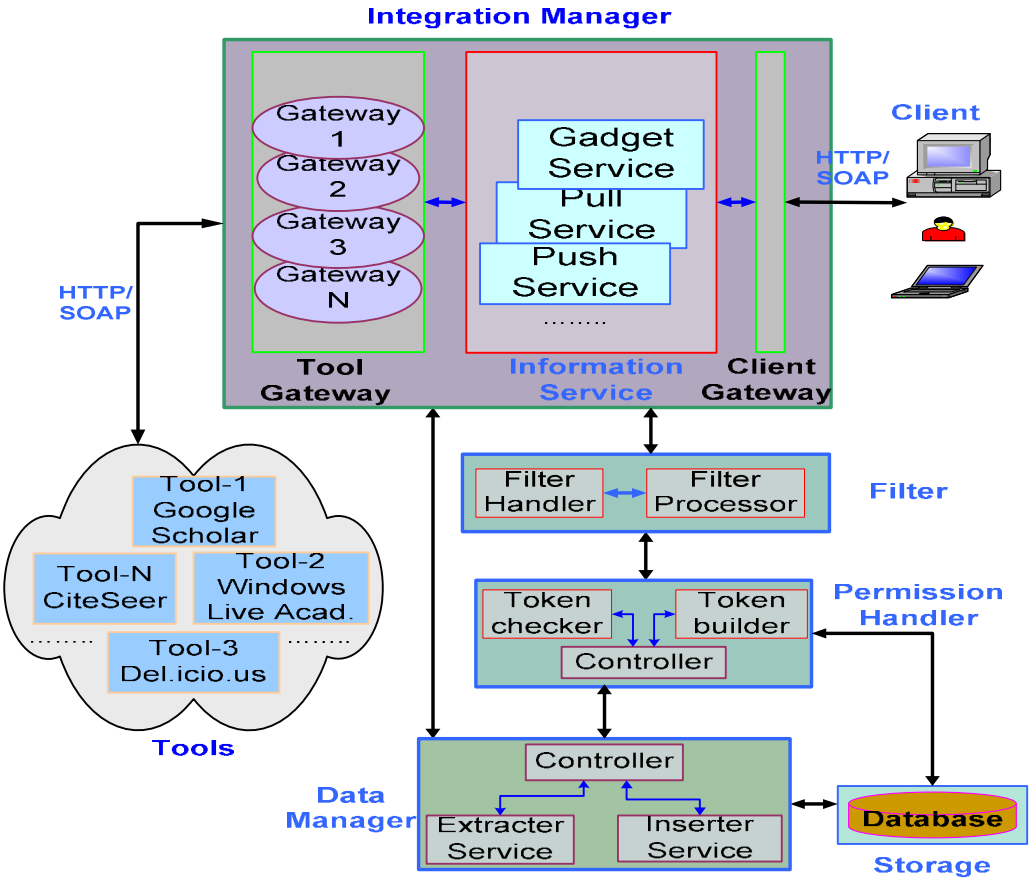


Figure 3-2 : Architecture of Integration of Collaborative Systems

We will explain the key component of the *Integration Manager*. It has two gateways and one core component called *Information Service*. *Tool Gateway* provides a channel between external web tools and *Information Services* such as request a search query or getting response from the tools. This service provides extensibility for the integrated system.

Client Gateway provides a mechanism for communication between *Client* and *Information Services*. The client has an option to select the operation in the client

interface such a search operation which is defined as *Pull Service* in the *Information Service* which gets actions such as search query from *Client Gateway*. Hence, the operation is sent to the *Information Service* to trigger required *Information Service* subcomponents such as *Pull Service*. As a result of the operation of the client request, resultant data is sent back to the *Client* from the *Information Service*.

Digital Entity (DE) is defined in our architecture as a collection of metadata fields and their data. The Figure 3-3 is depicted below shows the DE and its encapsulated metadata.

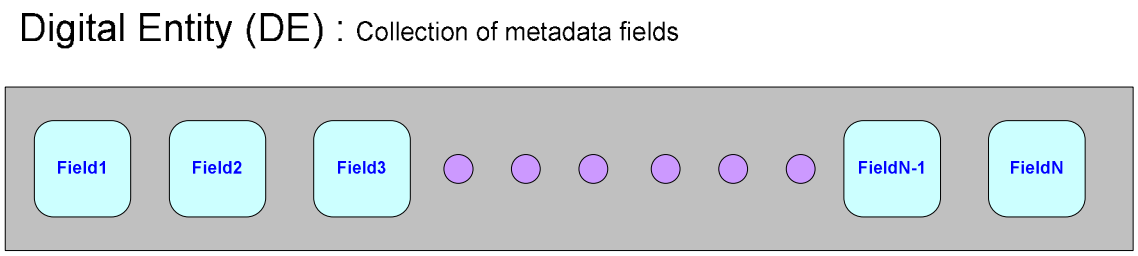


Figure 3-3 : Digital Entity (DE) content

3.1.1 Integration Manager

3.1.1.1 Pull Services

Figure 3-4 shows the *Integration Manager* and two services of the *Information Services* which are *Pull Service* and the *Presentation Service*. This figure shows also interaction of the services with gateways and the clients. *Pull Service* basically interacts with the tools using the HTTP method or the Web Services using the WSDL through *Tool Gateway* to handle the client request which is coming from the *Client Gateways*. The data communication between the tools could be any other HTTP base services having simple XML message formats or REST style Web Services. Resultant data which

may be in any format such as embedded HTML, RSS feed or any other objects. These data coming from tools send to the *Information Handler* again through *Tool Gateway*. *Information Handler* processes the incoming objects in order to extract data or metadata. *Information Handler* use different methods *Gateway* such as heuristics methods to extract data coming from the Tools. There are different ways to extract information coming from the Tools. There might not be standard methods to handle the incoming information. One possible methodology is to use heuristic method because the tool may not provide any other methods such as Web Services or API. In heuristic method, data is parsed and extracted in order to build a metadata using original incoming data hidden in different format. *Information Handler* provides extracted data required to build a new metadata. Metadata builder builds metadata elements in an XML format. This should be defined as common data format used in this Integration Systems. Each integration systems should define elements of metadata in order to have successful integration model. We could name this metadata object as Digital Entity (DE). *Presentation Service* provides an interface to display DEs whether coming from the *Web Tools* or from local integrated systems. *Clients* interact with these services to do some certain operations such as filter out the DEs or insertion of the DEs to storage or uploading DEs to other tools. *Presentation Service* has two major components: (a) *Simple presentations* show DEs in a RSS-style objects format; (b) *Detailed presentations*, shows the more metadata elements for each selected the DEs in the RSS-style summary display menu. User can also have option to use the different Information Services such as event-base[78] in order to insert the new DEs or modified the selected DEs in the display menu, and Search Services in the *Presentation Service*. We have explained the *Pull Services* and interaction with

gateways and presentation service in this section. Push, Gadgets, and Local Search Services are also components of the *Information Service*. These services can be used dynamically and if needed other services can be added to *Information Services* which provides flexibility in the *Integration Manager*.

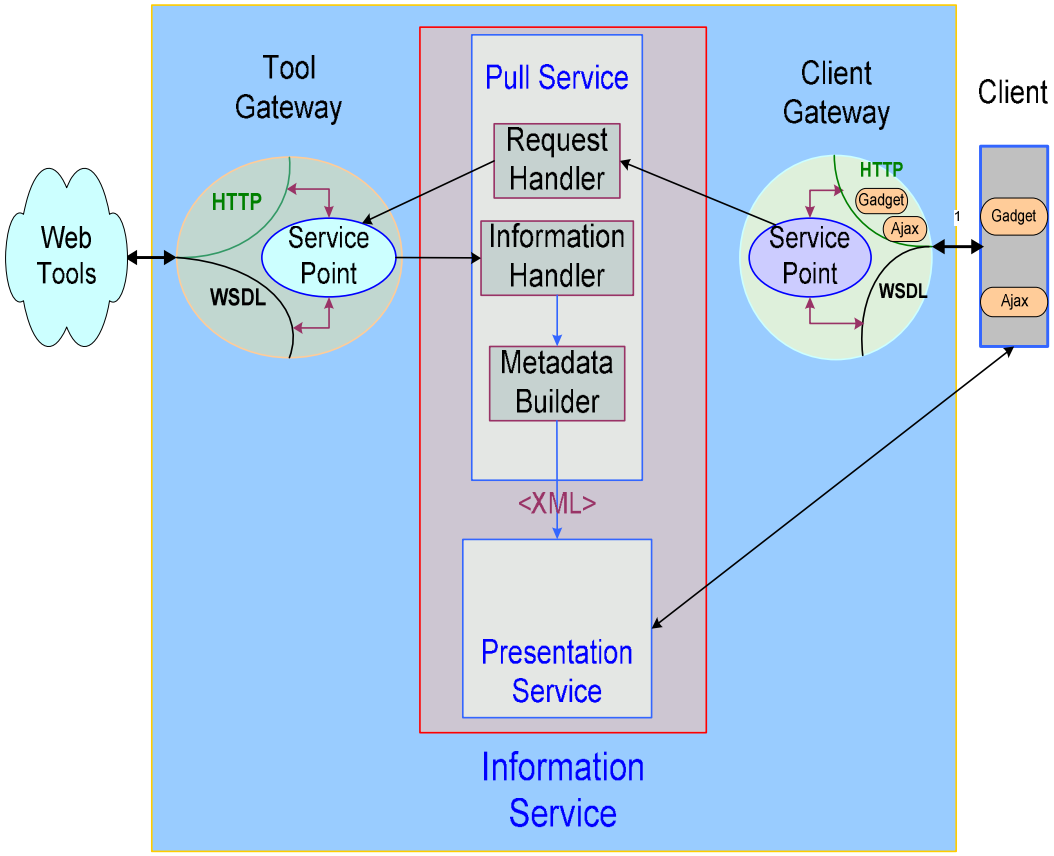


Figure 3-4 : Integration Model with Pull Service and interaction with clients and tools

3.1.1.2 Push Services

Push service accepts client requests coming from *Client Gateway* to extract information from the repository, and then upload this information to tools using *Tool*

Gateway or generate XML documents in RSS or some other XML document formats. Request Handler process the incoming request for example search a title keyword from repository. Request Handler communicates with *Data Manager Extracter Service* to get query the results. *Information Handler* gets extracted data from Request Handler and sends them to the Metadata Builder to build a new metadata. *Metadata Builder* builds metadata elements in an XML format as DEs. *Presentation Service* displays the DEs coming from *Metadata Builder*. *Clients* interact with these services to do some certain operations such as filter out the DEs or uploading DEs to the other tools. *Tool Gateway* gets the DEs coming from the *Information Handler* in order to upload them to the tools. The data communication between the tools could be any other HTTP base services having simple XML message formats or REST style Web Services. Push Service model is depicted below in Figure 3-5.

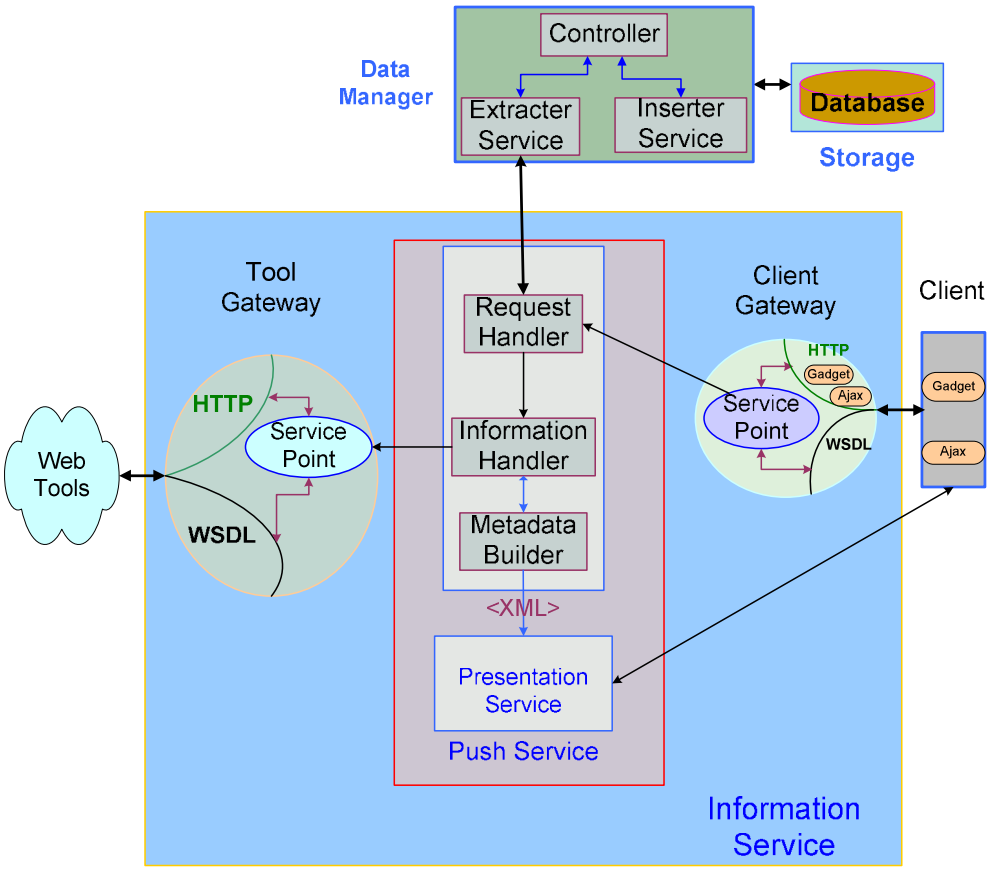


Figure 3-5 : Integration Model with Push Service and interaction with clients and tools

3.1.2 Filter

Filter provides two-way capability for reducing the number of input DEs after using selection operations. The collections of the metadata (DEs) are populated by users from the various tools after the pull or push operations. That is explained in the Integration Manager Section above detailed. Input DEs may come from local search result as well as search through web search tools or from other tools as a result of the pull or push service operations defined in the *Integration Manager*. Figure 3-6 shows the operation of two-way filter using the blue and red arrows. For each way of operation, number of input DEs is reduced by user's selection operation.

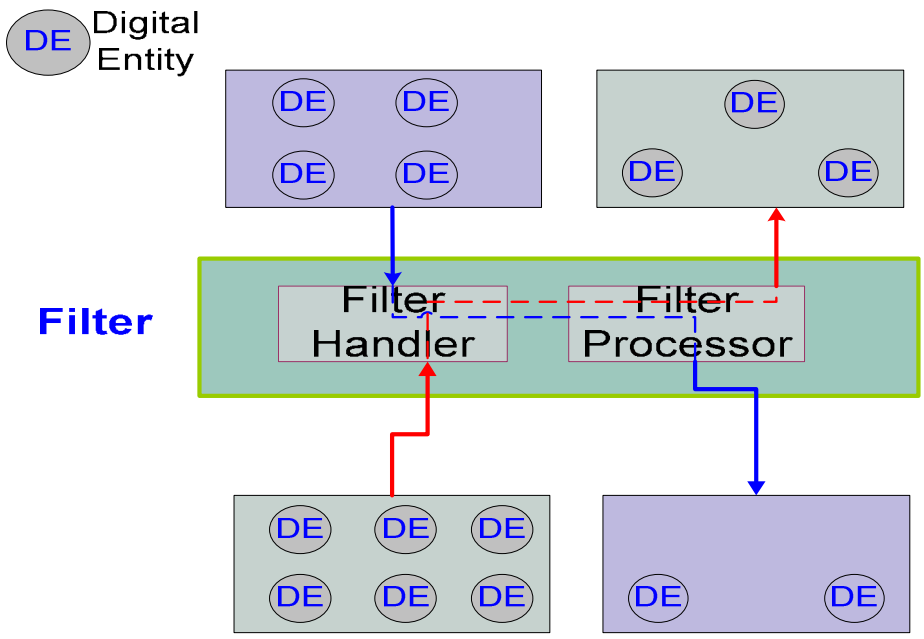


Figure 3-6: Two-way filter operations

3.1.3 Permission Handler

Current Web 2.0 tools do not have a defined clear security model. The restricted access to the resources should be defined and used to protect user community data. Otherwise, without having security model scientific communities suffer from lack of security while using the Web 2.0 tools. The model also should still allow using systems without any fine-grained security model. We defined a security model using access control matrix and roles [79]. Users have ability to define permissions such as Read, Write to grant/deny to DEs in the system. This security model can be adopted into the *Integration Model*. *Permission Handler* checks each DE to make sure that user has privilege to access DE. If a user needs to store new DE in the system, the user builds a new permission token for each DE. So, each DE will be protected from other users. A user also can build a security permission tokens for other users for the same DEs.

Therefore, users both protect their data and share them with the other user. *Permission Handler* also provides group access control for users. For example, each user associated with the group has access to all DEs of this group.

3.1.4 Data Manager

This service communicates with the storage through driver connection. *Controller* in the *Data Manager* module takes actions in order to decide whether they are data insertion or extraction of the data operations. *Inserter Service* does insertion operations of the DEs and their associate permission tokens. *Extraction Service* is responsible for getting DEs and their permissions from *Storage* using queries. For extraction or insertion operations for DEs, users should have required security tokens in order to do these operations. Otherwise, users can not access to the DEs stored in the *Storage*.

3.1.5 Storage

All the community building data metadata should be backend by storage. In the *Storage*, DEs and user and group permissions are stored. The storage can be defined as any databases or file systems.

3.1.6 Tools

The tools can be defined as the external web, annotation or other information tools which provides services for pulling metadata from their services or pushing metadata into the tool services for user or communities. Each tool has own structure or services in order to provides information to the users or pushing data into their systems,

for example, one tool may provide service API for pulling information from the service of the tool.

3.2 Fundamental Integrated Collaboration Information System Services

In this section we explain fundamental ICIS Services that provides collaborative environment for user communities. The main services are depicted below in Figure 3-7 and explained in the subsections of this section.

Figure 3-7 : Integration Collaborative Information Systems (ICIS) Services

3.2.1 Local Search Service

The repository stores information which is coming from various resources such as annotation and web search tools or any web pages or tools having valuable information.

This service provides a capability to search the resources to populate Digital Entities (DEs) from the local repository. The search results shown as RSS-like object and detailed description of metadata provided for captured DEs from repository.

3.2.2 Web Page Search Service

This service provides search methodology to search any web pages and collect metadata from these web pages. This tool extracts the information from any web pages to build DEs. Client communicates with web pages using HTTP methods. The collected information may be valuable to start building Digital Entities. Because extracted information from web pages may not have completed metadata related to the DE.

3.2.3 Web Search Tool Service

This service provides a search mechanism using Google Scholar (GS) or Windows Live Academic (WLA) web search tools using keyword. Search results will be processed by users for example if user has write permission for any folder in the system, selected the DEs can be inserted into the system in the repository. If user already has the same DE in selected database, user has option to update the DE by using event-base service will be explained in 3.2.6. The communication between ICIS and GS is established through HTTP Connection. Since there is no a RSS or an XML formatted resultant content exist for GS search, the search results are coming from GS needs to be parsed by heuristic method to extract metadata from the content that will be explained in section 5.2. However, WLA provided the RSS content for a search results. The digital

objects are parsed through the RSS files, and displayed to the users such as RSS-style format.

3.2.4 Tagging Service

Tagging provides a way of categorizing metadata by users. It is easy to discovery of new content, and support for the creation of niche communities by using the tags. This service provides a user to add a tag for metadata. User also can easily select the tags and then related document can be viewed, edited or uploaded to other tools in the system.

3.2.5 Access Control Service

This service provides capability of accesses for user to use the resources or denies the use of the resources by user. This service mechanism defined for Access Control in the CHAPTER 4 and implementation details in section 5.5. The aim of this service is to control and protect user data from other users or communities. This model is implemented in order to provide a capability to handle different types of citation metadata and store them in the repository. Moreover, the system has control mechanism for allowing multiple users and protects their metadata resources from other users.

3.2.6 Event Service

Event is defined as time-stamped action for Digital Entity (DE) in our system. It will ensure the working of the system as defined. Two types of event are defined in the system: (i) major event-insertion of new DEs into the system considered as major event; (ii) minor event-any modification on DE. Dataset is also defined as collection of the

minor events related to a user. Event Service provides major and minor event operations and allows users to build dataset using different combination of the events done by users. It also provides rollback on the changes done by users. This service is implemented by another PhD student.

3.3 Integration of a tool into Integrated Collaborative Information Systems (ICIS)

The question raised here that how to integrate a tool into the Integrated Collaborative Information Systems (ICIS)? We will address this question to explain ICIS integration steps. These steps are defined as:

1. Investigate the tools to find out whether one tool can be integrated into ICIS. The tools need to have common data or metadata. In ICIS Framework, there should defined data model before starting to integrate various tools. Next, if they have common metadata, tool can be integrated into ICIS Framework.
2. Map the tool metadata with defined ICIS metadata. Tool itself own metadata definition for Digital Entity. However, the name of the metadata might be different for the same object. We need to analyze tool metadata to map them with ICIS data correctly.
3. Investigate the tool in order to find out the available methods for data communications such as push or pull methods. Then, checks the methods they provide for extracting data or uploading information such as HTTP, API, Web services, AJAX, REST or some other available technologies. Sometimes, heuristic methodology needs to be defined and implemented to extract

information from the tools. Because the tool doesn't any technologies we mentioned. We have to decide on methodology and, choose the possible technology tool provides. Then, we can build a gateway which is specific to each tool in ICIS Framework.

4. Build a wrapper around tools based on the step 3 using Web services or Web 2.0 technologies.
5. Build necessary integration clients using client gateways in order to access and use the tools.

We have some limitations such as tools may have many features but they may not have the best performance and give full access to their resources that causes lack of performance for the tools. The integration model defined in this chapter provides flexibility that provides easy integration into ICIS or remove of the tools from ICIS. The Figure 3-8, Figure 3-9, and Figure 3-10 shows the integration steps of the tool into ICIS Framework.

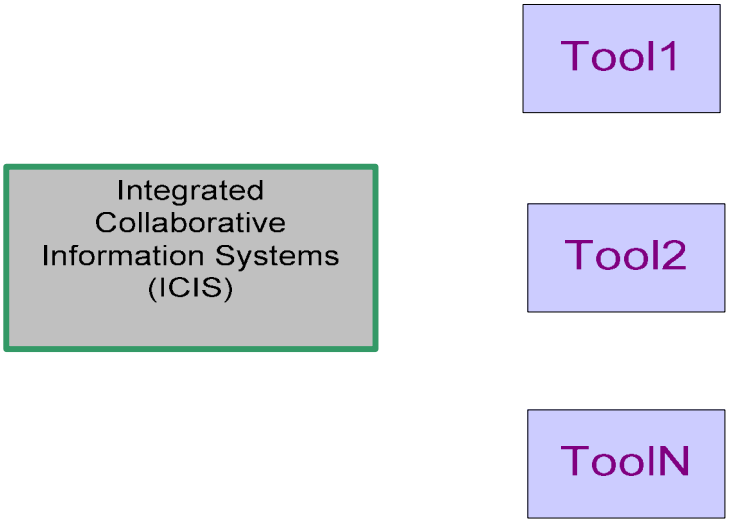


Figure 3-8 : Tools and ICIS Framework are not integrated

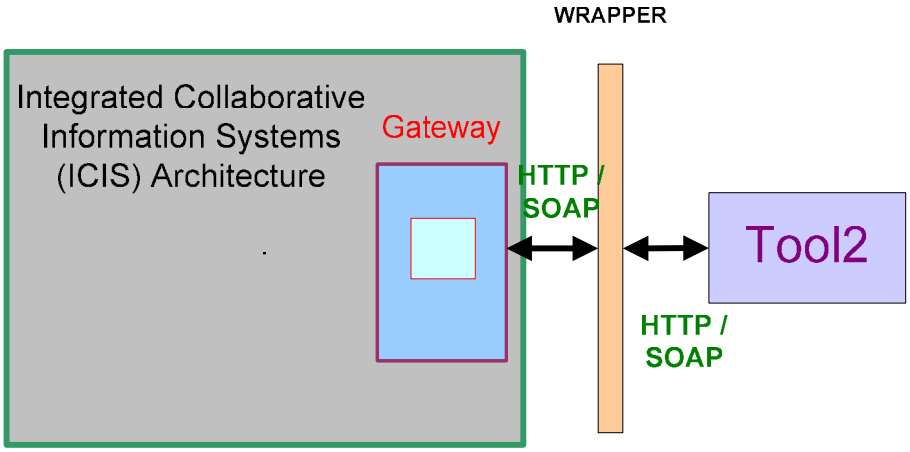


Figure 3-9 : Tool integrated into ICIS using wrapper and gateway

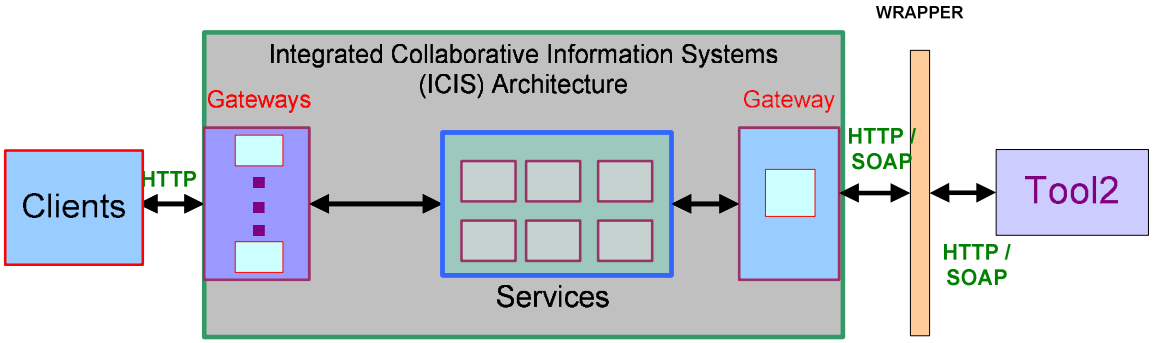


Figure 3-10 : Integrated tool, ICIS, and client interactions

CHAPTER 4

ACCESS CONTROL

In collaboration systems users may access other user's private data unless a protection mechanism is incorporated in the system [80]. We have designed and implemented an authorization module for Integrated Collaboration Information System in order to protect the databases and DOs. This module provides mechanisms for: (1) Login management; (2) Logout management; (3) Digital entity and folder(s) access rights management; (4) Administrative tools.

4.1 Users and Profiles

The system supports individual users and groups of users. Users' personal information and the login information for bookmarking web sites are accessible through the user's profile. More specifically, user's profile contains the system password, email address, full name, login information for annotation web sites (citeulike.org, connotea.org

and del.icio.us), and the group membership information. Users can access and modify their profile settings at any time; while logged in users can: (a) Change their system password; (b) Update their profile including the full name, email address and the username and password for the annotation web sites; (c) Make requests to subscribe to any available group.

.

4.2 Group Administration

Having permissions for DEs and databases for users and groups doesn't provide a complete authorization scheme. We need to have level of controls of the users and groups to complete authorization module. Administration of Authorizations is used for having flexible authorization mechanism [81]. So have defined a level of control of authorization in the system by having Super Administrator(SA) and Group Administrator(GA). The system allows having more than one SA. An existing SA can add other SAs to the system. SA can assign any user to become GA, and remove GA from group. Each group should at least one GA. if user creates a new group, users automatically become GA. Users are allowed to belong to more than one group. User can make a request to member of any group in the system. However, GA needs to confirm the request made by user. So SA controls groups and users. GA controls users in the group. The relations of the SA, GA, and user are depicted below in Figure 4-1.

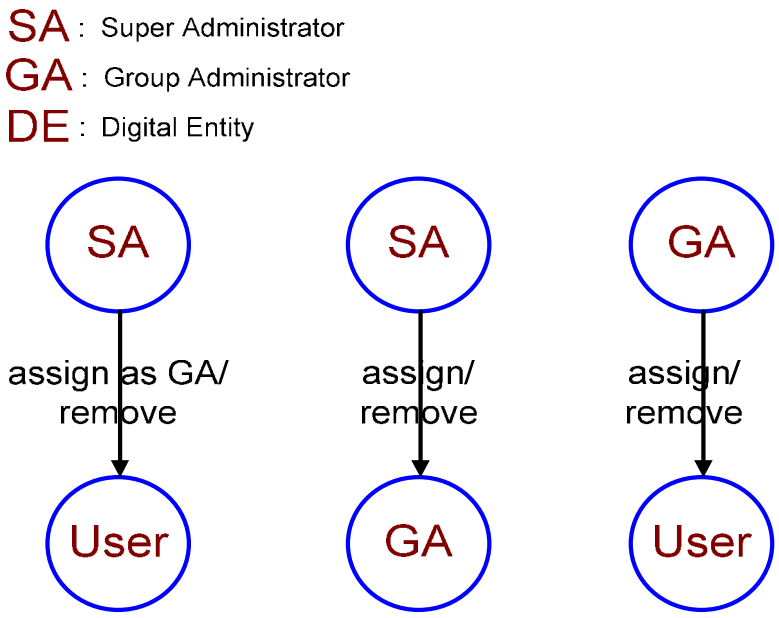


Figure 4-1 : SA, GA, and user relations

.

4.3 Access Rights

Different kind of access control methods have been previously used in various systems. These methods include the access-control matrix, the role-based access control (RBAC), and the task-based access control (TBCA) [80]. We have adopted the access-control matrix model with the addition of supporting multiple groups and multiple users for each object. The model should support for any changes for group of people. There are set of objects (DE or folder, access rights) pairs for users. Digital Entities may be created by the users of the Collaboration System in several ways: (a) using annotation tools (Delicious, CiteULike, and Connotea); (b) using search tools (GS, WLA); (c) manually, through “Insert New Citation” interface. There are two types of options defined to make these operations: (a) Public: DE creation must be associated with at least one group; (b) Private: DE can only be assessed by owner of it unless access rights are given to other

users or groups. For both public and private operation, read and write permission given to user or group for each inserted DE.

In the system, DEs stored for users and groups need to be protected from other users. The IDIOM system distinguishes between three kinds of users: *Owner* is initiator of the DE and folder creation, *Group* which is the any available groups in IDIOM system; *Other* users. The owner of DE and folder can specify the DE and folder permissions for all three kinds of users mentioned above.

We have used a model similar to UNIX file system [82]. The Unix RWX bits correspond to Read, Write, and Execute operation for each file and directory. In the system, DE corresponds to the file element and folder corresponds to the directory element. For each DE and folder, there are three types of access rights defined in the systems. Access rights, summarized in Table 1, are used in the implementation of the authorization module. Authenticated users can create databases dynamically. A user needs to have write permission for specified folder in order to put DEs into the folder. IDIOM system allows user to create user's own folder. All access control lists and permissions for authorization are defined and stored in the central storage.

Table 4-1 : Access Rights

Access Right	DE(Digital Entity)	Folder(contains collection of DEs)
Read	read DE	access to folder for viewing DEs
Write	modify DE	insert DEs into folder
Delete	delete DE	remove folder from system

The access matrix [83] describes data protection in operating system. This defines permissions that each subject holds for each object [80]. As an example in Table 2, users and groups are shown in the rows and DEs in the columns. In this example, Bob is the owner of the DE_1, DE_2, and DE_3. He has read(R), write (W) and delete (D) accesses for these DEs. $Group_1$ has write access for DE_1, and read access for DE_3.Alice is other user and has only read access to DE_2. The system allows having only one owner of a DE and folder. However, there might be more than one group for DE and database. Bob can modify DE_1, DE_2, and DE_3 user access rights or give them permissions to groups and other users.

Table 4-2 : Access Control Lists

Name	DE_1	DE_2	DE_3	User
Bob	R,W,D	R,W,D	R,W,D	Owner
$Group_1$	W		R	Group
$Group_2$	R,W	R,W	R	Group
Alice		R		Other
Henry	R,W			Other

Owner of the DEs may give access rights to other users. We have defined two methods for this operation: a) Owner can give permissions for all DEs stored in the folder. User can give permissions to any user(s) or group(s) which can be accessed by all users of that group for selected folder. However, users and groups need to have required permissions for owner's selected folder. b) Owners can modify their DE permissions for users and groups. Relations of the user and group with both DE and folder are depicted in Figure 4-2.

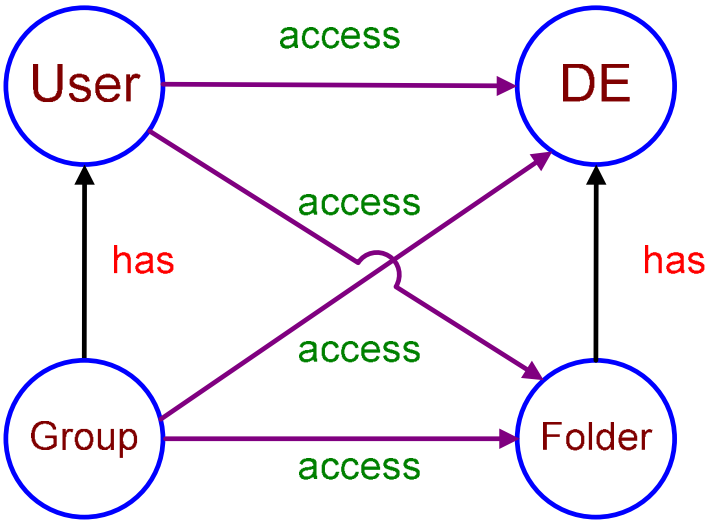


Figure 4-2 : User and Group relations with DE and folder

CHAPTER 5

PROTOTYPE IMPLEMENTATION

We have applied our proposed integration model to our prototype system called Internet Document and Integration of Metadata (IDIOM) described in detail in [79]. The IDIOM system provides a collaborative environment and it has been built based on the event-based model as explained in detail in [78]. The IDIOM system uses Web 2.0 technologies in its core services and provides extra capabilities to major existing annotation tools and search tools (Delicious[33], Connotea[3], Google Scholar and Windows Live Academic etc.). Tagging and rating are the most common capabilities in most of the Web 2.0 tools, and the IDIOM system allows its users to annotate/tag the Digital Entities (DEs) and general URIs in a flexible fashion. Users of the system are also allowed to read, to modify, to update, or to delete the DE based on their access rights. Digital Entity (DE) is represented in our implementation of the ICIS Framework as

collection of metadata that represents metadata fields of a scholarly publication. The fields of the DE in our IDIOM system are shown below in Figure 5-1.

Users of the IDIOM system have ability to share their DEs with other users or groups in the system by providing the necessary access rights. The IDIOM system consists of the following modules: (A) Session and Event Management; (B) Digital Entity Management; (C) Annotation Tools; (D) Search Tools; (E) Authentication and Authorization; (F) Other. A detailed description of the implementation of these modules may be found in [79]. The prototype of the IDIOM system can be accessed from the project demo website [84]

.

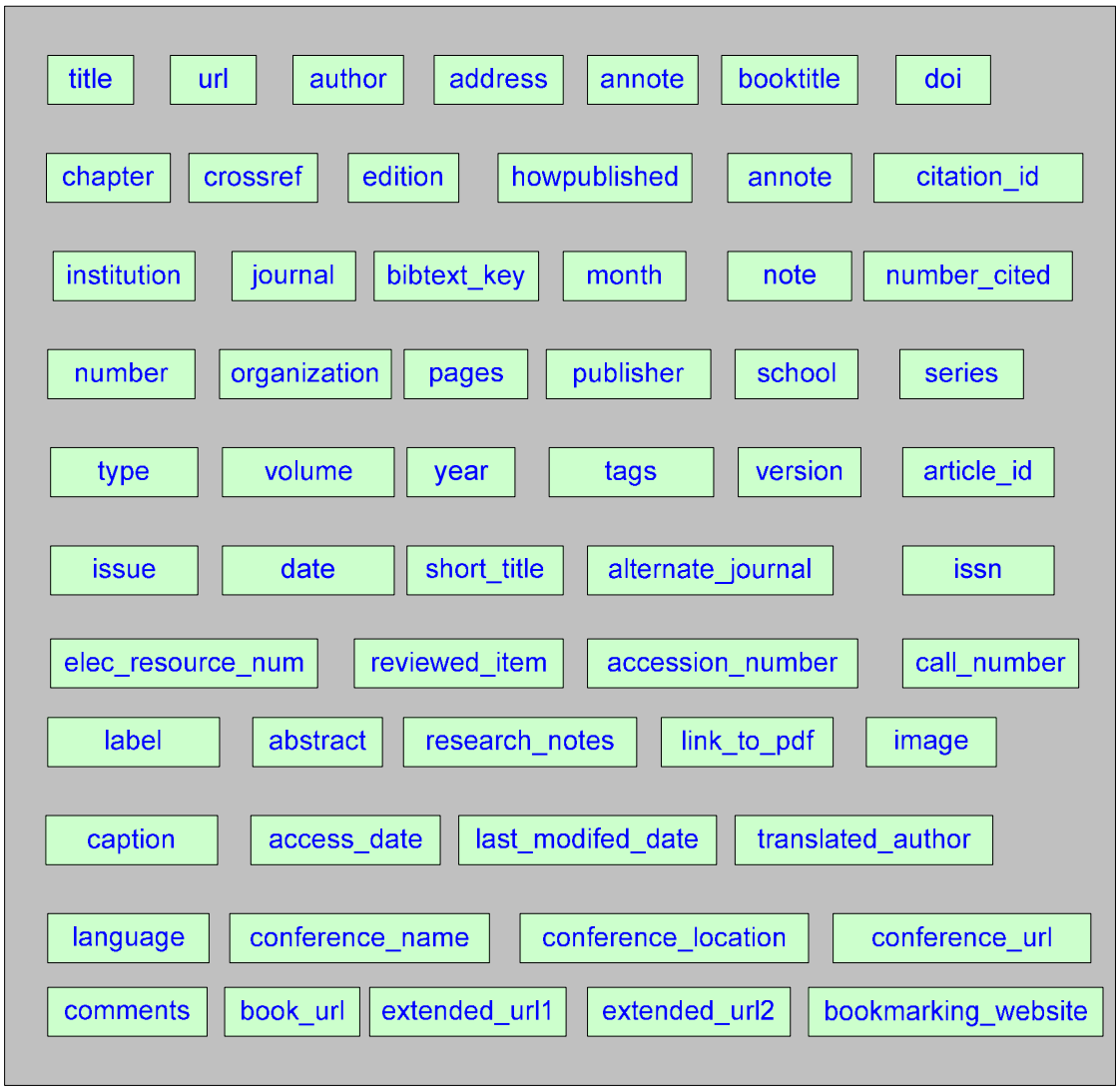


Figure 5-1 : Digital Entity (DE) metadata fields

We have followed Web 2.0 design patterns [1] in designing the IDIOM system. Below, we list these patterns and discuss how they were applied in designing IDIOM:

Delivering services, not packaged software: IDIOM provides a methodology to integrate tools and services that can be accessed over the Web (either through a user interface or programmatically through Web services). It will evolve by introducing new features; still its users won't have to install new versions of the software.

Producing hard-to-recreate data that gets richer as more people use the system: By combining data from a variety of sources, IDIOM will create added-value data and metadata generated with specific communities in mind. As more people participate in a community, the collection of the data and metadata managed by that community will increase in quantity, leading to the potential for improved precision of the automated system tools.

Harnessing collective intelligence: Through its integration with the social bookmarking tools, IDIOM can leverage data and metadata from a large number of researchers. Moreover, the system can handle both individual users and groups of users, and supports sharing and collaboration between group members.

Leveraging the long tail through customer self-service: The term "long tail" here refers to the concept formulated by Anderson [85] that non-hit products can collectively make up a market share that may exceed the relatively few current hits, bestsellers or blockbusters, provided the store or distribution channel is large enough (this business model is leveraged for example by Netflix or Amazon.com)[1]. IDIOM aims to support research communities, such as the members of a research project, a group interested in a particular chemical compound and so on, by allowing them to create system accounts and to use the community-building tools for their specific usage scenarios.

Software above the level of a single device: Currently, the IDIOM user interface runs in a browser. However, because of its layered design and the use of J2EE

[1] The term "long tail" is also used in statistics to describe certain statistical distributions.

technology (see Section 2.C), system front-ends for other devices, such as PDAs, can be developed at low cost.

In addition to these design patterns, we have followed two general principles: (a) every component is packaged as a service as long as this packaging does not imply an unacceptable performance degradation; b) if a needed capability exists and works well but is insufficient in some fashion, we try not to replace it but rather wrap it as a service so we can interact with its natural interface but easily input and output information through its service interface.

Figure 5-2 is depicted below shows the overall architecture of the IDIOM system. This system consists of three main layers: (a) the client layer; (b) the Web layer; and (c) the data layer. The client layer is made up of Java Server Pages (JSP) which is translated into Servlets by an Apache Tomcat J2EE Web container and generates dynamic content for the browser. The client layer communicates with the Web layer over the HTTP protocol through SOAP messages encapsulating WSDL-formatted objects. The Web layer consists of several Web services who handle communication with the existing online tools. The Web layer communicates with the data layer through JDBC connection. Finally, the data layer is composed of database.

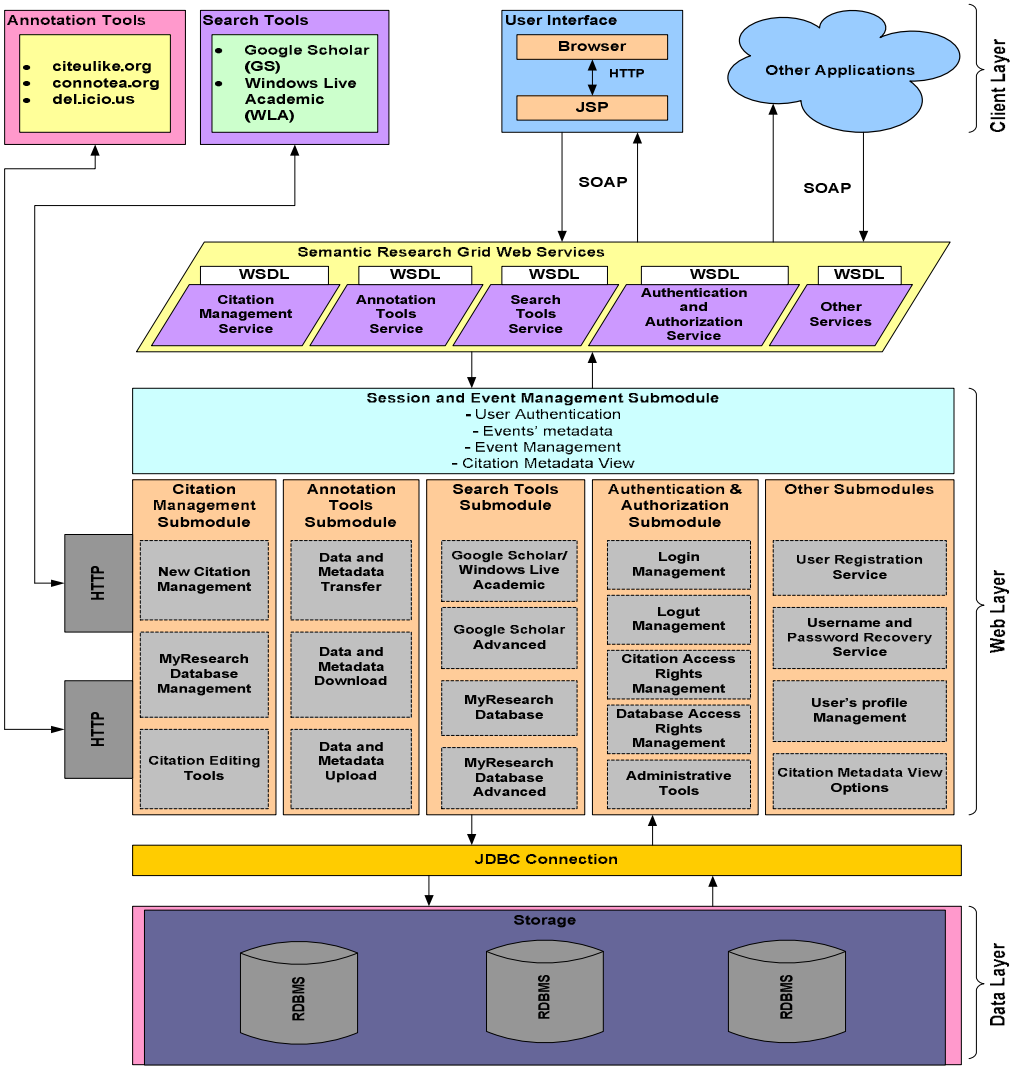


Figure 5-2 : IDIOM Project General Picture

5.1 Description of IDIOM Project Modules

The IDIOM system consists of bunch of modules. Each module has the same layered design consisting of a client layer, a Web layer, and a data layer. We discuss the technologies and software packages used in the implementation of each module:

The client layer of each module is composed of Java Server Pages (JSP). The JSP pages communicate with the Web layer over HTTP protocol through SOAP messages.

Table 5-1 : The APIs used in implementing the Web Layer

API	Purpose
JDOM	For parsing XML documents
Jakarta Commons HTTP Client	For handling HTTP communication
XPATH	For querying an XML document object
Castor	For XML-to-Java or Java-to-XML binding
JTidy	For parsing HTML documents
Apache Axis	For creating Java Web Services

The Web layer is a collection of Web services. The Web services are built using WSDL and SOAP. WSDL is a subset of XML that is used to describe the Web services and their location. SOAP is an XML-based lightweight protocol for exchanging information. The Web service provides methods for communicating with external tools. A number of APIs, summarized in Table 5-1, are used in the implementation of Web services. Web services are created using Apache Axis. The software modules are deployed in an Apache Tomcat Web container (IDIOM currently uses Tomcat version 5.0.28).

Web services communicate with the data layer using Java Database Connectivity (JDBC). The data layer is composed of several local and remote databases used for storing user specific information, such as the citation records, their access rights, datasets, and so on. Currently, we use MySQL as the Database Management System.

A detailed discussion of the implemented IDIOM modules can be found in the accompanying technical report [86].

5.2 Web Search Tools: Using Google Scholar (GS) and Windows Live Academic (WLA) to Search for Digital Entities

This part provides to make a search for any keyword using GS or WLA search tools. Search results will be populated and displayed using interface implemented in the IDIOM Project. If user has a write access for any folder/databases in the system, selected the DEs can be stored in the system for specific user or group. If user already has the same DE in selected database, user has option to update the DE by choosing edit option.

In the prototype implementation, Web Search Tools client provides an interface similar to GS and WLA native interfaces. The communication with GS established through HTTP Connection. Since there is no RSS or XML formatted resultant content exist for GS, the search result contents coming from GS needs to be parsed by heuristic method. However, WLA provided the RSS content for searched results. The digital objects are parsed through the RSS files, and displayed to the users such as RSS-style format.

In the Figure 5-3 below shows how the GS and WLA Web Tools are integrated into *Information Systems*. The figure displays the clients, services, tools levels, and relation of these levels. There are three types of clients defined in the system: GS Search Web Client, GS Advanced Search Web Client, and WLA Search Web Client. The GS and WLA clients accept any query as input parameters from any users. The GS Advanced

Search Client allows users to send a refined search queries by combining various search attributes, such as *author*, *title*, *subject area*, *and date etc*.

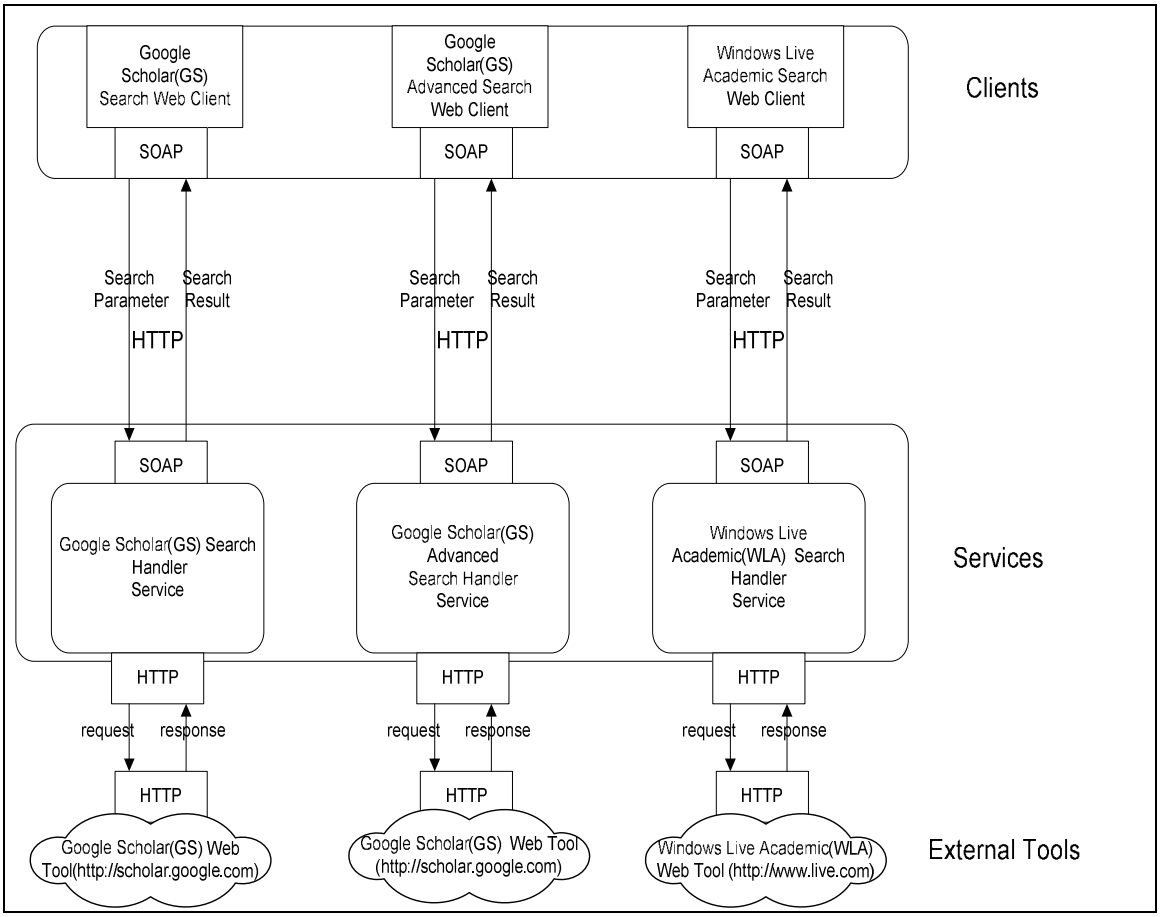


Figure 5-3 : Web Tool Search Layers for GS and WLA

The general flow of information during the search operation can be described as follows:

1. User provides a search query proving search keyword.
2. The *Search Handler Service* is initiated by the web service which is started at client level.
3. The *Search Handler Service* receives the SOAP request message from the client where query is established.

4. The *Web Service* communicates with the *Web Search Tools*. The communication between the *Web Service* and *Web Search Tools* (e.g., GS, WLA) established with HTTP using POST method.
5. The search results from *Web Search Tools* will be received using HTTP GET method.
6. The received information is processed by the Web Service. The bundle information holds embedded DEs.
7. The search results are sent back to the client using HTTP/SOAP.
8. The client display the search results in a RSS-like object format or different web view format on the screen.

The metadata coming from GS is consists of the authors, title, URL, and the number of citations of this document. WLA provides author, title, URL, publisher, and DOI. However, DOI is not provided by GS.

5.2.1 Search Handler Service

The structure of the *Search Handler Service* is depicted in Figure 5-4. The Search Handler Service has following operations that will be explained in the following sections: Web Clients, Query Handler, Parser, XML Handler, and Item Handler. Castor is used for marshaling and unmarshaling the metadata shown in Figure 5-4.

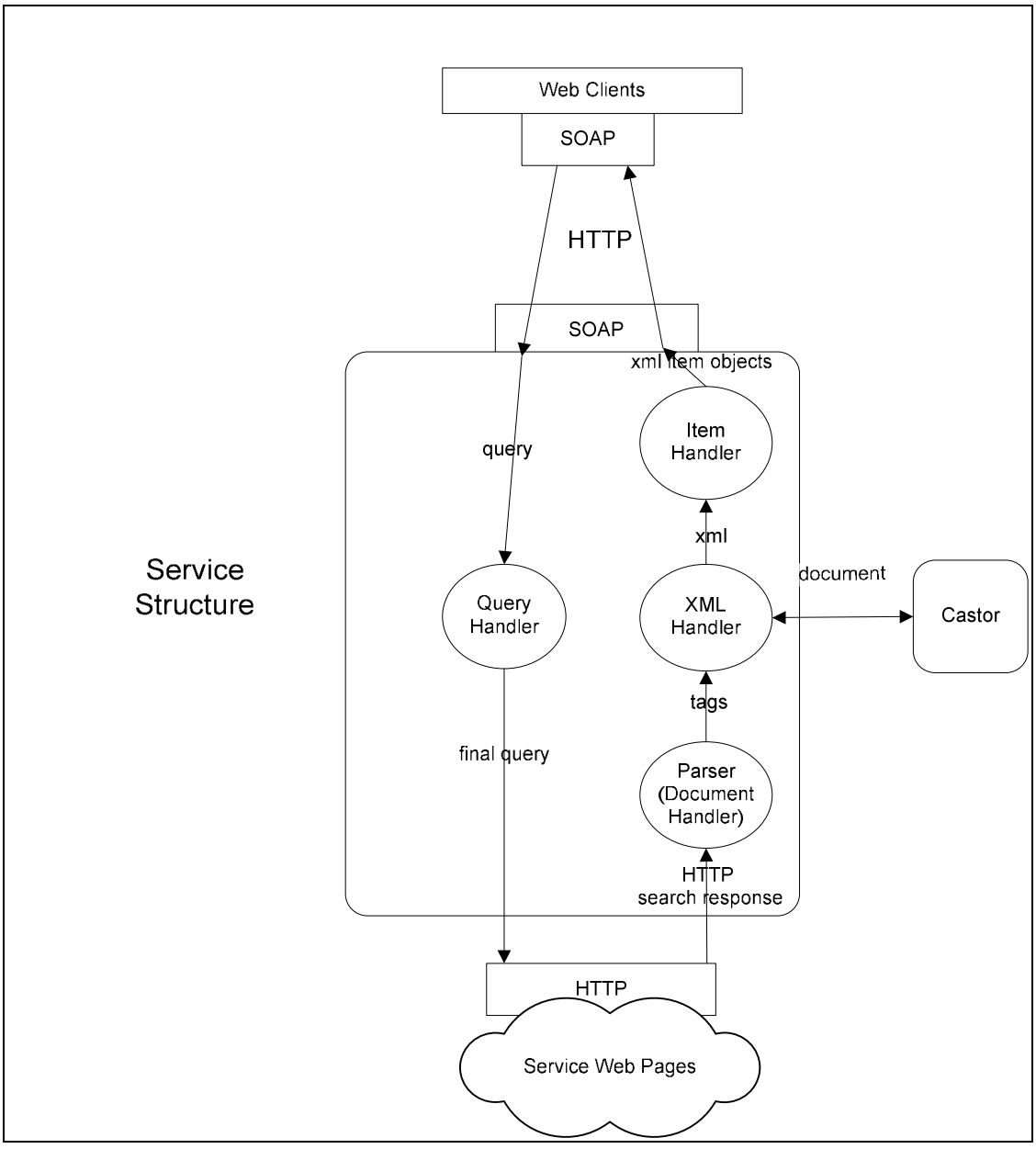


Figure 5-4 : Search Handler Service Structure

5.2.1.1 Query Handler

The *Query Handler* builds the query which will be sent to the *Web Search Tools* (the formats accepted by Google Scholar (GS) and Windows Live Academic (WLA)

differ) the appropriately formatted query will be sent to the *Search Tool* over the HTTP protocol using POST method.

5.2.1.2 Parser

The *Parser* processes the response coming from the *Web Search Tools*. It is received over HTTP as search response. This response contains metadata embedded in the HTML object. The *Parser* extracts the information to and sent them to the XML *Handler*.

5.2.1.3 XML Handler

XML Handler processes the metadata and generates Digital Entities in a XML format. These entities are generated in this step using Castor that is data-binding framework that provides the Java-to-XML binding. The XML object will be processed by the *Item Handler*.

5.2.1.4 Item Handler

Item Handler gets the XML entities having bunch of metadata. The resultant Digital Entities convey to the client using SOAP over HTTP

5.2.2 Web Search Tool: Basic and Advanced Search Interface

We need to have a mechanism for searching existing DEs in the system. User stores DEs many different ways explained in Access Control section. User need to find any DE stored in databases used in the IDIOM systems. There are two parts in the system

for searching the Local Repository: *Basic Search* and *Advanced Search*. Latter has more search query parameters to build a search query. Therefore, more refine search is possible for using *Advanced Search.*

5.2.2.1 Basic Search

The user interface has following sections: MyResearch Database part in Figure 5-5 provides a selection of the author names or titles of the Digital Entities stored in local information repository. Search query can be combined by selecting of these metadata from local storage or any keyword which is typed by user in the "Search Query" textbox shown in the Figure 5-5. Moreover, "AND" or "OR" query operation can be added to search query using the drop-down menus. However, user needs to check one of the "Add to Query" checkboxes associated with author or title respectively. The interface provides a wrapper for a search query through the Google Scholar (GS) or Windows Live Academic (WLA) for any keyword through GS and WLA. There are two web sites in the "Web site" drop down menu to make a search for query: GS or WLA. User need to specify the external tool which will be used for doing the search operation. The client interface for *Basic Search* operation is depicted in Figure 5-5.

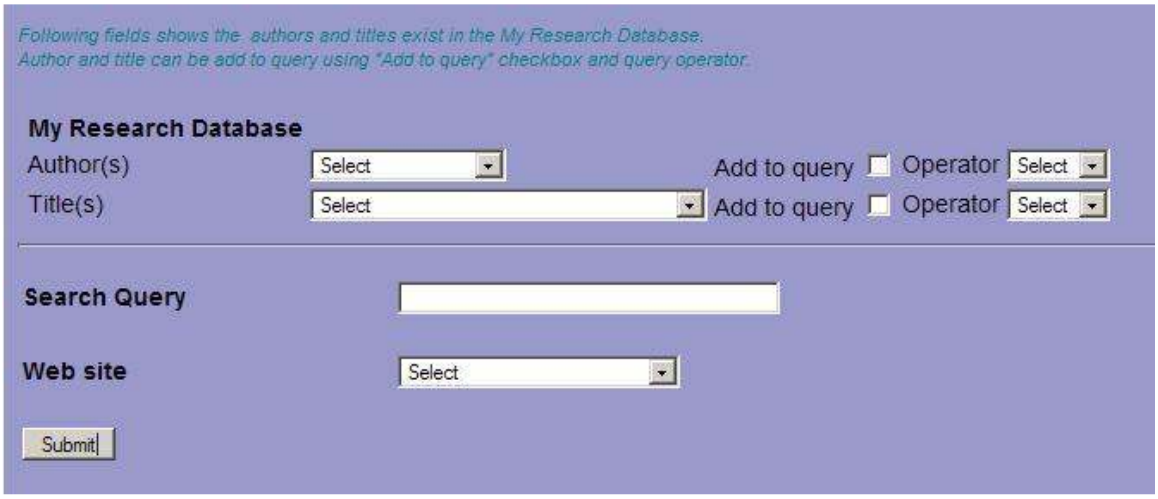


Figure 5-5 : Web Search Tools using Basic Search

5.2.2.2 Advanced Search

Advanced Search provides search capability for *GS Advanced Search*. This search provides similar interface to *GS Advanced Search* native interface. First, user provides similar parameters defined in the *GS Advanced Search* web page. They can choose any search query parameter in this page for any text fields. Then, the results are populated in the client interface. Finally, if user wants to store the results in local repository after the search operation, user needs to select the desired Digital Entities (DEs) by using the check boxes in the page. The "more info" and "edit" features defined in *GS Advanced Search* are the same as *Basic Search*. The Figure 5-6 shows the *Advanced Search* web client interface using the same search field elements of the *GS Advanced Search* parameters.

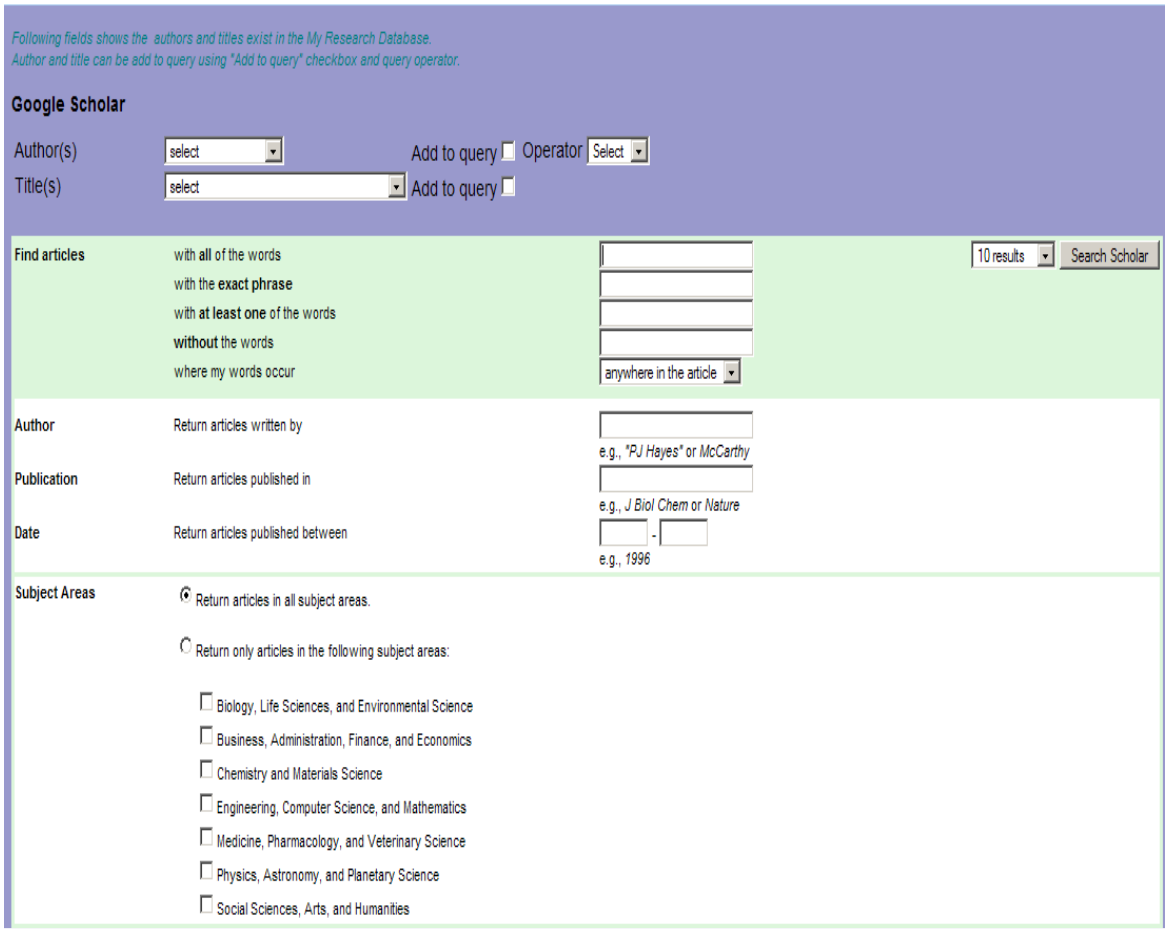


Figure 5-6 : Web Tool Search using Google Scholar Advanced Search

The difference between the *Basic Search* and *Advanced Search* is that the latter provides more metadata fields search and search subject areas as defined in native *Google Scholar Advanced Search* interface.

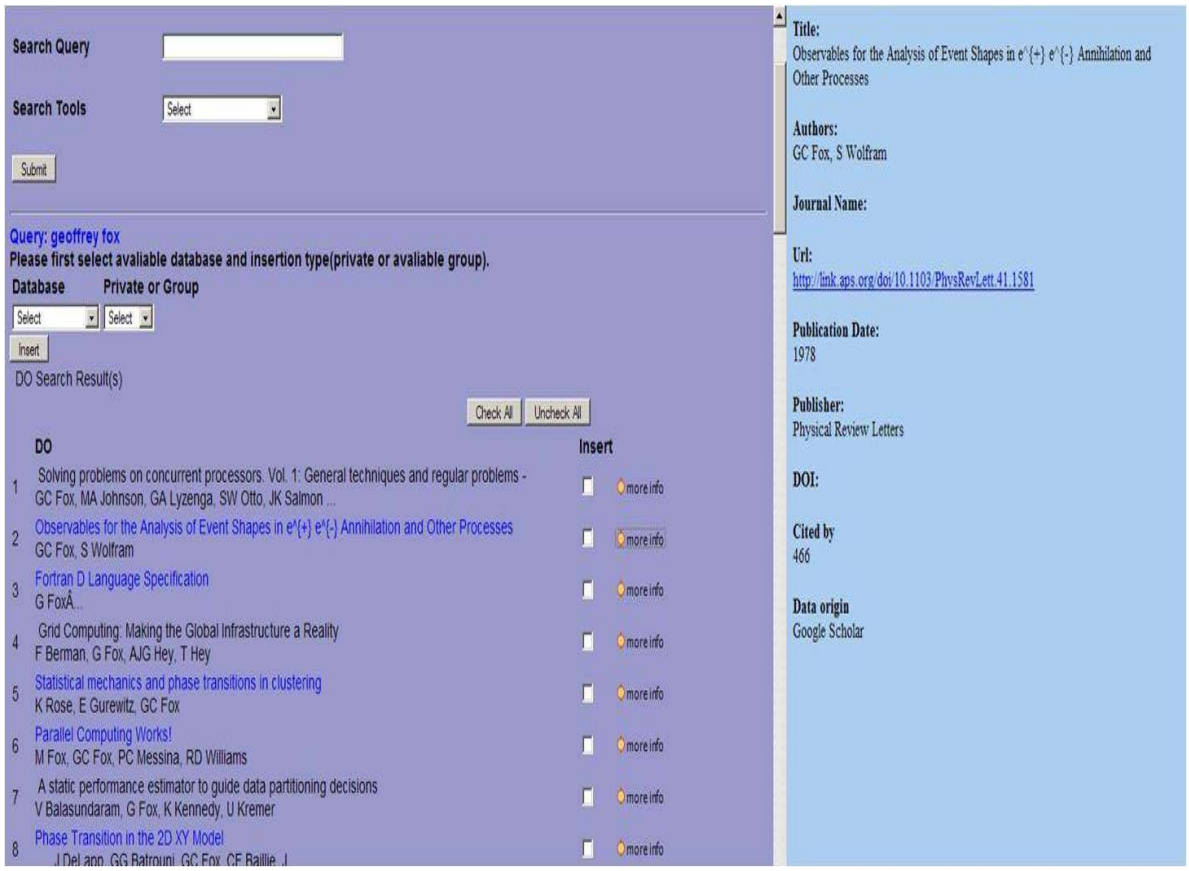


Figure 5-7 : Web Search Tools and search results

The search results are depicted in Figure 5-7 shows the query results populated in the left part of the figure. When user chooses any DEs on this page, more information about selected DE will be displayed. In the next step after having the search results, user may want to add them into private collections or group collection. In order to store them in repository, user should have selected the folder/database for insertion operation. After each insertion operation, request for insertion operation will be sent to the Event-base systems to decide whether the same DE exists in selected folder/database. If there are same DEs exists, user will be prompted having the edit option as showing in the depicted Figure 5-8. For the same DEs, user has option to modify the metadata by choosing the edit option.

Figure 5-8 : Web Search Tools after insertion request operation

5.3 Local Search Tools: Using Basic and Advanced Search Tools to search Information Service Storage

The distributed information coming from various resources such as annotation tools, web search tools or any web pages having valuable information is stored centralized local repository to provide mechanism to update information, push them to other services or tools and share with other users in defined service model. In order to use

these services initially we need to provide a search model to populate searched DEs and use them for some services mentioned earlier.

Having that *Local Search Tools* provides a search mechanism to populate Digital Entities (DEs) from local repository. The search results shown as RSS-like object and detailed metadata provided for captured DEs from the repository. The search query built when user enter any search metadata query. User may use “AND” or “OR” Boolean operation to build complex query. There are two sections for *MyResearch Database Search*: *Basic Local Search* and *Advanced Local Search*. The architecture is the same for both parts. *Advanced Local Search* has more search parameters to build a query. So, more refine search is possible using *Advanced Local Search Tool*.

5.3.1 Basic Local Search

Basic Search interface searches the Local Repository by using the author and title search query parameters. For author, user may use "AND” or “OR" operators to make a complex query. Users may use this operators using author’s full name in the double quotes (") in order to refine their search. If the user enters both title and author in the search query, the query gets the "AND" operation using the provided title and author by user. User also needs to select the folder/database name to specify the search repository using the collection drop-down menu. The Basic Search interface is shown in the following Figure 5-9.

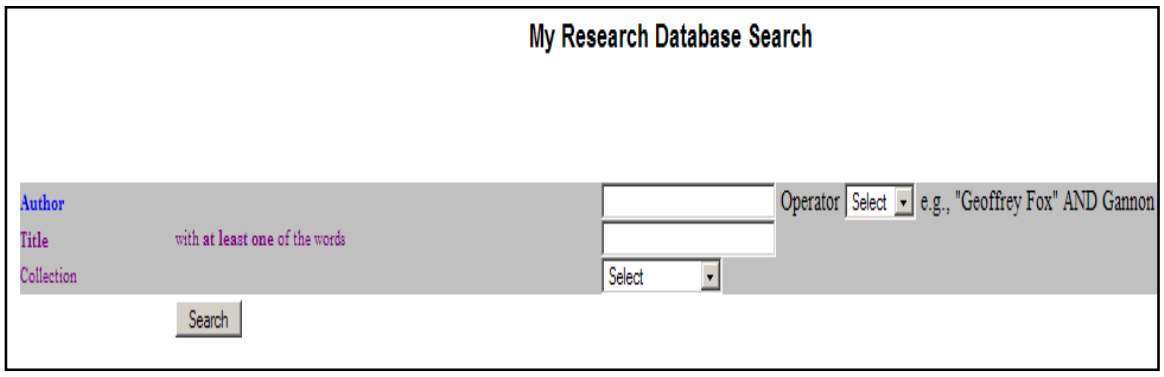


Figure 5-9 : Local Basic Search interface

5.3.2 Advanced Local Search

Advanced Local Search provides users to make a further refine search using publication name, data, and URL metadata besides author and title. User may use "AND" or "OR" boolean operators to make more detailed search. User may also use the double quotes (") within search query to specify the search query. If user enters any other search parameters such as title, publication, and date, final query will be built based on these entered metadata parameters. User also needs to select the Local Repository name to specify the search repository using collection drop-down menu. The Advanced Local Search Interface is depicted in the following Figure 5-10.

Figure 5-10 : Advanced Local Search Interface

The search results are displayed in the Result(s) section having link for each citation. If users want to see more metadata, they need to click on the "more info" button. The detailed information about metadata will be shown on the right pane of the search page. The search results are depicted in Figure 5-11.

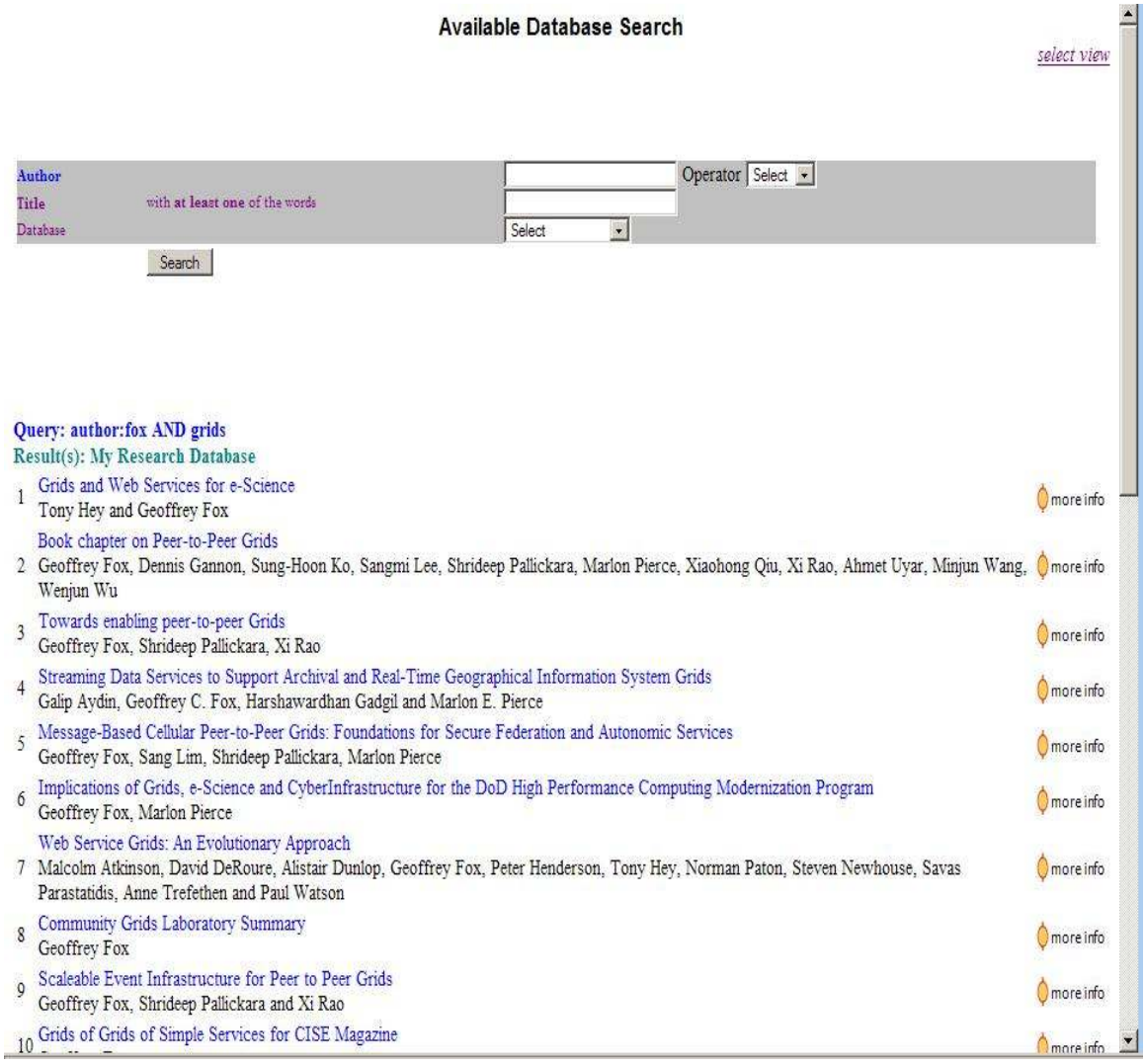


Figure 5-11 : Result of the Search Query

5.4 Web Page Metadata Collection Service

In addition to collection of the Digital Entities from various resources such as Web Search Tools (Google Scholar and Windows Live Academic etc.) or Annotation Tools (Delicious, CiteULike, and Connotea), In IDIOM project we have defined a search methodology to search any web pages and collect metadata from these web pages. This tool extracts the information from any web pages to build DEs. Client communicates with the web pages using HTTP methods.

There are two heuristic methods are implemented: Link Base and CGL Base. They provide methodology to extract data from web pages using the selected method.

1) Link Base (gets DEs by extracting data from the links using <a> html tags. It extracts any valuable links from any valid for any web pages.

2) CGL Base (gets DEs by Community Grids Labs publication web page html structure. This methodology can be applied to all web pages having same CGL publication HTML structure). In CGL Base method, authors also will be extracted from web pages and displayed in the main page. This tool is depicted in

Figure 5-12 also a mechanism to build your own selected metadata tags based on the information provided in the web page. Users can select up to the five tags that can be added by selecting information related to the metadata. They are also editable if user wants to change it.

All the web page search results as DEs are in format of the RSS/Atom feeds. All the successful results are listed as feeders automatically. They can be added to browser or imported to any web platforms using these technologies.

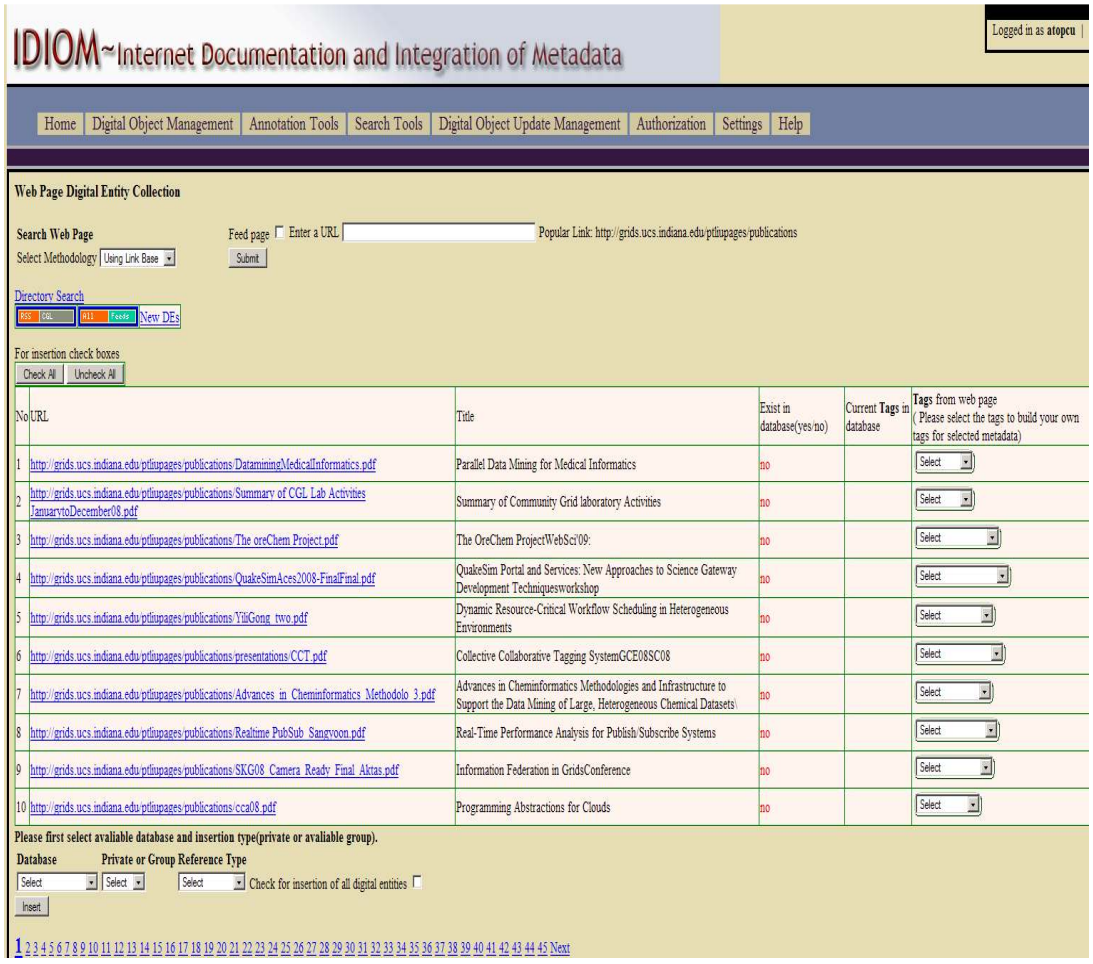
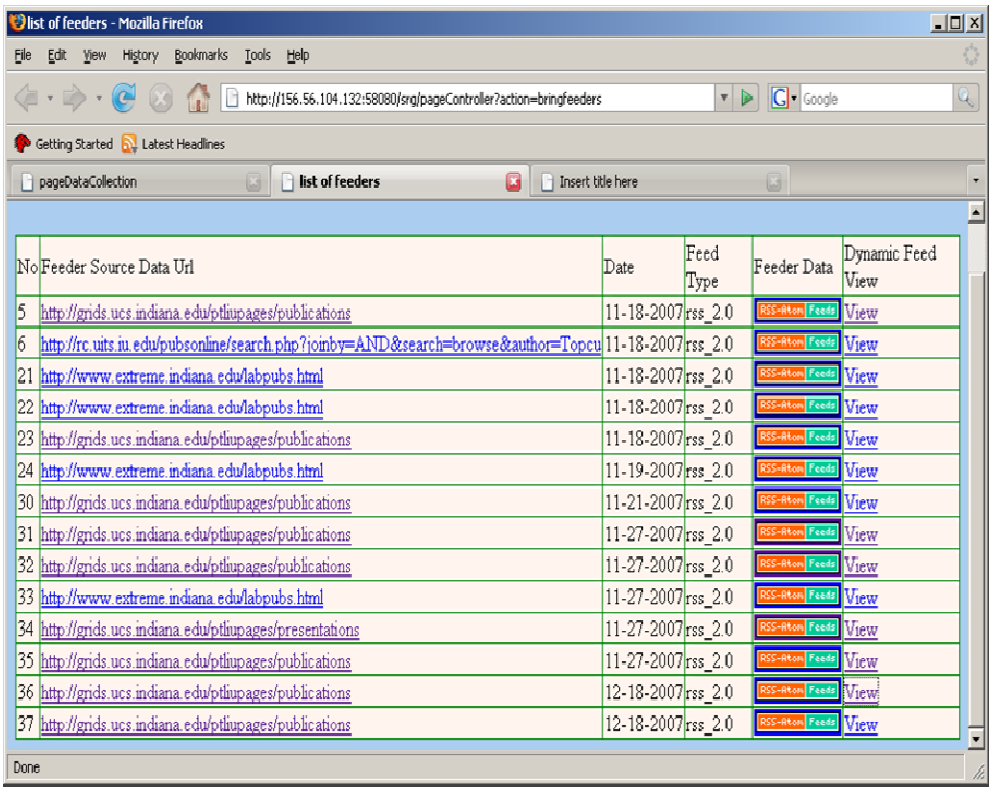


Figure 5-12 : Web page Collection of Metadata

User need to enter any URL in Figure 5-12 to build DEs from web page and select the methodology from the menu. After the submission of the search request, resultant DEs will be populated in the client page. Users have option to see all the feeds generated automatically by selecting the "All Feeds" image icon the client page. User need to click on the "RSS/Atom" image icon to see the feeders. Therefore, selected page will display the DEs in a RSS/Atom feeder format. Dates, links and RSS types are displayed for each search operation is depicted in Figure 5-13 in the new window.



No	Feeder Source Data Url	Date	Feed Type	Feeder Data	Dynamic Feed View
5	http://grids.ucs.indiana.edu/ptliupages/publications	11-18-2007	rss_2.0	RSS-Atom Feeds	View
6	http://rc.uits.iu.edu/pubsonline/search.php?joinby=AND&search=browse&author=Topcu	11-18-2007	rss_2.0	RSS-Atom Feeds	View
21	http://www.extreme.indiana.edu/labpubs.html	11-18-2007	rss_2.0	RSS-Atom Feeds	View
22	http://www.extreme.indiana.edu/labpubs.html	11-18-2007	rss_2.0	RSS-Atom Feeds	View
23	http://grids.ucs.indiana.edu/ptliupages/publications	11-18-2007	rss_2.0	RSS-Atom Feeds	View
24	http://www.extreme.indiana.edu/labpubs.html	11-19-2007	rss_2.0	RSS-Atom Feeds	View
30	http://grids.ucs.indiana.edu/ptliupages/publications	11-21-2007	rss_2.0	RSS-Atom Feeds	View
31	http://grids.ucs.indiana.edu/ptliupages/publications	11-27-2007	rss_2.0	RSS-Atom Feeds	View
32	http://grids.ucs.indiana.edu/ptliupages/publications	11-27-2007	rss_2.0	RSS-Atom Feeds	View
33	http://www.extreme.indiana.edu/labpubs.html	11-27-2007	rss_2.0	RSS-Atom Feeds	View
34	http://grids.ucs.indiana.edu/ptliupages/presentations	11-27-2007	rss_2.0	RSS-Atom Feeds	View
35	http://grids.ucs.indiana.edu/ptliupages/publications	11-27-2007	rss_2.0	RSS-Atom Feeds	View
36	http://grids.ucs.indiana.edu/ptliupages/publications	12-18-2007	rss_2.0	RSS-Atom Feeds	View
37	http://grids.ucs.indiana.edu/ptliupages/publications	12-18-2007	rss_2.0	RSS-Atom Feeds	View

Figure 5-13 : Feeders list

In the next Figure 5-14 and Figure 5-15 displays the populated DEs in RSS XML format and bookmark format. These different display formats provides flexibility to reach and use the DEs in different systems. For example user and communities use and extract the information using RSS or atom feeds.

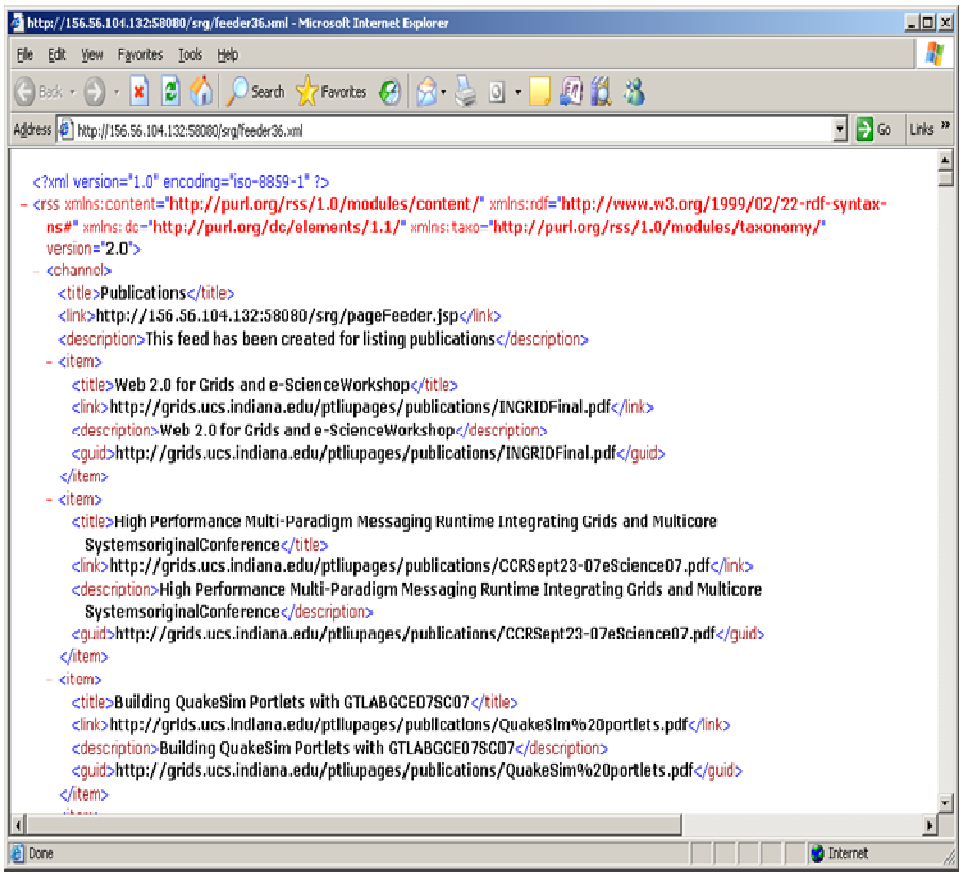


Figure 5-14 : DEs in RSS format

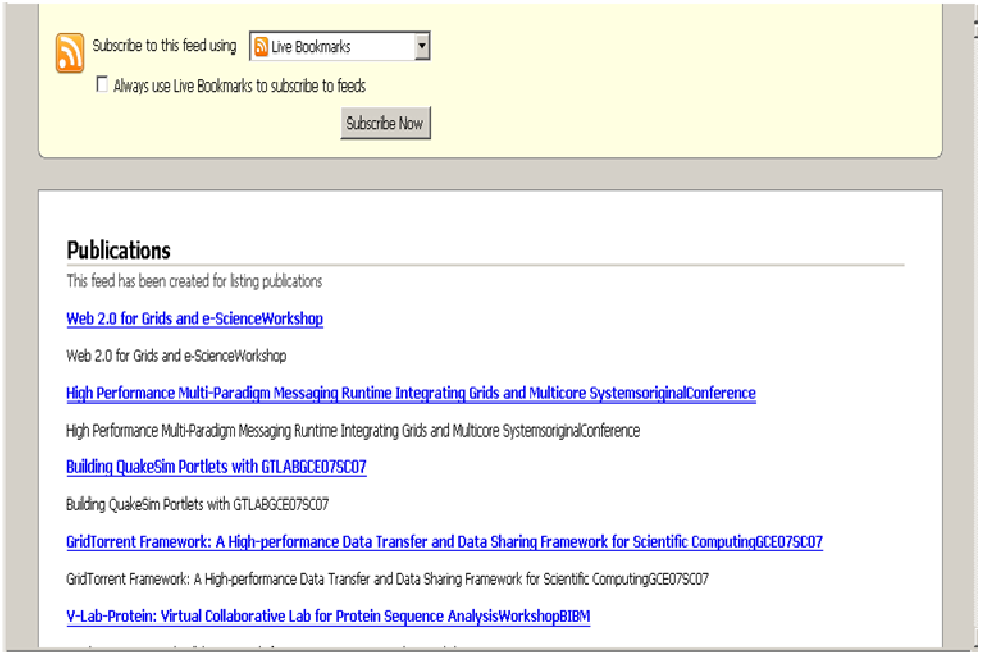


Figure 5-15 : DEs using live bookmarks

5.5 Access Control Service

The Access Control is the ability to give access to user for using resources or deny the use of the resources by user. This service mechanism defined in the Access Control in the CHAPTER 4 above. The aim of this service is to control and protect user data from other users or communities. IDIOM has implemented this model to provide capability to handle different types of citation metadata and store them in the repository. Moreover, the system has control mechanism for allowing multiple users and protects their metadata resources from other users. So, Access Control Service was defined and implemented for having data protection and sharing documents in the secure environment. Also, system provides a control for both group and private data. For example, user can use own group for keeping the important document which user doesn't want to share them with other groups or users.

Each DE record has three types of privileges; *read*, *write*, and *delete* permissions. For each DE, there is only one owner exist associated with the DE. There might be many groups or many other (users) may access to the DEs.

5.5.1 Service for Digital Entities (DEs)

The user interface depicted below in Figure 5-16 for user's DEs stored in the repository populated in the .The Figure 5-16 displays the DEs in the rows and they are associated with the owner user information and access rights of the DEs, and possible group, and other (user) user information and their rights. Functionalities can be summarized as:

- Users can make a request to subscribe to a specific group to use collaboration in the system.
- Users can be part in other users to access for the DEs.
- Users can be discarded from the group by group administrator.
- One user may join the multiple groups.
- Users can have private (own) DEs. User has control to share data with other users or groups
- Users can give permission to groups or other users or restrict the access rights for their own content.
- If user initiates any DEs creation on the repository; he/she will be owner of the DEs.
- User can modify access right for specific groups and users for selected collections.

Figure 5-16 : Managing Access Rights

5.5.2 Service for Folder (Container for DEs)

This service provides capability for creating or modifying the permissions of all DEs exists in the folder (database) in the system. This feature incorporates all the users in the group and selected others (users).Figure 5-17 and Figure 5-18 below displays the functionality of the service. User first need to select the folder/database to give permissions for selected the group and other user respectively from the drop down menu. This provides control of the all DEs retain in the specific folder container.

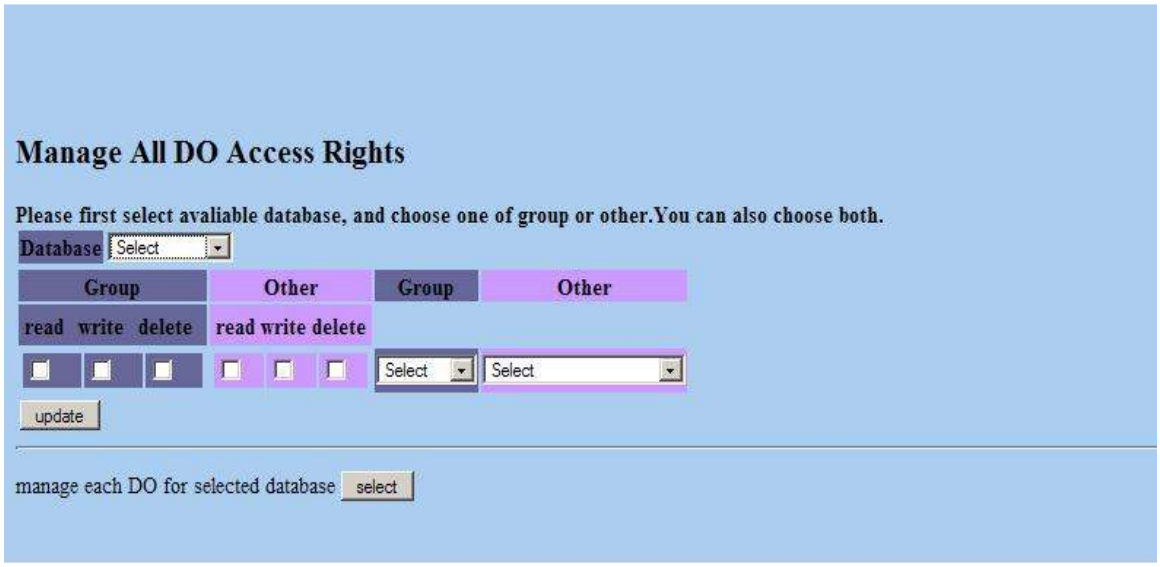


Figure 5-17 : Managing all DEs in folder/database

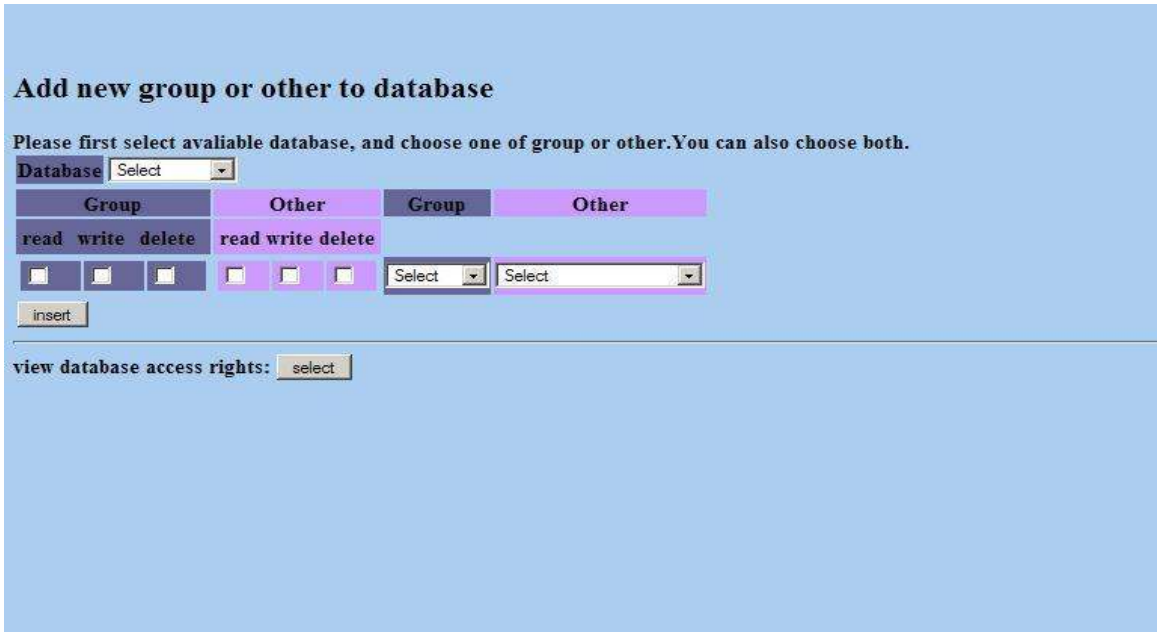


Figure 5-18 : Adding new group or others (users) to folder/database

5.6 Level of Authorization

In IDIOM project we defined authorization scheme for protecting user data from other users. Also, level of control for user and group needed to provide control mechanism providing hierarchy in the system. Hence, as mentioned in the Group Administration section in 4.2, the level of authorization defined having Super Administrator (SA) to control the Group Administrators (GAs) who manage the users in the specific group.

5.6.1 Super Administrator Control Model

Super Administrator is responsible from the Group Administrator, users, and groups in the IDIOM project. There might be more than one SA to eliminate the dependency on one SA. Super Administrator can assign any user to become Group Administrator of any group defined in the system. They can remove any user from the GA list. Hence, user no longer becomes GA in the system. Super Administrator can create a new group; assigns a user to new GA from user list; select a new user to become SA in the system. The module implementation of the Super Administrator service is depicted in Figure 5-19.

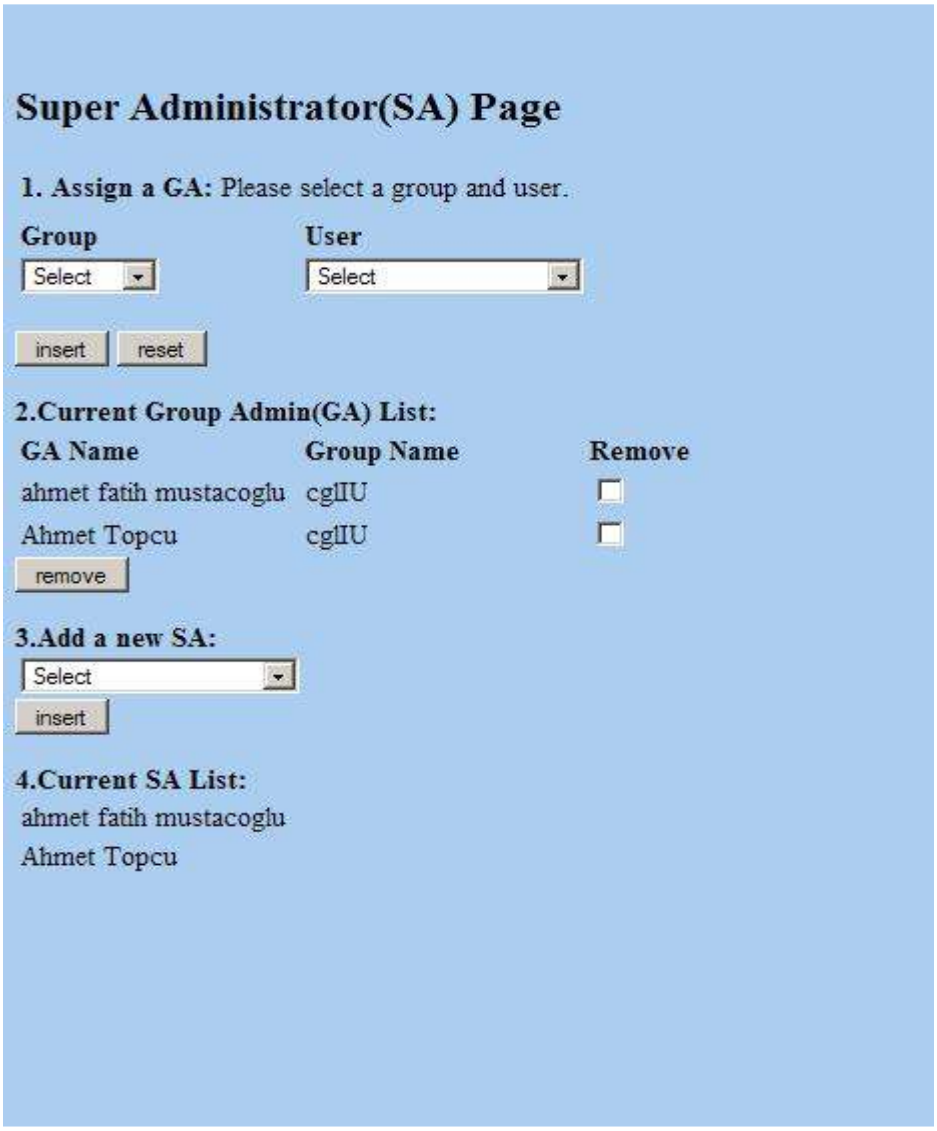


Figure 5-19 : Super Administrator (SA)

5.6.2 Group Administrator Control Model

The IDIOM system provides a user to make a request to join the specific group. However, Group Administrators need to approve this request. For each request Group Administrator approves/denies any user request for the specific group. Current GA list also be displayed on this section. We can summarize the functionalities:

- GA can assign users to group
- GA can remove users from the group
- GA can modify the group access rights.
- There might be more than one GA for each group.
- Private content doesn't controlled by the GA. Only users can control their private contents in the system.

The module implementation of the GA service is depicted in Figure 5-20. In this figure shown below interface has three parts: (1) User group request part, user request displayed for each group and GA has option to add user ;(2) Add user to group, GA may add user to specific group; (3) Current user group list, All the users and groups displayed respectively.GA also has authorization to remove the selected user from the group.

Figure 5-20 : Group Administrator (GA)

CHAPTER 6

PROTOTYPE EVALUATION

In this chapter we present evaluation of the prototype implementation using series of measurements and its practical usefulness in the applications. In this chapter we have addressed following research questions:

- What is the baseline performance of the Integrated Collaboration Information Systems implementation based on the search operations?
- How does the system behavior when the message rate/number of users per second is increased for search operation using Database access?
- How does the system behavior when the message rate/number of users per second is increased for search operation using memory utilization?
- How does the system behavior when search operation through Web Search Tools?

- How does the system behavior when access right checking operation using database access?
- How does the system behavior when access right checking operation using memory utilization?

This section includes description of the testing environment and approach and concludes key features of the measurements.

6.1 Experimental Setup Environment

We tested the Integrated Collaboration Information System framework implementation with various client programs by sending queries and servers having services for client programs. We have used gridfarm (gf) Linux machines which are part of the Indiana University Cluster located at Community Grids Labs. We have run our client programs on gf12-gf15 Linux machines, and deployed and used Integrated Collaboration Information System services on gf16 Linux machine. The following tables respectively summarize the configuration of these Linux machines are used during the testing of the Integrated Collaboration Information System Framework.

Table 6-1 : Summary of the cluster node for services

Cluster Node Configuration (gf16.ucs.indiana.edu)	
Processor	Intel® XeonTM CPU (E5345 2.33GHz)
RAM	8 GB
Network Bandwidth	1 Gbits /sec
OS	GNU/Linux 2.6.9-5.ELsmp #1 SMP

Table 6-2 : Summary of the cluster nodes for clients

Cluster Node Configuration (gf12-15.ucs.indiana.edu)	
Processor	Intel® XeonTM CPU (E5345 2.33GHz)
RAM	8 GB (each node)
Network Bandwidth	1 Gbits /sec
OS	GNU/Linux 2.6.9-5.ELsmp #1 SMP

We have implemented test codes using Java 2 Standard Edition compiler with Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_12-b04). In the evaluation of the prototype implementation Tomcat Apache Server with version of 5.0.28 and Axis software with version 1.2 are used as a container. In the Tomcat Apache Server, the Server represents the whole container and the Tomcat Server handles multiple threads to have concurrent requests. In order to handle the number of concurrent clients, the maximum number of threads in Tomcat Apache Server is increased to 1000. So, the system behavior for high number of concurrent clients will be tested. We also set Java Virtual Machine (JVM) Heap Size to 1024MB by using option command –Xmx1024m.

6.2 Responsiveness Experiments

In this section the results of the tests are presented and evaluated for prototype implementation. The main goal in doing these experiments is to measure the baseline performance of the Integrated Collaboration Information System Framework. The

primary interest is to evaluate responsiveness experiment in order to investigate the optimum performance of the systems. Prototype implementation and evaluation of the test results is based on the experiment data which is obtained during test procedures. The performance evaluation of the proposed system is done by measuring the response time of the operations. We have tested the performance of the inserting new access control tokens for the Digital Entity(DE), updating the user access control permission for specific DE, and selection operation for access control tokens from repository and memory respectively. We have investigated the latency values for those operations by database access and utilization of the memory. Hence, we compare the results and evaluate the response time for each access control update, insert, and select operations defined in the prototype system.

In these test cases we conducted following test cases:

- A client sends a request to insert a new access control permissions (read, write, delete) for specific DE into the repository.
- A client sends a request to modify the access control permissions (read, write, delete) for specific Digital Entity (DE) from database.
- A client sends a request to modify the access control permissions (read, write, delete) for specific DE from the cache while accessing memory.
- A client sends a request to select the access control permissions (read, write, delete) based on the user information for specific DE from database.
- A client sends a request to select the access control permissions (read, write, delete) based on the user information for specific DE from the cache while accessing memory.

The service-enabled Integrated Collaboration Information System was running on a cluster node gf16 while the client test programs were running on cluster nodes gf12-gf15.For each test case, client sends 500 sequential request for select, update, insert access control operations. As a test results we have recorded average response time, and this experiment was repeated five times to have better evaluation for the test results. These experiment designs are depicted in Figure 6-1.

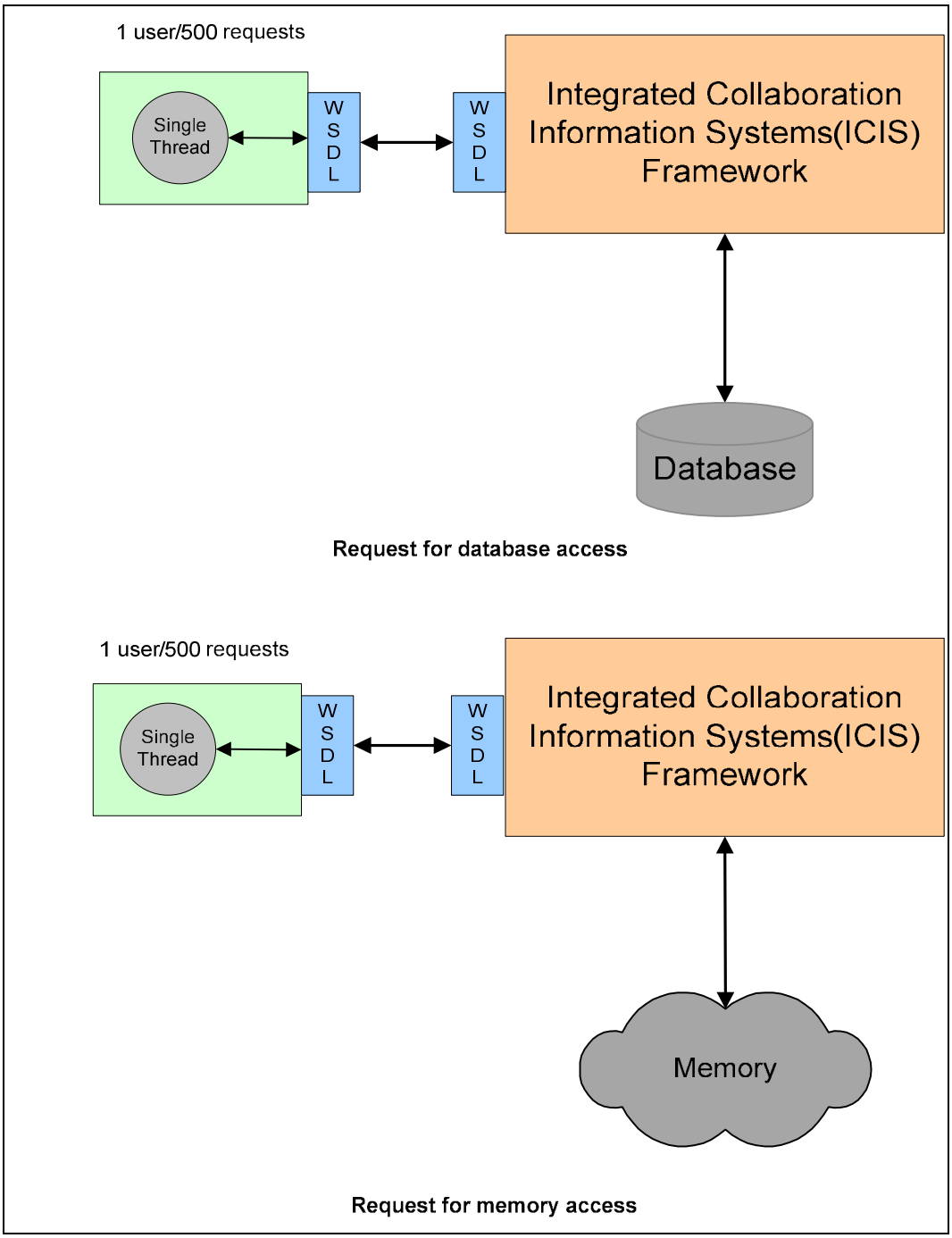


Figure 6-1 : System Responsiveness Experiment Testing Cases

6.2.1 Responsiveness Experiments Results

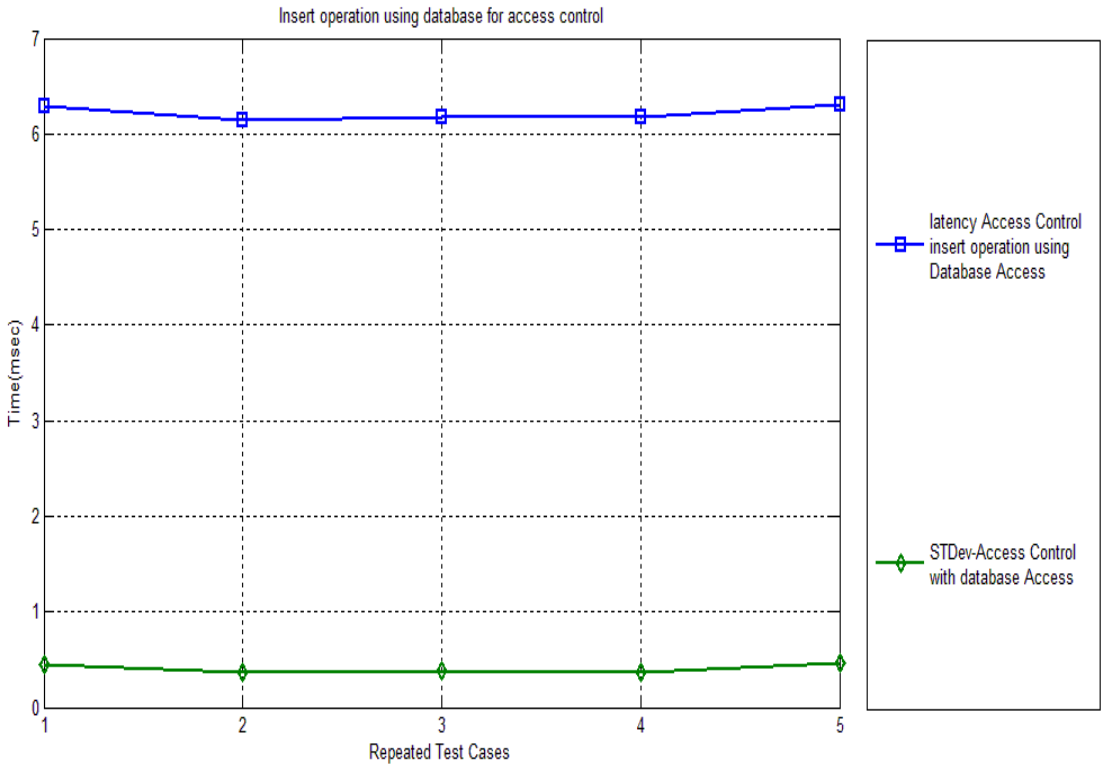


Figure 6-2 : Test results for inserting access control privileges using database

Table 6-3 : Statistics for the Figure 6-2

Repeated Test Case	1	2	3	4	5
Average Timings (msec)	6.29	6.15	6.17	6.17	6.31
STDev (msec)	0.45	0.36	0.38	0.37	0.47

Insertion operation average time for access control privileges for user into database can result in around 6.15 msecs. This result shows that processing of the client insert request does not require much time to process in the server side of the system.

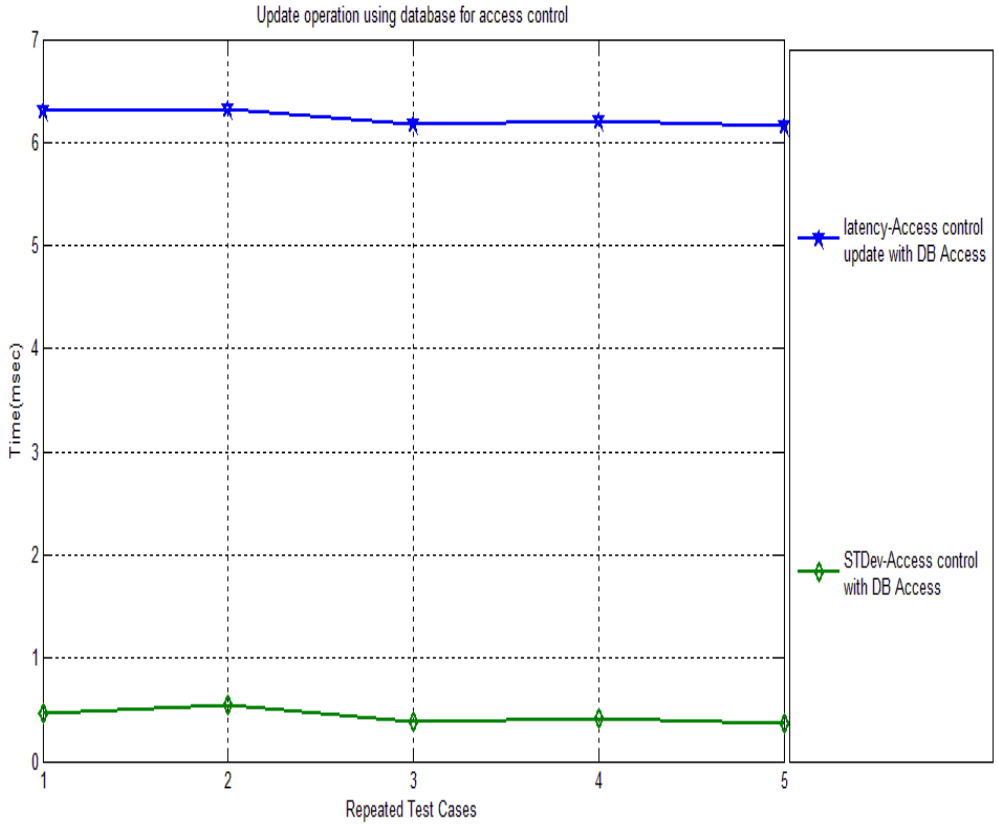


Figure 6-3 : Test results for updating access control privileges using database

Table 6-4 : Statistics for the Figure 6-3

Repeated Test Case	1	2	3	4	5
Average Timings (msec)	6.31	6.32	6.18	6.21	6.16
STDev (msec)	0.47	0.55	0.39	0.41	0.37

Update operation average time for access control privileges for user in the database can result in around 6.20 msecs. This results show that processing of the client update request does not require much time to process in the server side of the system. Moreover, this results show that update and insert operation of the access control

privileges average time is very close to each other. Hence, these update and insert operations are almost executed in the same time.

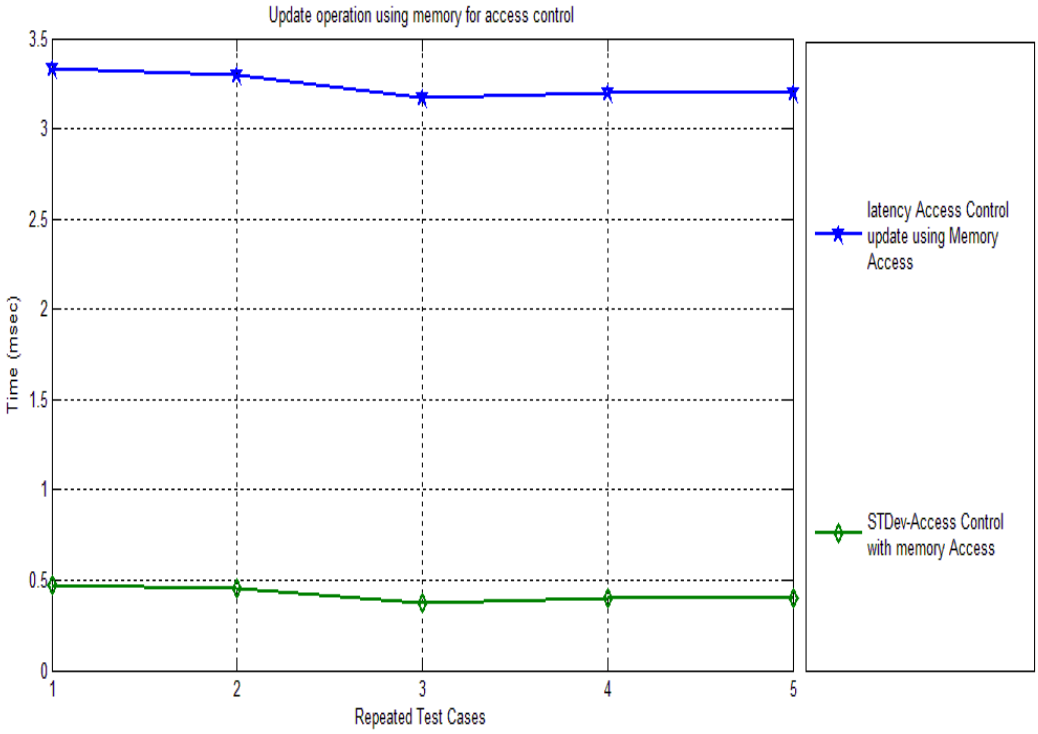


Figure 6-4 : Test results for updating access control privileges using memory

Table 6-5 : Statistics for the Figure 6-4

Repeated Test Case	1	2	3	4	5
Average Timings (msec)	3.33	3.30	3.17	3.2	3.2
STDev (msec)	0.47	0.45	0.37	0.40	0.40

Update operation average time for access control privileges for user using memory can result in around 3.20 msecs. This result shows that processing of the client

update request does not require much time to process in the server side of the system. This time is almost half of the average time for updating of the access control privileges. This results shows that update operation utilizing memory has advantage over database access in respect to the average execution time.

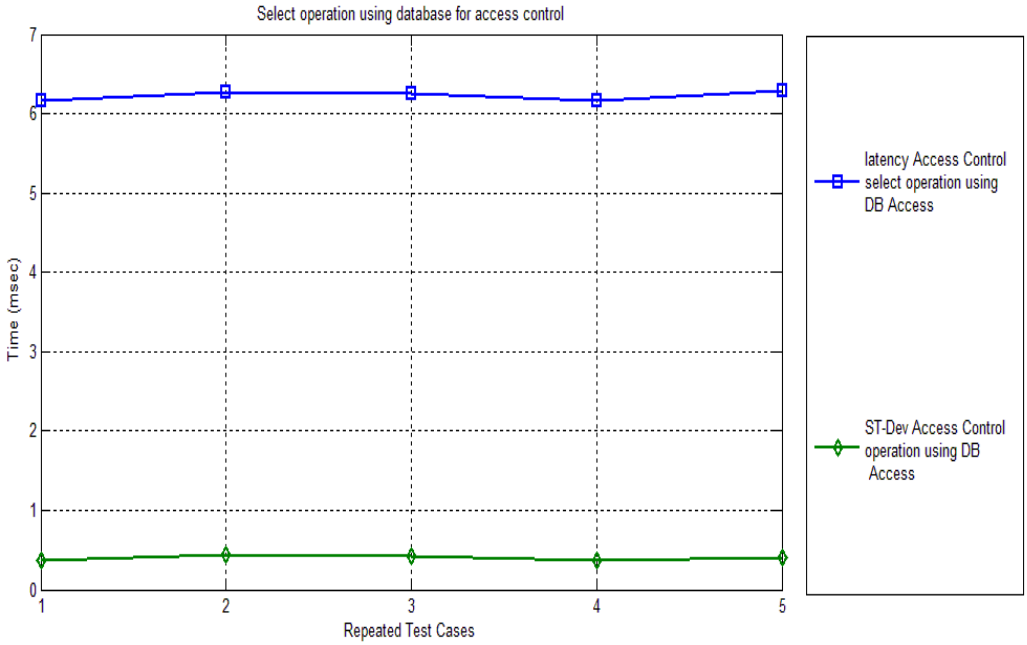


Figure 6-5 : Test results for selecting access control privileges from database

Table 6-6 : Statistics for the Figure 6-5

Repeated Test Case	1	2	3	4	5
Average Timings (msec)	6.17	6.27	6.26	6.16	6.19
STDev (msec)	0.37	0.44	0.43	0.36	0.40

Select operation average time for access control privileges for user from database can result in around 6.20 msecs. This result shows that processing of the client insert request does not require much time to process in the server side of the system.

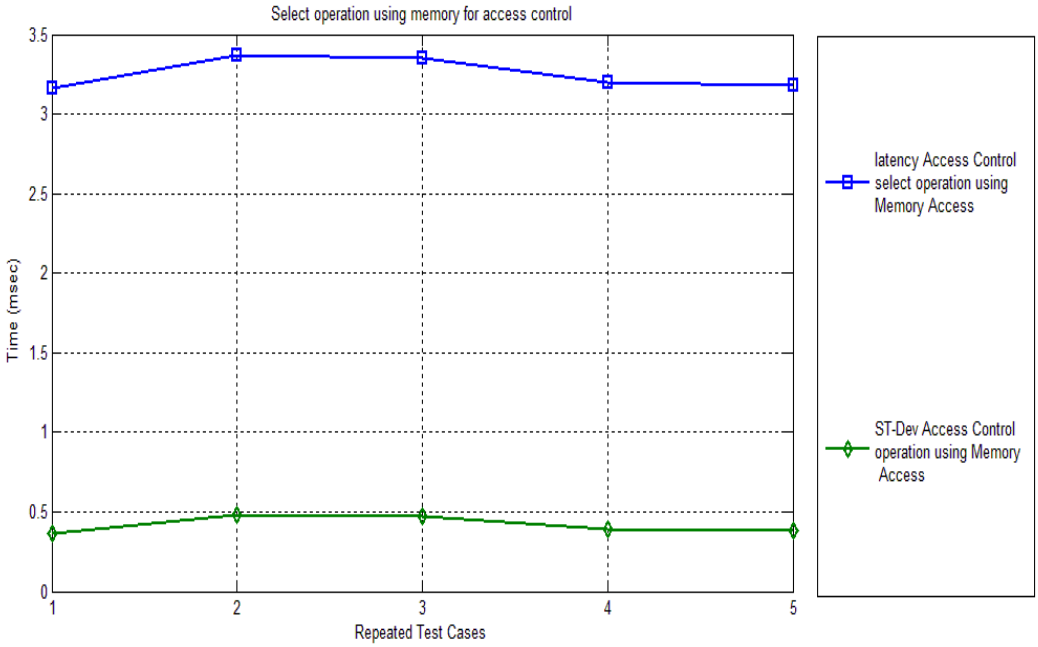


Figure 6-6 : Test results for selecting access control privileges from memory

Table 6-7 : Statistics for the Figure 6-6

Repeated Test Case	1	2	3	4	5
Average Timings (msec)	3.16	3.37	3.35	3.20	3.18
STDev (msec)	0.36	0.48	0.47	0.39	0.38

Select operation average time for access control privileges for user using memory can result in around 3.23 msecs. This result shows that processing of the client select

request does not require much time to process in the server side of the system. This time is almost half of the average time for selecting the access control privileges from database. This results shows that select operation utilizing memory has advantage over database access in respect to the average execution time.

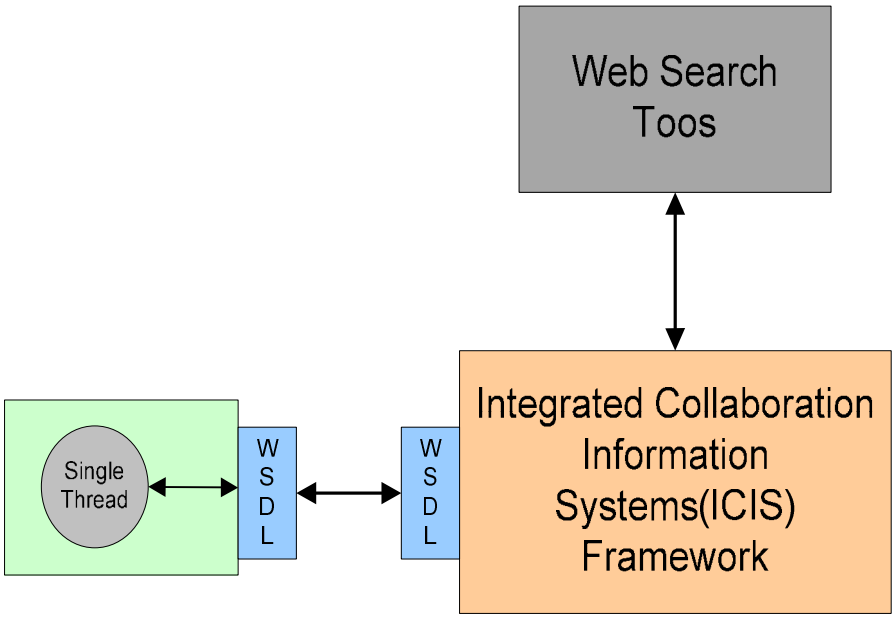


Figure 6-7 : System Responsiveness Experiment Testing Cases using Web Search Tools

The next experiment is to evaluate the baseline performance of the Integrated Collaboration Information System Framework using Web Search Tools. This experiment is depicted in Figure 6-7, and we have used Google Scholar Web Search Tool to evaluate performance of the test case.

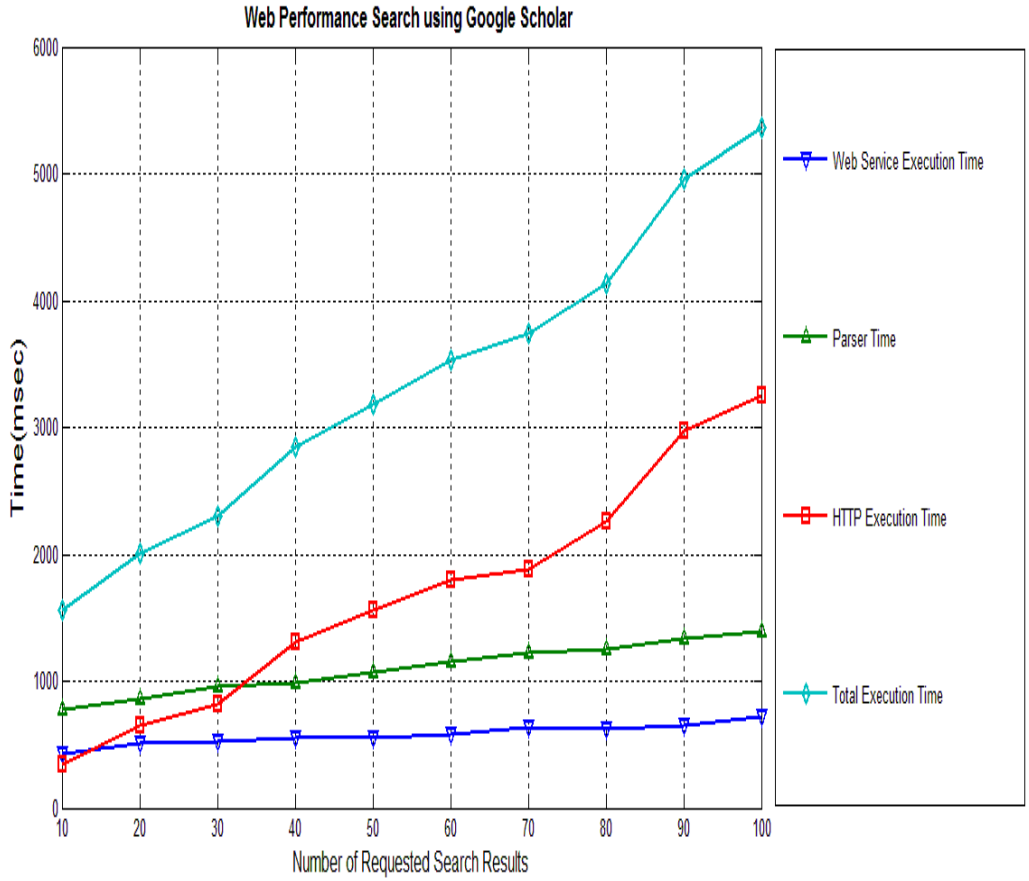


Figure 6-8 : Web Search Tool Performance using Google Scholar

Table 6-8: Statistics for the Figure 6-8

Number of Results	Web Service Time(msec)	Data Parser Time(msec)	HTTP Process Time(msec)	Total Process Time(msec)
10	430	779	346	1555
20	504	864	644	2012
30	521	961	816	2298
40	555	988	1308	2851
50	558	1068	1553	3179
60	586	1148	1795	3529
70	636	1224	1876	3736
80	623	1252	2254	4129
90	649	1330	2973	4952
100	716	1393	3257	5366

In this experiment we have explored the performance of the Google Scholar Search performance using our Integrated Collaboration Information Systems Framework. Test results are depicted in

Figure 6-8, and the detailed statistics are depicted in Table 6-8.In this experiment, number of requested results is increased by 10 up to the 100 results. The formula is for calculating the Total Time;

Total Time (*tt)* = Web Service Time (*wst*) + Data Parser Time (*dpt*) + HTTP Process Time (*hpt*);

In these results, when number of results are increased *wst* and *dpt* are linearly increased. Most of the operation time is spent during HTTP operations. Google Scholar does not provide an API or other web services. We need to use HTTP method to extract the information. Therefore, we have used heuristic approach to parse the HTTP response contents to extract Digital Entities from web page content. This operation should be done for each resultant pages coming from Google Scholar. The total execution time is depicted in Figure 3-2 and we observed that total response time is linearly increased while increasing the number of search requests.

6.3 Scalability Experiment

The scalability tests are designed to investigate the scalability of the Integrated Collaboration Information Systems Framework. We have performed a series of analysis to do these tests for this framework implementation. We have performed following test cases in order to analyze the scalability of the system: a) Increase the message rate per second for search operation request for any keywords from stored Digital Entities (DEs)

with Database Access; b) Increase the message rate per second for search operation request for any keywords from stored DEs with memory utilization; c) Increase the size of the stored data for search operation. By doing these test cases we tried to answer following questions: a) How well does the system performs for search operation by increasing message rate per second for both Database access and memory utilization?; b) How well does the system performs for search operation by increasing data size? The

Figure 6-9, Figure 6-10, and Figure 6-11 below shows the design of the testing configuration for the Integrated Collaboration Information Systems Framework.

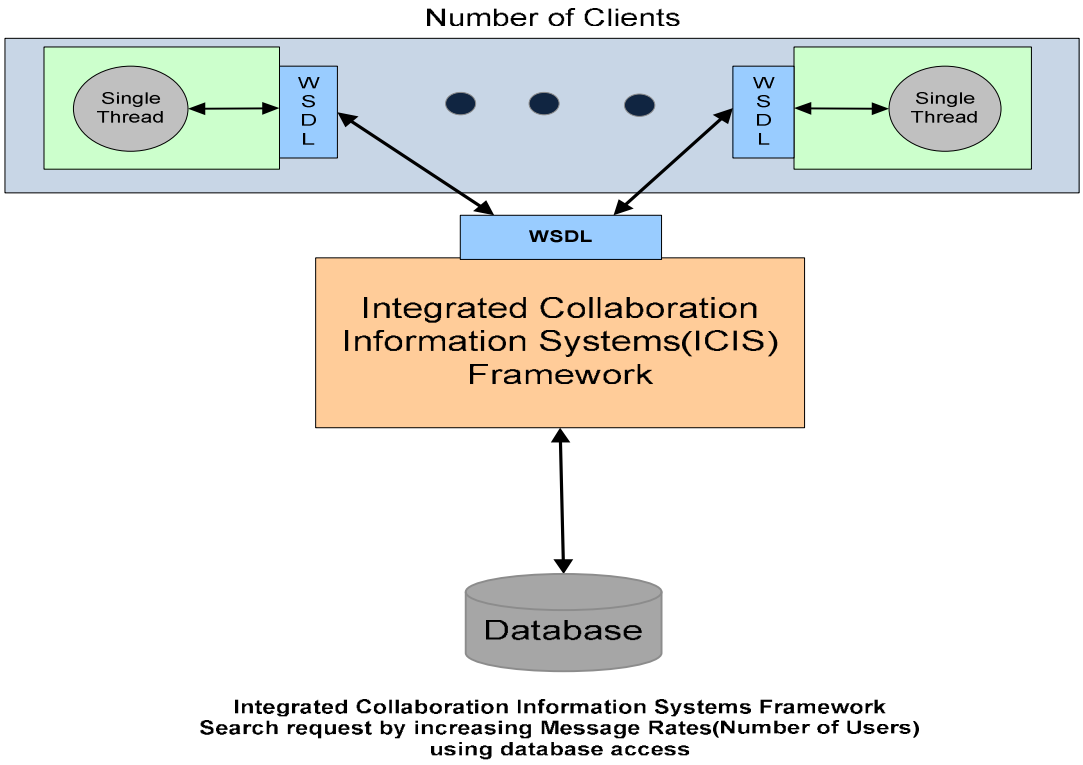


Figure 6-9 : Testing cases for scalability experiment for search request using DB

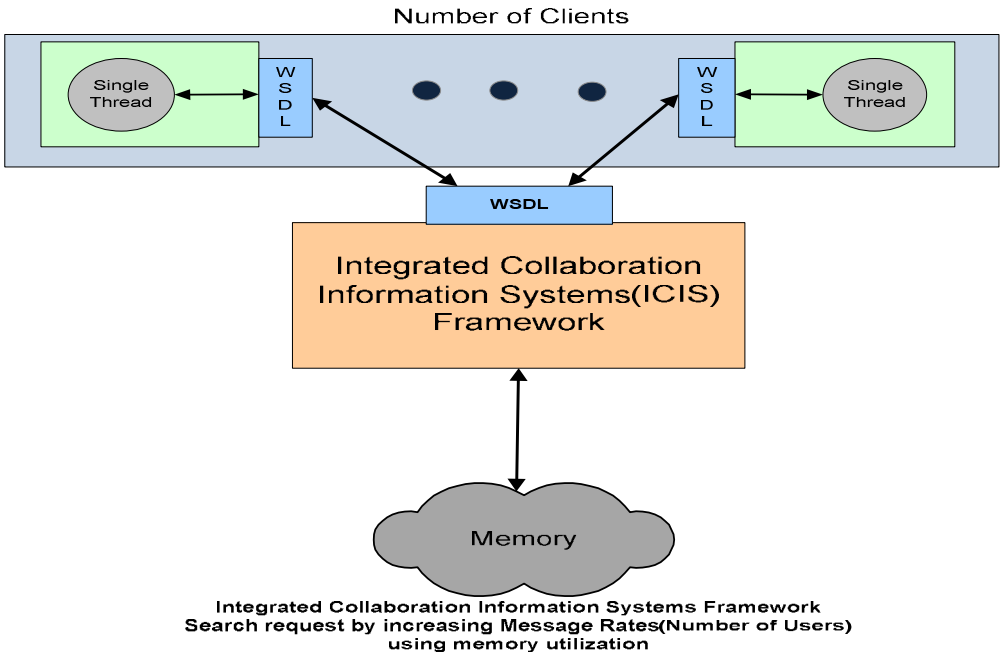


Figure 6-10 : Testing case for scalability experiment for search request with memory utilization

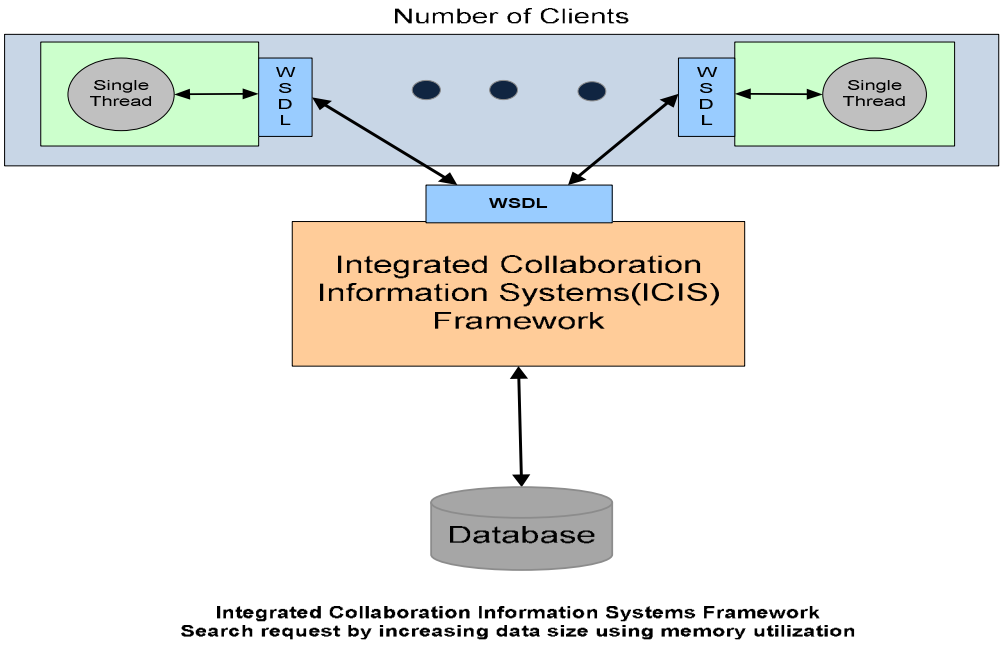


Figure 6-11 : Testing case for scalability experiment for search request by increasing data size with memory utilization

In the first experiment shown in Figure 6-10 above, our goal is to quantify the performance degradation by increasing number of the search requests requiring Database access in the Integrated Collaboration Information Systems Framework. In this test case, we have increased the message rate /number of users per second to observe the results until the response time degrades. In these steps, we have also measured the round trip time for each search request by specifying the keywords to populate successful search results among Digital Entities stored in the database.

In the second experiment shown in Figure 6-10 above, we have used the same test strategy as defined in the previous experiment. Only difference in this test case is using memory utilization instead of database for search request.

In the third experiment shown in Figure 6-11 above, we performed test case to analyze the performance degradation by increasing data sizes in the Integrated Collaboration Information Systems Framework. In this test case, 600 users performed a search request per second. So, average round trip time is measured for different number of Digital Entities stored in the database.

6.3.1 Results of the Scalability Experiment

We have performed the test cases are depicted in Figure 6-12 and Figure 6-13, and results listed in Table 6-9and Table 6-10. Based on these results, concurrent requests may well be responded without any error by Integrated Collaboration Information Systems Framework. However, we have observed that performance starts to degrade after a certain number of messages per second using both database access and memory utilization. The performance starts to drop down after around 1790 requests per second

for search operation with Database access, after around 2600 requests per second for search operation with memory utilization. These results shows that performance start to degrade because of the high message rate. The main reason for these thresholds is Apache Tomcat thread scheduling and context switches. Because, Apache Tomcat thread scheduling and context switches try to handle incoming request at high message rate. In order to satisfy this operation, performance starts to degrade at some points shown in Figure 6-12 and Figure 6-13.

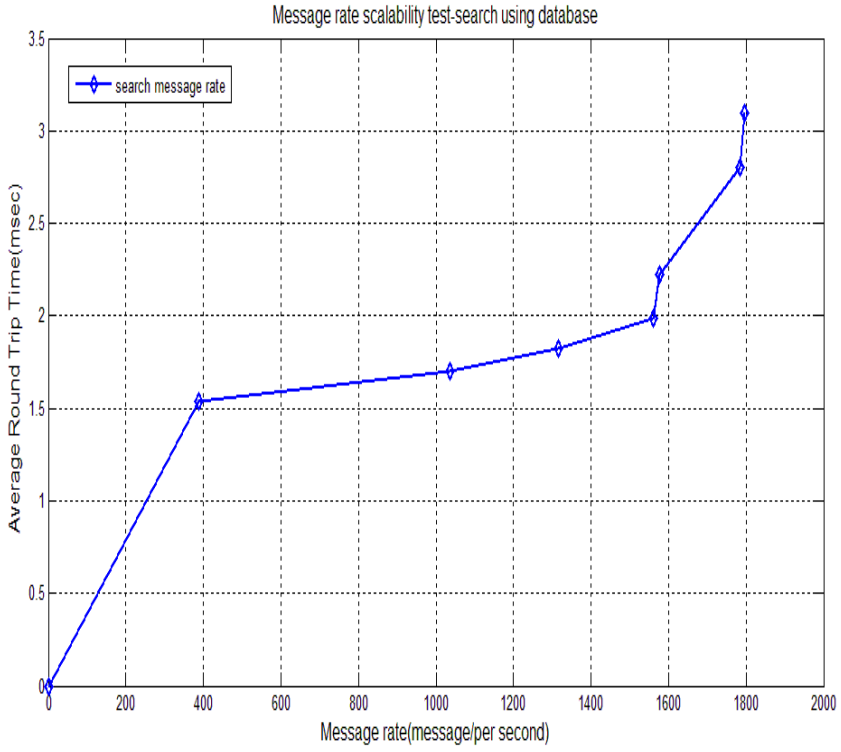


Figure 6-12 : Search message rate with Database access

Table 6-9: Statistics for the Figure 6-12

Message per second	Average response time(msec)
0	0
388	1.54
1036	1.70
1315	1.82
1561	1.99
1578	2.22
1784	2.80
1796	3.10

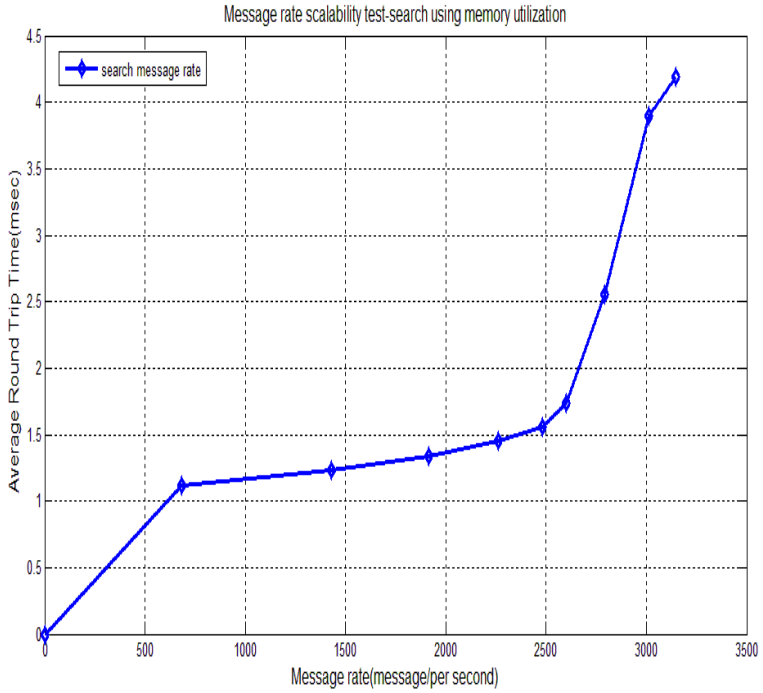


Figure 6-13 : Search message rate with memory utilization

Table 6-10 : Statistics for the Figure 6-13

Message per second	Average response time(msec)
0	0
685	1.12
1430	1.23
1915	1.34
2261	1.45
2481	1.56

2599	**1.74**
2792	**2.55**
3013	**3.90**
3147	**4.19**

We have performed the test case are depicted in Figure 6-14 and results listed in Table 6-11. In this test case, 600 users performed search request per second. Based on these results, average response time is increased while increasing data size. These requests are responded without any error by Integrated Collaboration Information Systems Framework.

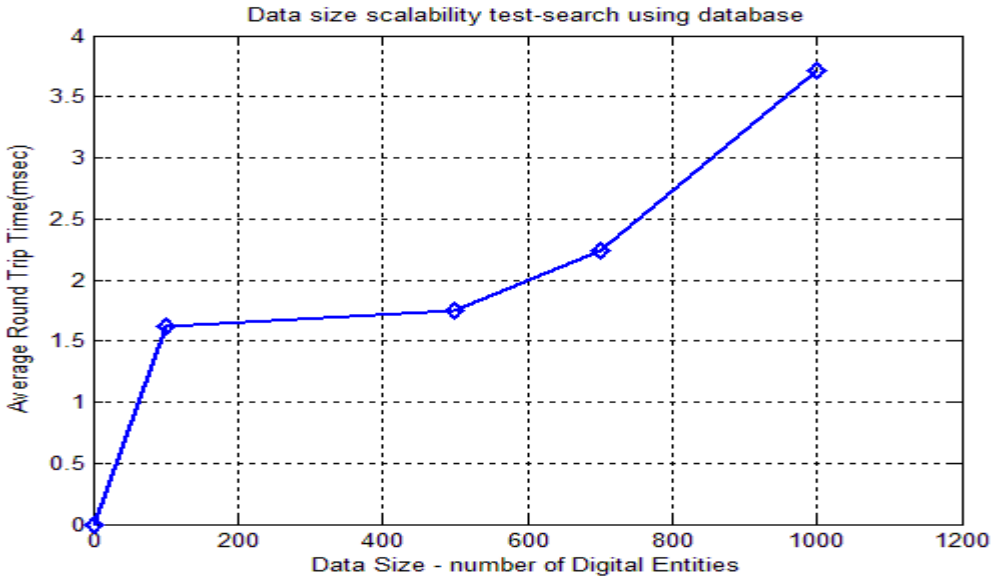


Figure 6-14 : Data size scalability test with Database access

Table 6-11 : Statistics for the Figure 6-14

Data size(number of Digital Entities)	Average response time(msec)
0	0
100	1.62
500	1.75
700	2.24
1000	3.71

The possible causes for threshold values which shown in previous figures: a) Apache Tomcat thread scheduling and context switches; b) Network Bandwidth; c) Limitation of the open sockets in Linux operation systems.

For the possible cause of the Apache Tomcat thread scheduling and context switches, we investigate the possible degradation by sending echo request message per second. So we have observed same graphical pattern. We concluded that the main reason for these thresholds is Apache Tomcat thread scheduling and context switches at higher message rate.

The network capacity in Community Grids Lab (CGL) is 1GBits/sec. Average message size for search request is ~1KB. Message size is 8192 bits.

A total network is required at threshold value for using memory utilization is 2600 message/sec * 8192 bit/message =25.3MBits/sec.

The network capacity of the message request is almost % 2.5 of the CGL network. Hence, the network is not the reason for threshold in our test results.

The default number of the open socket connection for Linux user is 1024. We have increased this default number to 2048. We have conducted same results as we retrieved using default socket connection. Hence, the threshold value, which is obtained during test cases, is not cause of the default number of open socket connection.

6.4 Summary

This chapter presented and evaluated the performance of the Integrated Collaborative Information Systems Framework. In the experiments explained in the previous sections, we have investigated the possible threshold values for the search operation using database access and memory utilization. The performance starts to drop down for search operation using database access after 1790 simultaneous messages per second. The same experiment using memory utilization is around 2600 simultaneous messages per second. These results indicate that higher scalability lowers the performance when the system exceeds certain threshold values. Hence, system is able to scale while increasing message sizes and number of users and performs well.

CHAPTER 7

CONCLUSIONS AND FUTURE RESEARCH ISSUES

7.1 Thesis summary

This thesis studied Integrated Collaborative Information Systems Framework to develop a community building system consist of mechanism to collect information stored in the "central" location that offers input/output services. These services are completed with WSDL (Web Service Definition Language) interfaces to provide wrapper services. The Integrated Collaborative Information Systems provides architecture to integrate various tools using benefits of the tools. In this architecture one can assemble information coming from various resources.

First, we have determined the scope of research by describing motivations in 1.1 and identifying the problems described in 1.2. Next, we reviewed the background

information and the core technologies in CHAPTER 2. Next, we proposed our architecture and design approach of the Integrated Collaboration Information Systems Framework by identifying our motivations and reviewing the related research works and core technologies in CHAPTER 3. Then, we described access control model used in our Integrated Collaboration Information Systems Architecture in CHAPTER 4. Afterward, we defined and implemented our research implementation in CHAPTER 5 in order to demonstrate the applicability of the Integrated Collaboration Information Systems Framework. Finally, we presented the prototype evaluation and analysis of the test results in CHAPTER 6.

The main purpose of the collaboration provides sharing information easily between user communities. People want to share more information easily and they want to tag, comment, and rate the information by using the community tools. Web Search Tools such as CiteSeer and Google Scholar have enabled open, fast and easy access to vast online repositories of linked scientific documents. People can access digital documents by using these tools besides using digital libraries directly. Social bookmarking tools such as del.icio.us, CiteULike, and Connotea provide tagging and sharing of scholarly publications. However, search and bookmarking tools have limitations such as they do not have common metadata definition, and they have limited metadata support. One tool also provides service for specific type of field of study and the information in other field of study remains largely outside the scope of the system. For example a search in Google Scholar does not return all publications when they are in different category.

The Integrated Collaborative Information Systems (ICIS) Framework provides integration of the tools which have common metadata to support complete metadata defined in a scientific area. The framework provides a mechanism that ICIS uses the benefits of the integrated tools. There are many efforts for collaboration and sharing between users and communities. Web 2.0 also represents new-based services that is explained in section 2.3 and provides reusable services and data. However, tools in this web domain are not complete and they are separated from each other. They do not have direct interaction between them. ICIS architecture provides community-centric platform of tools and services that integrate the tools to Cyberinfrastructure based scholarly research. Thus, we have maximized re-use of the world's tools and ease of customizing the tool for different applications.

The Integrated Collaborative Information Systems (ICIS) Framework does not depend on one tool or platform or services. It provides flexible architecture to allow integration of new tools. This provides extendibility and integration easily. Therefore, we use the advantage and capabilities of the best tools in their area for scientific community.

Proposed ICIS Architecture provides combining data from variety of sources. It creates add-value data and metadata generated within specific communities. ICIS Architecture also supports level of authorization and access control to provide a protection of the users and groups from other users. Users can be in any group to share metadata with other members and modify the metadata if users have required permissions as explained in Section 4.2 and 4.3.

ICIS Architecture supports the Event-based Infrastructure and Consistency Framework explained in [87]. This service provides flexibility to have versions of the

documents and one can navigate among the versions of documents. It also provides management of set of documents in the system. Therefore, ICIS Architecture allows one to have consistent data and use them in the system to protect the both original and updated information.

7.2 Answering the research questions

Here, we answer our research questions mentioned in Section 1.2.

Can we implement a framework by integrating tools that handle data and metadata from various tools and resources in Service Oriented Architecture?

The answer to this question is “yes”. We introduced the Integrated Collaborative Information Systems (ICIS) Architecture that integrates a number of tools supporting various data and metadata from these to develop add-value community building scientific community to use benefits of the web resources. The architecture defines a mechanism to collect information stored in "central" location that offers input/output services. These services are completed with Web service to provide wrapper services. In this architecture one can assemble information coming from various tools and resources.

What are the architectural and implementation principles to use the Integrated Collaborative Information Systems (ICIS) Architecture? How easy is it to integrate new tool to the ICIS?

We have defined the gateways explained in Section 3.1 to build a communication between integrated tools and ICIS Architecture. The key feature is to reuse the tools by

building wrappers around these resources, and use the benefits of them by integration. The model is easier to link together all relating information coming from various integrated tools. In this architecture we have also defined the Integration Manager, which is explained in Section 3.1.1, have information service and provide a communication between tools, client, and responsible for integration operation in the system. Two major operations, which are pull and push services, are defined in this systems. Pull Service interacts with the tools using the HTTP method or the Web Services through the Tool Gateway to handle the client requests coming from the Client Gateways defined in Section 3.1.1.

How does Integrated Collaborative Information Systems (ICIS) use the tools? How does ICIS architecture provide a mechanism to link data between the tools?

Integrated Collaborative Information Systems (ICIS) pull and push services provides communication through the tools using the gateways defined for each tool. Some of the tools support both uploading and downloading information. For example, del.icio.us, Connotea supports these operations and ICIS defined both pull and push operations using Web service for these tools. Having these services, information coming from one tool can be stored in ICIS repository. ICIS can send all or part of the information to the other tools using push operations built as Web services as defined in Section 3.1.1. In summary, interoperability can be established through the integrated tools.

How does ICIS architecture provide data sharing between users and protect data from other user?

ICIS handles both individual users and groups of users, and supports sharing and collaboration between users and group members. We have defined three permissions for each Digital Entity: Read, Write, and Delete. Owner of the DEs can give permissions to the other users for each DE and also folders which consist of many DEs. Users in a specific group can access DEs associated with the group. User also can give permissions to any user(s) or group(s) both for specific DEs or all DEs associated with the folder. The concept is defined and explained in CHAPTER 4.

Does system architecture provide to use the benefit of the integrated tools without developing new tool?

Integrated Collaborative Information Systems (ICIS) Architecture provides flexibility to integrate the tools into the system. Having that we can easily integrate new web resources developed in rapid environment using benefits of the resources. We do not need to implement for example a search tool similar to Google Scholar. It is covering many organization's publications and it has powerful search features. Instead of developing new web resources, we extract the information, which Google Scholar provides to the users, by building wrappers around the tool.

Does the integration architecture approach scales well?

We have prototype implementation using series of measurements and its practical usefulness in the applications and results are discussed in *6.3.*We have concluded that our Integrated Collaboration Information Systems scales very well.

What is the performance of the Integrated Collaborative Information Systems (ICIS) and what factors influence the performance?

We have investigated the performance of our prototype system and results are given in Section 6.2. We have explained the reasons for performance degradation in Section 6.3.1. Apache Tomcat thread scheduling and context switches cause the possible degradation. Increase in the data size is also effects the performance.

Can we support data sharing and uploading and downloading for scholarly publications using our integrated architecture?

The answer to this question is "yes". Integrated Collaborative Information Systems Framework provides services that user can easily both upload information to the tools and push data to other communities using Web 2.0 features such as RSS or Atom feeds. Moreover, information can be extracted from the web resources that allow user to download data from their systems. The detailed concept and implementation explained in Section 3.1.1 and CHAPTER 5 respectively.

How does our integration model support the consistent data for scholarly publications?

Integrated Collaborative Information Systems Framework supports and use the event-based mechanism to provide consistency of the Digital Entities. Because in our collaborative environment users work together and may share the same resources with each other. To avoid undesired changes in the system and restore the previous versions of the documents we adapted and used Event-based Infrastructure and Consistency model.

7.3 Future research directions

This thesis represents the Integrated Collaborative Information Systems Framework which provides integration of the tools which have common metadata to support complete metadata defined in a scientific area. The framework provides a mechanism that ICIS uses the benefits of the integrated tools. It also allows one to exchange information easily between tools and use one's favorite interface for a particular task by aggregating information from a variety of sources (i.e., "mash-up" tools) and provide added value to communities of researchers. We plan to expand this approach to open-access scientific databases, such as PubMed, PubChem, and Science.gov, have been created over the years. These databases constitute the "deep Web" and have been estimated to contain 400-500 times more public content than the "surface Web" [75]. Since the deep Web is largely invisible to current search engines (including academic ones), this wealth of information has not been integrated with the online research tools. Also, we plan to apply this integration approach to other application domains such as streaming collaboration systems.

Appendix A

REFERENCES

1. O'Reilly, T. (2005) *What is Web 2.0: Design patterns and business models for the next generation of software*.
2. *CiteULike web site*. [cited; Available from: http://www.citeulike.org.
3. *Connotea web site*. [cited; Available from: http://www.connotea.org.
4. *Bibsonomy web site*. [cited; Available from: http://www.bibsonomy.org.
5. Lee Giles, C., K. Bollacker, and S. Lawrence. *CiteSeer: An automatic citation indexing system*. in *Proceedings 3rd ACM Conference on Digital Libraries (DL'98)*. 1998. Pittsburgh, PA.
6. Giles, J., *Science in the Web age: start your engines.* Nature, 2005. **438**: p. 554-555.
7. Weerawarana, S., et al., *Web services platform architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and more* 2005, Upper Saddle River, NJ Prentice Hall.
8. Hey, T. and A.E. Trefethen, *Cyberinfrastructure for e-Science.* Science, 2005. **308**(5723): p. 817-821.
9. Fox, G. *Collaboration and Community Grids*. in *Proceedings International Symposium on Collaborative Technologies and Systems (CTS 2006)* 2006.
10. Fox, G., *Some comments on CiteULike, Connotea and related tools*. 2006, Community Grids Lab, Indiana University.
11. John Musser, T.O.R., and O'Reilly Radar Team *Web 2.0 Principles and Best Practices*.
12. *RSS 2.0 Specification*. David Winer.

13. Sayre(Eds), M.N.a.R., *The Atom Syndication Format*.

14. Garrett, J.J., *Ajax: A New Approach to Web Applications*.

15. *Microformats*. [cited; Available from: http://microformats.org/.

16. Fielding, R.T., *Architectural Styles and the Design of Network-based Software Architectures*, University of California, Irvine USA.

17. *Google Scholar help web site*. [cited; Available from: http://scholar.google.com/intl/en/scholar/about.html.

18. Sadeh, T. (2006) *Google Scholar versus Metasearch Systems*. High Energy Physics Libraries Webzine,

19. Jasco, P., *Google Scholar: the pros and the cons*. Online Information Review, 2005. **29**(2): p. 208-215.

20. Jasco, P. *Google scholar beta.* Péter's Digital Reference Shelf 2004 [cited; Available from: http://www.gale.cengage.com/servlet/HTMLFileServlet?imprint=9999&region=7&fileName=/reference/archive/200412/googlescholar.html.

21. *The New Zealand Digital Library Project*. [cited; Available from: http://www.dlib.org/dlib/november96/newzealand/11witten.html.

22. Petricek, V., et al. *A comparison of online computer science citation databases*. in *Proceedings 9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL05)*. 2005.

23. Zhuang, Z., R. Wagle, and C.L. Giles. *What's there and what's not? Focused crawling for missing documents in digital libraries*. in *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*. 2005. Denver, CO, USA: ACM Press.

24. Chakrabarti, S., M. van den Berg, and B. Dom, *Focused crawling: a new approach to topic-specific Web resource discovery*. Computer Networks, 1999. **31**(11-16): p. 1623-1640.

25. Li, H., et al. *CiteSeer^X: an architecture and web service design for an academic document search engine*. in *Proceedings 15 International Conference on the World Wide Web (WWW06)*. 2006. Edinburgh, Scotland.

26. *Live Search Academic*. [cited; Available from: http://en.wikipedia.org/wiki/Live_Search_Academic.

27. Jasco, P. (2006) *Windows Live Academic*

28. O'Leary, M., *Windows live academic search...why?*, in *Information Today*. 2007. p. 45-50.

29. Quint, B. (2006) *Windows Live Academic Search: The Details*.

30. *Natura Publishing Group*. [cited; Available from: http://npg.nature.com.

31. Lund, B., et al., *Social Bookmarking tools (II): A case study - Connotea.* D-Lib Magazine, 2005. **11**(4).

32. Winer, D. *RSS 2.0 Specification*. [cited; Available from: http://cyber.law.harvard.edu/rss/rss.html.

33. *Delicious web site*. [cited; Available from: http://del.icio.us.

34. Biddulph, M. *Introducing del.icio.us*. XML.com

35. Ian, F., K. Carl, and T. Steven, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*. 2001, Sage Publications, Inc. p. 200-222.

36. *TeraGrid web site*. [cited; Available from: http://www.teragrid.org.

37. *The Open Science Grid Web Site*. [cited; Available from: http://www.opensciencegrid.org.

38. Atkinson, M., et al., *Web Service Grids: an evolutionary approach.* CONCURRENCY AND COMPUTATION, 2005. **17**(2/4): p. 377-390.

39. David Booth, W.C.F.H.-P., et al. *Web Services Architecture*.

40. *Towards Open Grid Services Architecture*. [cited; Available from: http://www.globus.org/ogsa/.

41. *LBase: Semantics for Languages of the Semantic Web*. W3C Working Group Note 10 October 2003

42. Martin Gudgin, M., et al. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*.

43. Christensen, E., et al. *Web Services Description Language (WSDL) Specification*.

44. Ian Foster, C.K.J.M.N.S.T., *The Physiology of the Grid*, in *Grid Computing*, G.F.T.H. Fran Berman, Editor. 2003. p. 217-249.

45. *YouTube web site*. [cited; Available from: http://www.youtube.com/.

46. *Flickr web site*. [cited; Available from: www.flickr.com.

47. Geoffrey Fox, M.P., *Grids challenged by a Web 2.0 and multicore sandwich.* Concurrency and Computation: Practice and Experience, 2009. **21**(3): p. 265-280.

48. *Sakai project web site*. [cited; Available from: http://sakaiproject.org/portal.

49. *Drupal project we site*. [cited; Available from: http://drupal.org.

50. *Drupal handbook*. [cited; Available from: http://web.archive.org/web/20041015205922/drupal.org/node/10250.

51. *Microsoft Office SharePoint Server*. [cited; Available from: http://www.microsoft.com/Sharepoint/default.mspx.

52. *Microsoft SharePoint*. [cited; Available from: http://en.wikipedia.org/wiki/Microsoft_SharePoint.

53. Hinchcliffe, D. (2008) *Ten leading platforms for creating online communities*.

54. *Joomla web site*. [cited; Available from: http://www.joomla.org/.

55. *PHP-NUKE web site*. [cited; Available from: http://phpnuke.org/.

56. *Community Server web site*. [cited; Available from: http://communityserver.com/.

57. *jive web site*. [cited; Available from: http://www.jivesoftware.com/.

58. Mahmoud, Q.H. (April 2005) *Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI)*.

59. Erik Christensen, F.C., IBM Research, M. Greg Meredith, and I.R. Sanjiva Weerawarana *Web Services Description Language (WSDL) 1.1*.

60. *OASIS UDDI Specification*. [cited; Available from: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec.

61. *Axis web page*. [cited; Available from: http://ws.apache.org/axis.

62. *SOAP Specifications*. [cited; Available from: http://www.w3.org/TR/soap.

63. *Document Object Model(DOM)*. [cited; Available from: http://www.w3.org/DOM/.

64. Mike Champion, et al. *Document Object Model (Core) Level 1*.

65. Scott Means, W., and Michael A. Bodie., *The Book of SAX: The Simple API for XML*. 2002.

66. *Simple API for XML(SAX)*. [cited; Available from: http://www.saxproject.org/.

67. *Hypertext Transfer Protocol -- HTTP/1.1*.

68. *Jakarta Commons HTTP Client*. [cited; Available from: http://hc.apache.org/httpclient-3.x/features.html.

69. Garrett, J.J. (2005) *Ajax: A New Approach to Web Applications*.

70. Ullman, C. and L. Dykes, *Beginning Ajax*. 2007.

71. Clark, J. and S. DeRose. *XML Path Language (XPath) Version 1.0*. [cited; Available from: http://www.w3.org/TR/xpath.

72. *XSL Transformations (XSLT)*. [cited; Available from: http://www.w3.org/TR/xslt.

73. *XML Pointer Language (XPointer). W3C Working Draft*. [cited; Available from: http://www.w3.org/TR/WD-xptr.

74. *Atom as the New XML-Based Web Publishing and Syndication Format*. [cited; Available from: http://xml.coverpages.org/ni2003-10-22-a.html.

75. Bergman, M.K. (2001) *The Deep Web: Surfacing Hidden Value* Journal of Electronic Publishing

76. Anthony, I.W., *Tool integration in software engineering environments*, in *Proceedings of the international workshop on environments on Software*

engineering environments. 1990, Springer-Verlag New York, Inc.: Chinon, France.

77. Aleman-Meza, B., et al. *Semantic analytics on social networks: experiences in addressing the problem of conflict of interest detection*. in *Proc. 15th International Conference on World Wide Web (WWW'06)*. 2006. Edinburgh, Scotland: ACM Press.

78. Ahmet Fatih, M., et al., *A Novel Event-Based Consistency Model for Supporting Collaborative Cyberinfrastructure Based Scientific Research*, in *Collaborative Technologies and Systems CTS 2007*. 2007: Orlando.

79. Geoffrey Fox, A.F.M., Ahmet E. Topcu, Aurel Cami. *SRG: A Digital Document-Enhanced Service Oriented Research Grid*. in *Information Reuse and Integration (IEEE IRI-2007)*. 2007. Las Vegas, USA.

80. William, T., et al., *Access control in collaborative systems*. 2005, ACM Press. p. 29-41.

81. Elisa, B., J. Sushil, and S. Pierangela, *A flexible authorization mechanism for relational data management systems*. 1999, ACM Press. p. 101-140.

82. Andrew, S.T. and R. Robbert Van, *Distributed operating systems*. 1985, ACM Press. p. 419-470.

83. Butler, W.L., *Protection*. 1974, ACM Press. p. 18-24.

84. Thomas, K., *Review: Microsoft academic search with a very personal touch*. 2006.

85. Anderson, C., *The long tail: why the future of business is selling less of more*. 2006: Hyperion.

86. Fox, G.C., et al., *SRG: A Grid of Services for Supporting Cyberinfrastructure Based Scientific Research*. 2006, Community Grids Lab, Indiana University.

87. Mustacoglu, A.F., et al., *A Novel Event-Based Consistency Model for Supporting Collaborative Cyberinfrastructure Based Scientific Research*, in *The 2007 International Symposium on Collaborative Technologies and Systems*. 2007.

Bibliography

Vitae

Name of Author: Ahmet E. Topcu

Date of Birth: June 24, 1973

Place of Birth: Uskudar, Turkey

Degrees Awarded:

M.S. in Computer Engineering, August 2001, Syracuse University

B.S. in Electrical and Electronics Engineering, August 1997, Middle East Technical University