LAKEHEAD UNIVERSITY

Building a Semantic Blog Support System for Gene Learning Objects on Web 2.0 Environment

by

Wei Yuan

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

THUNDER BAY, ONTARIO, CANADA

SEPTEMBER 2008

Your file Votre référence
ISBN: 978-0-494-43440-6
Our file Notre référence
ISBN: 978-0-494-43440-6

Abstract

Blogging has become a popular practice on Internet in recent years, and it has been used as information publishing and participate platforms. In recent years, a style of blog called 'semantic blogs' have been introduced into the field. Semantic blogs are blogs enriched with machine-understandable metadata (Möller, Breslin, & Decker, 2005). They are an extension of regular blogs.

Recently, a new web technology theory was proposed called Web 2.0. Unlike traditional web technology which only allows web users to accept information passively, Web 2.0 provides web users the option to actively modify web information.

Learning Objects are digital entities deliverable over the Internet. Any number of people can access and use them simultaneously. Moreover, users can collaborate on learning objects and benefit immediately from adding their information or appending others' work to Learning Objects and share with other users over the Internet.

This thesis is dedicated to the development of a semantic blog prototype for Gene Ontology annotation and navigation as a Web 2.0 support system. We are developing this semantic blog specifically because we did not find an effective system already in place that can provide support for biomedical researchers.

The existing Gene Ontology systems can be classified into various categories: offline applications, client-server applications, web search engines, portals, and FTP servers. Researchers face a number of bottlenecks within the current system; all of them are based on traditional web technology with no collaboration among individual gene ontology researchers, and annotation can only be published by certain organizations.

This thesis seeks the possibility to use Learning Object with Gene Ontology along with the semantic of how researchers collaborate as represented by FOAF. We have therefore introduced a new Gene ontology Annotation and navigation System. Colloquially referred to as GAS, it is based on Web 2.0 technologies with extended semantic capabilities that include Gene Ontology semantics, SCORM semantics, FOAF semantics, RSS syndication, aggregation semantics, as well as a useful and important gene ontology and annotation navigation system – Gene Ontology Navigation (GON). Our evaluation of the GAS prototype has proven to be extremely effective.

Acknowledgements

First of all, I would like to express my deep and sincere gratitude to my supervisor, Professor Jinan Fiaidhi, Ph.D., Department of Computer Science, Lakehead University. Her understanding, patience, encouragement and personal guidance have provided a good basis for this thesis. The financial support from her NSERC grant and the GA support of Lakehead University are also greatly appreciated.

I am deeply indebted to my co-supervisor, Professor Sabah Mohammed, Ph.D., Department of Computer Science, Lakehead University, for his detailed and constructive suggestions, as well as for his continued support throughout this thesis.

My sincere thanks are also due to Professor Ruizhong Wei, Ph.D., for his support as the Computer Science graduate coordinator.

Finally, I owe my loving thanks to my parents. Without their encouragement and faith in my abilities, it would have been impossible for me to finish my thesis.

Table of Contents

List of Tables

List of Figures and Illustrations

List of Symbols, Abbreviations and Nomenclature

Symbol	**Definition**
ASP	Active Server Pages
CMS	Content Management System
FOAF	Friend of a friend
GAS	Gene Annotation System
GO	Gene Ontology
IIS	Internet Information Services
JSP	Java Server Pages
LMS	Learning Management System
LO	Learning Object
LOM	Learning Object Metadata
PHP	Personal Home Page
REST	Representational State Transfer
RPC	Remote Procedure Call
RSS	Really Simple Syndication
SOAP	Simple Object Access Protocol
XML	Extensible Markup Language

Chapter One
Introduction

Learning Objects have been developing for years on traditional internet. Recently, a new web technology theory was proposed called Web 2.0. Unlike traditional web technology which only allows web users to accept information passively, Web 2.0 provides web users the option to actively modify web information. Different from Web 1.0, Web 2.0 provides web users an option of getting information actively. Instead of only reading web pages, web users can donate their own opinion or knowledge without modifying the resource itself. In other words, the resource itself is not changed, but other users still can see the modification made by web users. All the users subscribed to the web resource would be notified with the latest changes. To paraphrase, the difference between Web 1.0 and Web 2.0 is that the former is read-only while the latter is writable.

The advantage of Web 2.0 is important for learning objects to make individual learning and collaborative learning more interactive. Unfortunately, learning objects have not been transplanted on Web 2.0 so far.

This chapter introduces the concept of Learning Objects, as well as a list of popular Web 2.0 services and their explanations. The chapter also introduces the technology standard for Web 2.0 and clarifies Web 2.0 features as they pertain to Learning Objects.

1.1 Overview

Learning Objects are generally understood to be digital entities which are deliverable over the Internet, so that a number of people can access and use them simultaneously (as opposed to traditional instructional media, such as an overhead or a video tape, which can only exist in one place at a time). Moreover, those who incorporate Learning Objects can collaborate on and benefit immediately from new versions. This is a significant difference between learning objects and other types of instructional media existed previously (Wiley, 2000).

Learning objects brought new methods of study and research to the world of academics. With Learning Objects, courses are given to students according to their background, demands, and interest. The size of learning objects is not fixed. They can be merged into bigger Learning Objects or divided into several smaller Learning Objects easily to fit individual student's learning habit. Unlike traditional courses, which are available at a specific time, learning objects are accessible at any time when learners feel like they want to study as long as they can connect to internet.

1.2 Web 2.0 Services

Web 2.0 is a business revolution in the computer industry caused by the move to the internet as platform, and an attempt to understand the rules for success on that new platform. The most important of those rules is this: Build applications that harness network effects to get better when more people use them (O'Reilly, 2006).

If Web 1.0 connects people to web servers, Web 2.0 connects people to other people through the internet and allows people to collaborate with each other by exchanging knowledge and resources.

1.2.1 Blogs

A blog is a multi-purpose personal web page with low technique threshold. Blogs are customizable; users can change the blog's physical appearance by choosing a template from the service provider, or by creating their own custom template. Additionally, users can post various other applications to their blogs, such as flash videos, hit counters, and so on. Service providers usually include common word processing features, such as bold and italicized text, in their publishing interfaces. Blogs can be used to publish any type of content, ranging from personal entries to research progress.

Blog entries can also be commented on, if the blog's owner has enabled that option. After the owner finishes adding or editing a post, it becomes public and is displayed in a web page. This data is normally displayed in chronological order with earlier entries appearing below or on a separate page from the latest entry. If the option is enabled, other readers can post comments to any entry on the blog. Comments can be public to both the owner and other readers, or only viewable by the owner. The owner can also post his own comments and reply to others' comments. This interaction between writer and readers makes blogs popular on internet.

In addition, there are some other features most blog providers offer such as archiving, RSS feeds, and links. The blog's archive is a list of all the articles in the blog. By clicking the title, readers can view the article easily. The list can be sorted alphabetically, chronologically, or by keyword. RSS feed is a protocol can be used with RSS readers to notify all blog subscribers about the latest update. It will be discussed in-depth later in this chapter. Links can be considered to be the blog owner's bookmarks. Sometimes, blog hosts add a hyper link to others' blogs or web pages. A popular blogging feature that involves linking is called "trackback" (Anderson, 2007) – Trackback (or pingback) allows a blogger (A) to notify another blogger (B) that they have referenced or commented on one of blogger B's posts. When blog B receives notification from blog A that a trackback has been created, blog B's system automatically creates a record of the permalink of the referring post. Trackback only works when it is enabled on both the referring and the referred blogs. Some bloggers deliberately disable trackback as it can be a route in for spammers (Anderson, 2007).

As blogs become more popular and technology develops, new styles of blogging appears. A new feature that is growing more popular among bloggers is called "group blogging". Instead of blogs being held by a single person, these blogs are owned by a group of people. For example, all the students in a class of a university can use the same blog as an alumni; each member of this group maintains it as blogs being hosted by a single person. Another new style of blogging is blogging coterie. Different from group blogging, which is one blog owned by a group of people, blogging coterie is a group of single-owned blogs that are affiliated by a similar aspect of their blogs. This aspect can range from physical location (we all attended the same high school or university) to common interests (we all like to listen to the same kind of music) to business or research (we are all a part of the same research project). Anybody owning a blog can ask the group's leaders if they can join the coterie. It can also be regarded as an index of several other blogs. Similar to the publishing process of single user blogs, each article is listed in coterie blog groups in chronological order. Coterie blogs will direct readers to the individual blog where each article is from. Take Canadian Idol as an example: people can create a coterie for this TV show; Idol fans can get their information quickly by surfing a single group of Idol blogs rather than browsing them one by one.

1.2.2 Wikis

A wiki is a web page that can be viewed and modified by anybody with a Web browser and access to the Internet. The name's origin is from Wikipedia, the online encyclopaedia that works towards cataloguing all possible information in existence. Since Wikipedia's introduction, other imitators have appeared to focus on more specific branches of information. Any visitor to a wiki can change its content if they desire. While the potential for mischief exists, wikis can be surprisingly robust, open-ended, collaborative group sites (Keller, 2005).

Different than blogs, all wiki users can both contribute and benefit from it. They are able to create new wikis, read existing wikis, and modify existing wikis whenever they want. However, wikis generally have a history function, which allows previous versions to be examined, and a rollback function, which restores previous versions (Anderson, 2007).

Simply speaking, a wiki is an effective asynchronous web-based collaborative system, which grows with its users' knowledge and contribution.

1.2.3 Tags

As the as the amount of information increases on the internet, it becomes more and more difficult to accurately search for information. In the Internet's early public days, people would use keywords to search for pieces of information they desired. A keyword search simply searches the Internet for that word. For example, searching for the word "basketball" would bring up every page on the Internet that has the word "basketball" in it somewhere. This has proven to be inefficient, as a page may not be about basketball at all, but simply have the word on the page for reasons other than content, such as links or advertisements. Tags were created to help remedy this problem.

A tag is a keyword that is added to a digital object (e.g. a website, picture or video clip) to describe it, but not as part of a formal classification system (Anderson, 2007). Although "keyword" is used to define "tag", there is difference between them.

Keywords can be extracted from text documents by machine searches, but tags can only be set manually by users. In this way keywords are more convenient to use, but machines cannot extract keywords from non-text content such as images or video clips. On the other hand, tags allow users to label content with words that would easily and obviously describe it without the words being present.

In conclusion, keywords are objective while tags are subjective; keywords are accurate and tags are semantic; keywords are machine-oriented and tags are user-oriented; searching by keyword is more accurate and searching by tag is more human-intelligent.

Another usual way to classify information is its category. The difference between categories and tags is that the former can have levels, and the latter cannot. The word used as a category is fixed; authors have to choose a category from an available selection of common words instead of setting them as anything they choose.

Since tagging has become popular among internet users, the idea of tag clouds have appeared. A tag cloud is a group of related tags. Each tag within the group is set by different authors. They are listed together in different font, size and color according to their popularity, so people can easily tell which tag is more welcomed by users.

Keywords, categories, and tags all have advantages and disadvantages. Rarely do they conflict with each other, and no one can take the place of the other two completely. It is both possible and necessary for them to work together.

1.2.4 Multimedia Sharing

Multimedia sharing websites such as YouTube or Flickr provide a web platform for their users to upload / view multimedia resources like video or pictures. These popular services take the idea of the 'writeable' Web (where users are not just consumers but contribute actively to the production of Web content) and enable it on a massive scale (Anderson, 2007).

1.2.5 Syndication

Syndication is a process that allows subscribers to receive updates from web service providers through an XML-based data format, which is usually RSS or Atom. Syndication feeds can be read by aggregators and after adding feeds from their providers into aggregators which check the update automatically on a regular frequency, subscribers can read the summary of new content updated into that web page in the aggregators when the updates are available without opening the web page in browsers. Syndication feeds can bring the user text, audio and video content which will be discussed in-depth in the Podcasting and VODcasting section.

As mentioned in an earlier section of this chapter, most blog systems provide the feature of generating syndication feeds automatically when a new article is published.

RSS and Atom does a similar job concerning syndication, however they are different in their structure and grammar. Now, there is a trend of RSS and Atom aggregation, and some feed readers already support both RSS and Atom feeds.

1.2.6 Podcasting / VODcasting

Podcasting or VODcasting can be considered as an extension of the processes of traditional RSS process. Instead of receiving text feed from RSS providers (websites or blogs) they allow RSS feed bring audio or video content respectively.

Podcasting is the process of capturing an audio event, song, speech, or mix of sounds and then posting that digital sound object to a Web site or "blog" in a data structure called an RSS 2.0 envelope (or "feed"). Using specialized news readers like iPodder or iPodderX, users can subscribe to a Web page containing RSS 2.0 tagged audio files on designated web pages and automatically download these files directly into an audio management program on their personal computer like iTunes, Windows Media Player or MusicMatch. When a user synchronizes their portable audio device with their personal computer the podcasts are automatically transferred to that device to be listened to at the time and location most convenient for the user (Meng, 2005).

VODcasting (also called "vlogging") – the "VOD" stands for "video-on-demand" – is almost identical to podcasting. The difference is that the content is video instead of audio, and the content is more likely to be played on a laptop than a PMA (personal media assistant) due to their newness and relative expense (Meng, 2005).

1.3 Web 2.0 Technology Standards

This section introduces some technology standards which are used in implementation of this thesis.

1.3.1 XML

The Extensible Markup Language (XML) is a general-purpose specification for creating custom markup languages. It is classified as an extensible language because it allows its users to define their own elements. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet and it is used both to encode documents and to serialize data.[1]

There are many advantages of the XML format. First of all, XML is based on text supported by unicode which allows almost any information in any written human language to be communicated through it. Secondly, XML's self-documenting format describes structure and field names as well as specific values, and the strict syntax and parsing requirements make the necessary parsing algorithms extremely simple, efficient, and consistent. Thirdly, XML is heavily used as a format for data storage, processing and transfering, on the other hand, it can be updated incrementally. At last, it is platform-independent, therefore, it can be used cross different systems. [2]

[1] http://en.wikipedia.org/wiki/XML
[2] http://en.wikipedia.org/wiki/XML

1.3.2 XPath

The primary purpose of XPath is to address the nodes of XML 1.0 or XML 1.1 trees. XPath gets its name from its use of a path notation for navigating through the hierarchical structure of an XML document. XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values. (Berglund, et al., 2007)

XPath is usually used to query through XML files with specific conditions. This turns XML into the usage of database and gets useful information from it when necessary. It is used to implement searches through a blog system with search conditions selected by the user.

1.3.3 XPointer

The XPointer is intended to be used to provide a high level of functionality for addressing portions of XML documents. It is based on XPath and adds the ability to address strings, points, and ranges. It supports addressing into the internal structures of XML documents and external parsed entities and allows for examination of a document's hierarchical structure and choice of portions based on various properties, such as element types, attribute values, character content, and relative position. In particular, it provides for specific reference to elements, character strings, and other XML information, whether or not they bear an explicit ID attribute. (DeRose, Maler, & Daniel, 2002)

Since XPointer can connect XML elements together, it is useful when appending annotation to original resources without modifying them for more convenient mainteinance. Details will be introduced in Chapter 3.

1.3.4 REST vs. SOAP

REST is an architectural style which treats all web content as resources. It provides a unified set of interface to access resources – POST, GET, PUT and DELETE and use resource URI to identify resources. Therefore, "REST is considered the simpler of the two techniques. It exploits the World Wide Web (HTTP) protocol to communicate between computers, and initiating requests are usually in the form of URL's. From the library world, the most popular Web Service fitting this description is the Open Archives Initiative - Protocol for Metadata Harvesting. Another very good library example is called Search and Retrieve via URL (Morgan, 2005).

SOAP (Simple Object Access Protocol), a successor of RPC (Remote Procedure Call), is protocol for exchanging XML-based message through web site. It is not dependent on the World Wide Web as a transport mechanism. Requests can be made directly from one computer program to another, via telnet or SSH, via HTTP, or even through email. Unlike REST, SOAP requests as well as responses are encoded within a specific XML syntax called a SOAP envelope. For all these reasons, SOAP is seen as being more complicated and at the same time more flexible when compared to REST. SOAP, unlike REST, is formally supported by the World Wide Web Consortium (Morgan, 2005).

1.3.5 AJAX

AJAX is not a new programming language, but a technique for creating better, faster, and more interactive web applications. With AJAX, JavaScript can communicate directly with the server, using the JavaScript XMLHttpRequest object. With this object, JavaScript can trade data with a web server, without reloading the page.[3]

AJAX uses asynchronous data transfer (HTTP requests) between the browser and the web server, allowing web pages to request small bits of information from the server instead of whole pages.[4]

The AJAX technique makes Internet applications smaller, faster and more user-friendly. It is a browser technology independent of web server software.[5]

1.3.6 RSS

RSS is a format for syndicating news and the content of news-like sites, including major news sites like Wired, news-oriented community sites like Slashdot, and personal weblogs. But it's not just for news. Pretty much anything that can be broken down into discrete items can be syndicated via RSS. Once information about each item is in RSS format, an RSS-aware program can check the feed for changes and react to the changes in an appropriate way. RSS-aware programs called aggregators are popular in weblog systems. Most weblogs provide content available in RSS. An aggregator can help users keep up with all their favourite feed providers by checking their RSS feeds and displaying new items from each of them. (Pilgrim, 2002)

A RSS feed from the e-learning blog implemented for this project is given blow.

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
  <channel>
    <title>LU Gene Annotation Semantic Blog</title>
    <link>http://localhost/wordpress</link>
    <description>Wei Yuan's Thesis</description>
    <pubDate>Tue, 18 Mar 2008 03:11:45 +0000</pubDate>
    <generator>http://wordpress.org/?v=2.3.3</generator>
    <language>en</language>
    <item>
      <title>Mouse Anatomy by admin at Mon May 19 14:50:36 EDT
2008</title>
      <link>http://localhost/wordpress/wp-
includes/single.php?category=/anatomy/adult_mouse_anatomy&#38;ID=MA:0
000001&#35;annotation-1211223036</link>
      <description>mouse anatomy is usually found on mouse
</description>
```

[3] http://owl.english.purdue.edu/owl/resource/560/10/
[4] http://owl.english.purdue.edu/owl/resource/560/10/
[5] http://owl.english.purdue.edu/owl/resource/560/10/

```
      </item>
    </channel>
  </rss>
```

1.4 Features of Learning Object for Web 2.0

1.4.1 Collaborative Learning

Collaborative learning is an emerging educational method. It is a student-centered environment, in which students are responsible for their studying progress by constructing knowledge with other students or teachers. Similarly to other Web 2.0 services, users of collaborative learning can contribute and benefit from the system.

1.4.2 Social Blogs as Learning Object

The concept of social blogs is derived from social networks. Social blogs treat each member as a node and connect members through ties (relationship) as social network. However, there are a few points that fundamentally distinguish it from a social network. First of all, within a social network, each node represents a real human; but within a social blog, each node stands for a blog, which can be hosted by any blog service anywhere on the Internet. Social blog systems are not necessary to provide blog services. In addition, a social network connects nodes by hobby, political opinions, or other topics; social blogs connects nodes by blog category or blog tags.

As a Learning Object platform, social blogs meet all feature requirements – tags, syndication feeds, PODcasting, resource sharing, etc. These features will be discussed in depth in later chapters.

1.4.3 Folksonomies for Free Tagging

In collaborative web, people contribute their knowledge, research, and opinions by publishing the data over the Internet. To save other users' time when searching for information they need, authors also add tags to their resources. Other users can modify previously published resources to make them more accurate. However, this is not completely collaborative. If the original author of a resource set an inaccurate, misleading, or incorrectly spelled tag, it will be impossible to find with regular searches and nobody will be able to correct it. Under this situation, folksonomy is introduced.

Folksonomy is an extension of tagging. Folksonomies allows other users to modify the resource's tags. In this way, the tags are more accurate via the corrections of others should errors arise.

1.4.4 Annotation

Annotation is a process through which people contribute their own knowledge to others resources. It's also a method to achieve learning collaboration through network. Original resources can be annotated without modification; therefore, it is easy to restore the resource before its annotation.

Annotation theory, and its relation to Gene Ontology Annotation, will be discussed in-depth in later chapters.

1.5 Summary of Thesis

Due to Web 2.0's propagation, weblogs, collaborative e-learning systems, and FOAF networks are popular and useful applications by themselves. Is it possible to integrate these applications into one system?

This question will be explored in regards to a specific field of biology. Throughout the course of this project, the Gene Annotation Semantic Blog system (GAS Blog system) was developed to make gene annotation easier. It is an extension of regular weblog systems by adding the support of several semantics such as Gene Ontology semantics, SCORM semantics, syndication & aggregation semantics, and FOAF semantics.

Chapter 1 introduces Web 2.0 and its popular applications, as well as technologies related to them. Chapter 2 describes the GAS Blog from a general point of view, compares it with existing gene ontology annotation systems, and discusses their advantages as well as disadvantages. Chapter 3 explains the usage scenario, as well as detailed architecture and implementation of the GAS Blog. In Chapter 4, gene ontology navigation will be introduced. Chapter 5 describes future work to be done on this project.

Chapter Two
Gene Annotation Support System for the Web 2.0 e-Learning

Annotation is useful to people's learning experience. It can be used in person or groups. For personal use, it works as reading notes to make readers themselves understand documents better or remind themselves with some important information for the future. For example, a lot students like writing notes on blank area of textbooks according to teachers' lecture; therefore, they can be reminded that some certain section should be paid more attention to when they read textbook next time. As among groups, it allows group members collaborative on same documents by letting each person add notes to documents. Still take students and textbook as the example, sometimes students share the notes by borrowing textbook from each other which is a process of collaborative annotation.

Annotation has been introduced to computer for a long time both personally and collaboratively. For personal use, there are lots of documents organization tools such as Personal Document Organizer which allows users to add description to both text and multimedia files on their computers. For collaborative use, the most popular example is Microsoft Word comment feature. It provides an easy and efficient way to collaborate people's comment in team-working environment. Annotation appears on the margin of pages with the comment author's name and it becomes visible to other team members with full access to it including reading or modification.

After Internet becoming popular, web annotation is used to add annotation to web resources. Over years' development, many annotation websites appear over Internet, they provide web annotation services to allow registered users add annotation on resources (text, images, multimedia, etc.) from other websites, and share annotation with other registered users.

This chapter introduces web annotation in details, discusses about existing web annotation systems, and describes web 2.0 annotations as well as how it differs from existing annotation systems. In later sections, this chapter talks about gene ontology and explains why use a learning blog as a platform to implement gene ontology annotation system.

2.1 Traditional Gene Ontology Annotation

Gene Ontology Annotation is the process of assigning GO terms to gene products (Nature, 2000). This process is usually achieved by manual annotation and electronic annotation.

- **Manual Annotation**: Manual assignment of GO terms by annotators using published literature. Associations that are made manually are given one of ten evidence codes that describe what evidence supports the annotation.[6]

---

[6] http://www.ebi.ac.uk/GOA/newto.html

- **Electronic Annotation**: electronic annotation systems use existing information within database entries, including Swiss-Prot keywords (SPKW2GO), Enzyme Commission numbers (EC2GO) and cross-references to InterPro (InterPro2GO) and HAMAP (HAMAP2GO), which are manually mapped before. Electronically combining these mappings with a table of matching UniProt entries generates a table of associations. For each GOA association, they provide an evidence code, which summarizes how the association is made. Associations that are made electronically are labelled as 'inferred from electronic annotation' (IEA).[7]

Manual annotation is the basis of electronic annotation, and electronic annotation can also be considered as a procesure of reducing redundant gene ontology annotation. This thesis is focus on developing a system (GAS Blog) to assist manual annotation procesure.

Currently, the manual annotation systems can be divided into three categories: offline application, Client-Server application, web search engine, portal, and FTP server.

2.1.1 Offline Application

Offline gene ontology application let users view and modify gene ontology files. The advantage of applications in this categoty is that they provide users a frendly interface to work on, however, the disadvantages is that users have to download ontology files from GO portal or FTP server before viewing or doing modification, and after annotators finish modifying the file, they have to go back to the GO portal or FTP server to upload their work. An example applications in this categories are: COBrA.

2.1.2 Client-Server Application

Client-Server gene ontology application goes a step further than offline gene ontology applications. It connects to a server to retrieve gene ontology for users to view instead of letting users download the ontology files beforehand. However, the disadvantage of it is that users still can only view the ontology data and can not add annotation to it. Moreover, users are still isolated from each other and have no collaboration in this process. An example application in this category is DYNGO.

2.1.3 Web Search Engine

Web search engines are web application. To use this kind of GO application, users have to fill a search form to indicate several query condition, and results under the condition will be displayed in web pages. Users can view them by clicking the link of each result.

GO search engine is useful for users who only want to view specific gene ontology annotations. It does not allow annotators to add or modify annotation to gene ontology and does not have enough information for users who want to know the latest annotation, and users have to search if there is any update to specific annotation of their interest on a regular basis.

[7] http://www.ebi.ac.uk/GOA/newto.html

On the other hand, current gene ontology search engines are inefficient because of the top-down search mechanism they are using. Users have to go through level by level to reach the information they are looking for, and this top-down process may take some time. An example application in this categories are: AmiGO.

2.1.4 Portal

Gene ontology portals are web pages that provide users to download the latest modified ontology files or upload the one users worked on. After downloading ontology files from gene ontology portals, users can view or modify the files using offline gene ontology applications.

There are several disadvantages for portal annotation system:

First of all, regular users can only receive the information published by gene ontology annotation organizations, and are not able to donate their knowledge and collaborate with other users. There is no connection between users, and they are isolated with each other.

Secondly, bioinformatics database resource groups publish the information on webpage of portal. Portal is an inefficient way to publish information. Users have to check the portal on regular basis to make sure that they can get the latest update.

At last, the annotation information is in text format, and newly added information is mixed with the old one. Users have to download the annotation file and go through all the information and find out the updates by themselves.

An example application in this category is BBOP (Berkeley Bioinformatics and Ontology Project).

2.1.5 FTP Server

Similarly to portal, FTP server is another method allows users to download gene ontology files. Instead of downloading from web pages like portals, it let users access ontology files from FTP servers. Therefore, FTP server share the similar disadvantage with portals that users can not find the latest change inside a ontology file easily.

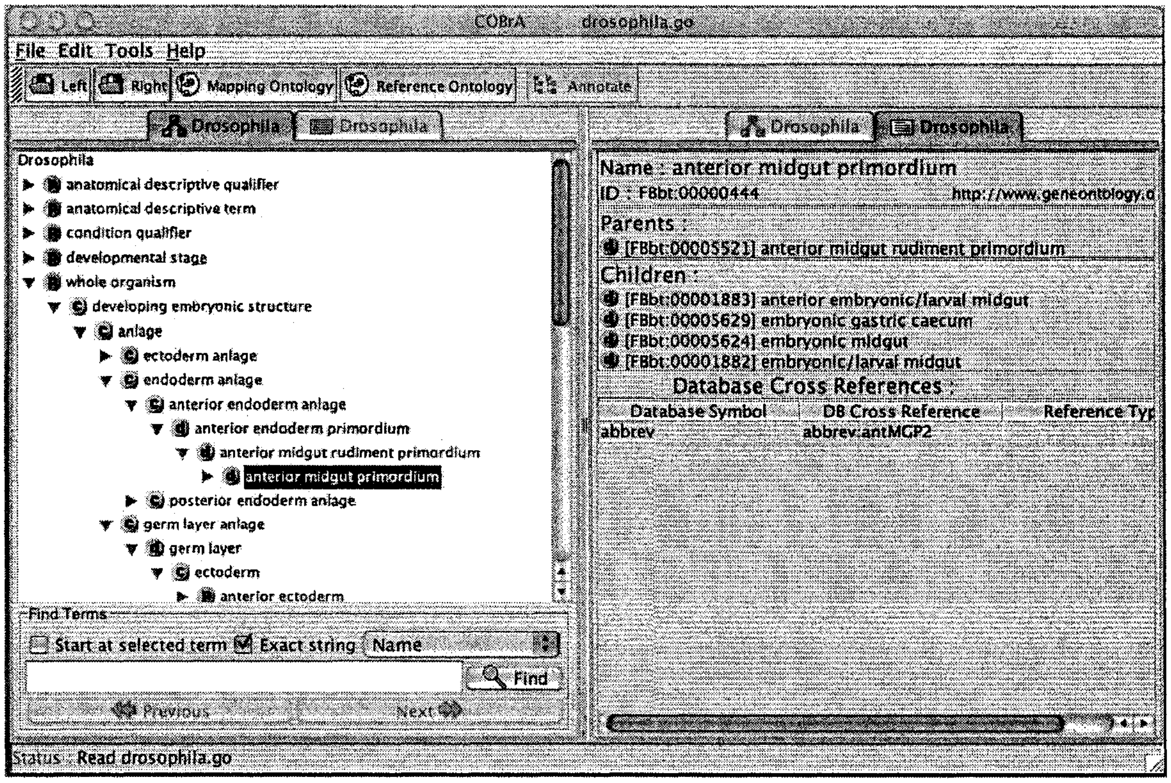
2.2 Existing Gene Ontology Annotation Systems

Although there are many Gene Ontology annotation systems available on market, they can be divided into the categories mentioned in earlier section of this chapter. This thesis selects a representative application in each category to compare the advantages and disadvantages of them.

2.2.1 COBrA

COBrA is an ontology browser and editor for GO and OBO ontologies. It has been specifically designed to be usable by biologists to create links between ontologies, and has the following features. (Stuart, 2005)

- allows drag-and-drop editing of GO ontologies (Stuart, 2005)
- supports mapping between two ontologies (Stuart, 2005)
- supports translation to OWL and other Semantic Web languages (Stuart, 2005)



```
Drosophila
▶ anatomical descriptive qualifier
▶ anatomical descriptive term
▶ condition qualifier
▶ developmental stage
▼ whole organism
    ▼ developing embryonic structure
        ▼ anlage
            ▶ ectoderm anlage
            ▼ endoderm anlage
                ▼ anterior endoderm anlage
                    ▼ anterior endoderm primordium
                        ▼ anterior midgut rudiment primordium
                            ▶ anterior midgut primordium
                ▶ posterior endoderm anlage
            ▼ germ layer anlage
                ▼ germ layer
                    ▼ ectoderm
                        ▶ anterior ectoderm
```

Name : anterior midgut primordium
ID : FBbt:00000444 http://www.geneontology.o

Parents :
[FBbt:00005521] anterior midgut rudiment primordium

Children :
[FBbt:00001883] anterior embryonic/larval midgut
[FBbt:00005629] embryonic gastric caecum
[FBbt:00005624] embryonic midgut
[FBbt:00001882] embryonic/larval midgut

Database Cross References :

Database Symbol	DB Cross Reference	Reference Typ
abbrev	abbrev:antMGP2	

Status : Read drosophila.go

Figure 2.1 Interface of COBrA[8]

As other offline gene ontology applications, the advantage is that they provide users a frendly interface, however, the disadvantages is that users have to download ontology files from GO portal or FTP server by themselves, and after annotators finishing modifying the file, they have to go back to the GO portal or FTP server to upload their work.

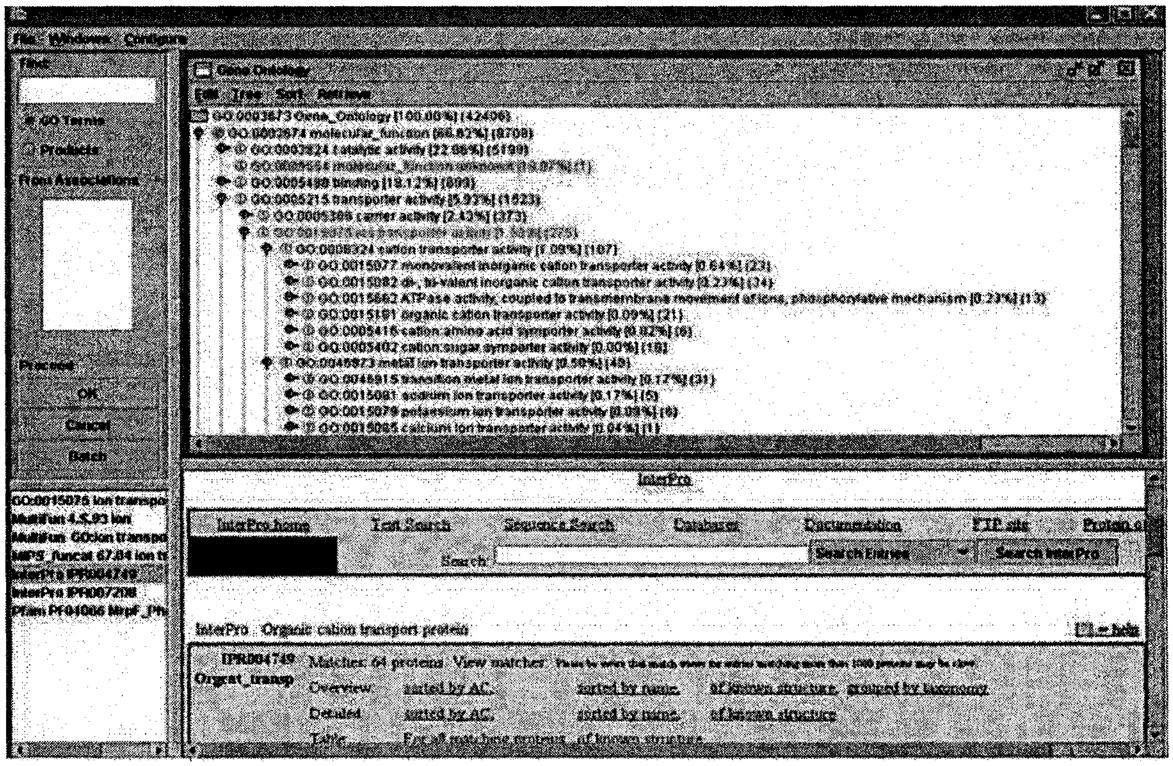
2.2.2 DYNGO

DYNGO is a standalone package that provides browsing functionality of gene ontology. DYNGO also allows users to load a list of entities and retrieve the corresponding GO annotations. It enables users to retrieve gene or gene products that hold similar annotations. The retrieved result is shown in a tree organized according to GO hierarchies, and the tree can be manipulated dynamically by sorting and changing orientation.

---

[8] Image retrieved from http://www.xspan.org/cobra/index.html

DYNGO can also be used for Microarray data analysis using GO annotations and for other applications. (Liu, Hu, & Wu, 2005)



```
GO:0003673 Gene_Ontology [100.00%] (42406)
  GO:0003674 molecular_function [66.82%] (8708)
    GO:0003824 catalytic activity [22.86%] (5199)
    GO:0005554 molecular_function unknown [0.07%] (1)
    GO:0005488 binding [18.12%] (899)
    GO:0005215 transporter activity [5.93%] (1523)
      GO:0005386 carrier activity [2.43%] (373)
      GO:0015075 ion transporter activity [1.58%] (275)
        GO:0008324 cation transporter activity [1.09%] (107)
          GO:0015077 monovalent inorganic cation transporter activity [0.64%] (23)
          GO:0015082 di-, tri-valent inorganic cation transporter activity [0.23%] (34)
          GO:0015662 ATPase activity, coupled to transmembrane movement of ions, phosphorylative mechanism [0.23%] (13)
          GO:0015101 organic cation transporter activity [0.09%] (21)
          GO:0005416 cation:amino acid symporter activity [0.02%] (6)
          GO:0005402 cation:sugar symporter activity [0.00%] (18)
        GO:0046873 metal ion transporter activity [0.50%] (49)
          GO:0046915 transition metal ion transporter activity [0.17%] (31)
          GO:0015081 sodium ion transporter activity [0.17%] (5)
          GO:0015079 potassium ion transporter activity [0.09%] (8)
          GO:0015085 calcium ion transporter activity [0.04%] (1)
```

```
GO:0015075 ion transpo
MultiFun 4.S.93 ion
MultiFun GO:ion transpo
MIPS_funcat 67.04 ion tr
InterPro IPR004749
InterPro IPR007208
Pfam PF04006 Mrpf_Ph
```

InterPro Organic cation transport protein

IPR004749 Orgcat_transp Matches: 64 proteins View matches
Overview sorted by AC. sorted by name. of known structure. grouped by taxonomy.
Detailed sorted by AC. sorted by name. of known structure.
Table For all matching proteins of known structure.

Figure 2.2 Interface of DYNGO[9]

Client-Server gene ontology application such as DYNGO has the advantage of friendly user interface, retrieve gene ontology data directly from network instead of letting users to download from portals or FTP server as offline gene ontology application such as COBrA. However, the disadvantage of it is that users still can only view the ontology data and can not add annotation to it. Moreover, users are still isolated from each other and have no collaboration in this process.

2.2.3 AmiGO

AmiGO provides an interface to search and browse the ontology and annotation data provided by the GO consortium. Users can search for gene products and view the terms with which they are associated; alternatively, users can search or browse the ontology for GO terms of interest and see term details and gene product annotations. AmiGO also provides a BLAST search engine, which searches the sequences of genes and gene products that have been annotated to a GO term and submitted to the GO Consortium. (Nature, 2000)

[9] Image retrieved from http://gauss.dbb.georgetown.edu/liblab/dyngo.html

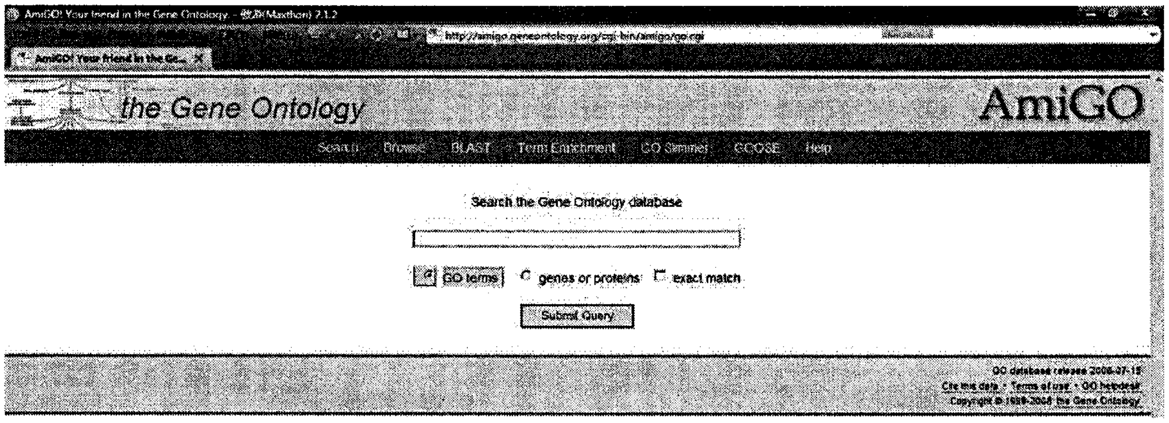


Figure 2.3 Interface of AmiGO[10]

GO search engine like AmiGO is useful for users who only want to view specific gene ontology annotations. It does not allow annotators to add or modify annotation to gene ontology and does not have enough information for users who want to know the latest annotation, and users have to search if there is update to specific annotation of their interest on a regular basis.

On the other hand, current gene ontology search engines are inefficient because of the top-down search mechanism they are using. Users have to go through level by level to reach the information they are looking for, and this top-down process may take some time to complete.

2.2.4 BBOP Portal

BBOP Portal is a page is for downloading OBO ontologies in a variety of formats as well as the reports of ontologies. The data is derived automatically from the primary sources, available from the main OBO website.[11]

---

[10] Image retrieved from http://amigo.geneontology.org/cgi-bin/amigo/go.cgi
[11] http://www.berkeleybop.org/ontologies/

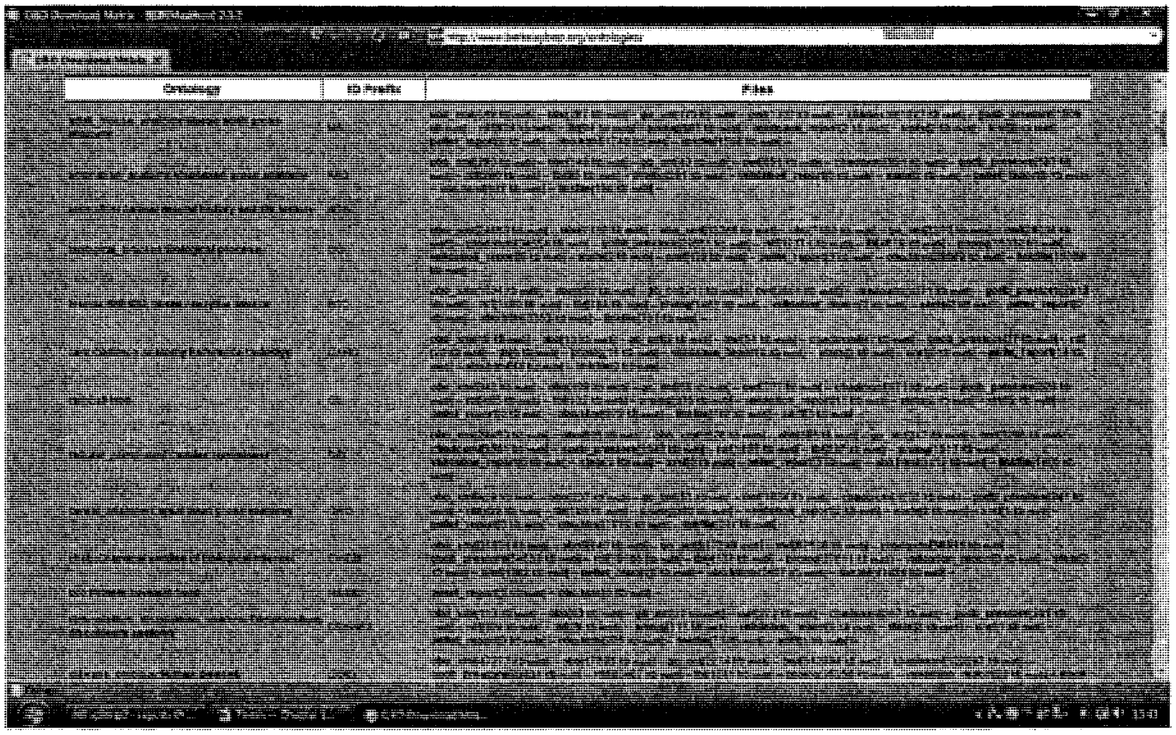


Figure 2.4 Interface of BBOP Portal[12]

There are several disadvantages for portal annotation system such as BBOP Portal:

First of all, regular users can only receive the information published by those groups, and are not able to donate their knowledge and collaborate with other users. There is no connection between users, and they are isolated with each other.

Secondly, bioinformatics database resource groups publish the information on webpage of portal. Portal is an inefficient way to publish information. Users have to check the portal on regular basis to make sure that they can get the latest update.

At last, the annotation information is in text format, and newly added information is mixed with the old one. Users have to download the annotation file and go through all the information and find out the updates by themselves.

2.2.5 Summary of Existing Gene Ontology Annotation System Comparison

These five categories of current gene ontology annotation system are compared in four criteria:

1. Check latest change inside GO file: to see if the GO annotation system indicate users which term in a gene ontology file is updated recently;

[12] Image retrieved from http://gauss.dbb.georgetown.edu/liblab/dyngo.html

2. Automatic data retrieve: to check if the GO annotation system can retrieve GO data automatically for users instead of letting them download from portals or FTP servers;

3. Notification for latest update: to see if the GO annotation system can notify users whenever the update is available instead of letting users check manually on a regular basis;

4. User collaboration: to see if the GO annotation system allows users to share their knowledge or experience and collaborate with each other.

With Web 2.0 technology, these criteria can be achieved, and they are the most popular usage for Web 2.0.

	Offline Application	**Client-Server Application**	**Web Search Engine**	**Portal**	**FTP Server**
Check Latest Change inside GO file	No	No	No	No	No
Automatic Data Retrieve	No	Yes	Yes	No	No
Notification for latest update	No	No	No	No	No
User Collaboration	No	No	No	No	No

Table 2.1 Existing GO Annotation Systems Comparison Table

For all the reasons discussed above, a new generation of gene ontology annotation system is very necessary to help biologists and make their work more efficient.

2.3 Towards Gene Ontology Annotation Support System for a learning Blog

There are many advantages of weblogs being a Web-based Support System. Weblogs provide an excellent new channel for research discussion, communication, and collaboration. A Web-based research support system may gain from the diary feature in blogs to support researchers' daily activities. Blogs also provide a new dissemination channel for research results. Blog data management include organizing, classifying, backup, and retrieving blog contents (Yao J. , Supporting Research with Weblogs: A Study on Web-based Research Support Systems, 2006).

Learning blogs are blogs used for collaborative learning either in public or among groups such as the Learning Blog by Alex Ragone (2005): Exploring Learning through Blogging13. It is a learning blog specialized in web technologies. From the example, the difference between learning blogs and other blogs is the content. A learning blog has all the features a blog should have.

[13] http://www.learning-blog.org/

One of the most innovative features of the system developed in this thesis is taking the advantage from the new web technology to benefit Gene Ontology annotation. The idea is building the system on top of a social blog. As mentioned in Chapter 1, social blog is a system that connects distributed nodes through their relationship. With this connection, separated users can be associated to collaborate with their work of annotation. Gene Ontologies can be annotated collaboratively by people who registered to the blog instead of by a certain organization only.

To solve the disadvantages of portal style annotation system mentioned in earlier section, a collaborative learning system based on Web 2.0 needs to be designed for Gene Ontology annotation. Since it is aimed to be a collaborative e-learning system, users can share their resources and collaborate with each other for Gene Ontology annotation in real time without waiting for bioinformatics database resource groups to publish. It is unnecessary for users to check the system on a regular basis any more. Semantic features of the system can record users' interest over gene ontology categories and notify them whenever a new annotation or new gene is available in any of their interested categories after subscribing to feeds. The annotation is no longer saved in format of text, and it is in XML format with timestamp, in this way, users can easily find the latest change without browsing through the entire annotation file. Moreover, the system also provides an efficient navigation to help users find out the gene ontology or annotation which match the navigation command inputted by them.

As for the disadvantages of annotation systems with top-down search mechanism, gene ontology e-learning annotation blog (GAS) introduced in the thesis provides a more efficient and advanced navigation engine with bottom-up mechanism. Therefore, users do not have to go through level by level from top to reach the data at the bottom to save the time spend on obtaining useful information.

To solve these disadvantages of current gene ontology annotation systems, a semantic collaborative blog system for gene learning objects called "Gene Annotation Semantic Blog" (GAS Blog) is developed in the thesis.

The features of GAS Blog are as followed:

- Adding new gene ontology;
- Annotating to existing gene ontology;
- Easily displaying gene ontology term annotation which are added or modified recently;
- Automatically displaying the information from gene ontology file on web pages;
- Notifying users the recent update of their interest through RSS;
- User collaboration;
- Create learning object for each gene ontology term;

- User FOAF network in order to make collaboration more efficient;
- Bottom-up navigation mechanism for faster and more accurate and relevant search result.

2.3.1 GAS Blog Overal Architecture

GAS Blog is consisted by modules of Learning Object Wrapper, Syndication Processor, GO Data Output processor, User Data Collector, GO Creator, Annotation Processor, FOAF Information Collector, and Graphical FOAF Generator.

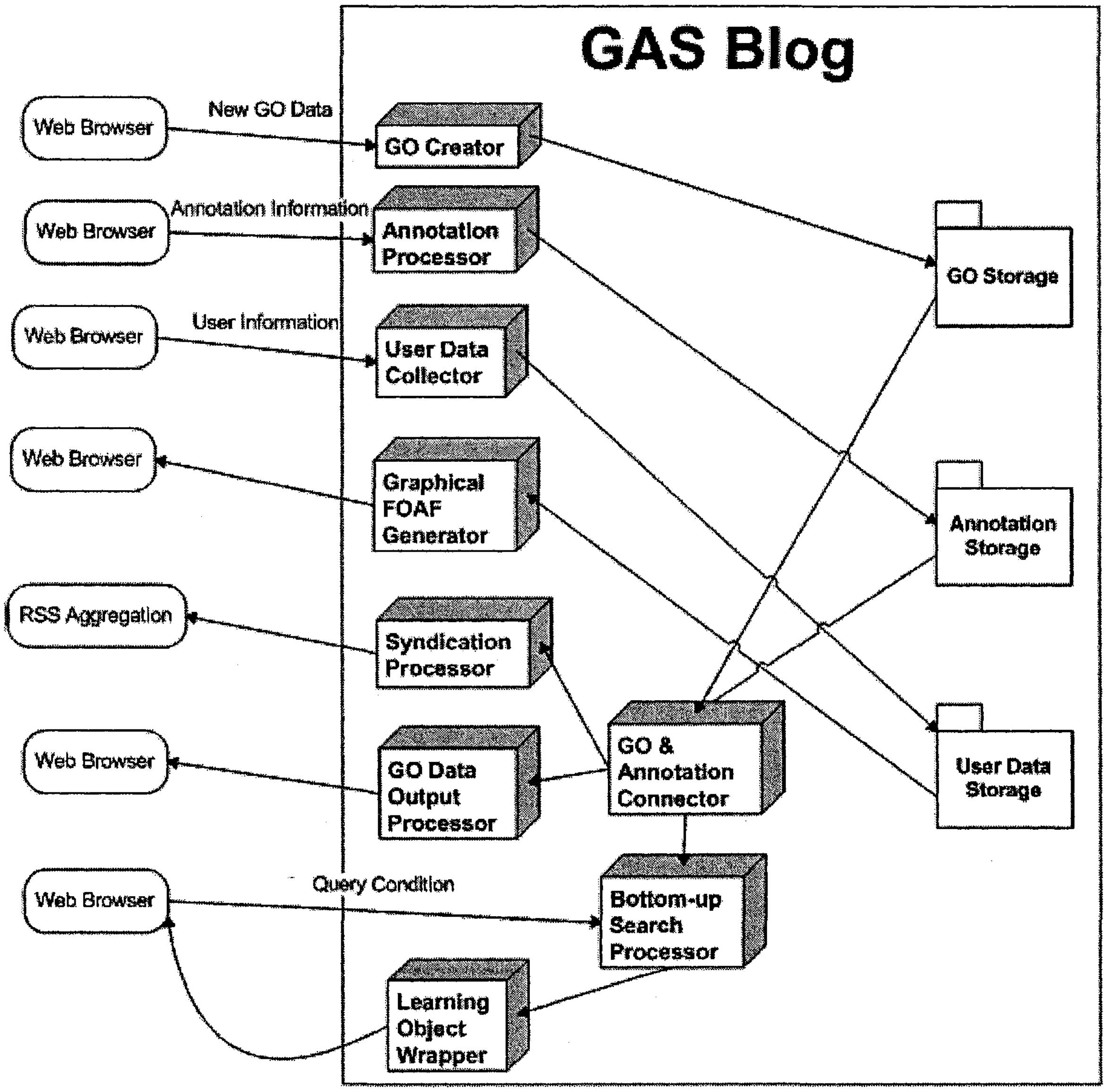


Figure 2.5 GAS Blog Overall Architecture

- GO Creator: GO Creator creates a new gene ontology according to data retrieved from users' web browser;
- Annotation Processor: Annotation Processor annotates to gene ontology according to data retrieved from users' web browser;
- User Data Collector: User Data Collector collects user information from users such as user name, email, interested gene ontology categories, groups willing to join, etc.;
- Graphical FOAF Generator: Graphical FOAF Generator generates graphical FOAF network information;
- GO & Annotation Connector: GO & Annotation Connector prepares data for Bottom-up search Processor by combining the information of gene ontology, annotation, and FOAF network;
- Syndication Processor: Syndication Processor gets the well-wrapped data from Learning Object Wrapper and generates syndication for RSS aggregation;
- GO Data Output Processor: GO Data Output Processor gets the data from gene ontology and annotation, and displays them on web page;
- Learning Object Wrapper: Learning Object Wrapper gets the gene ontology and annotation data and wraps them as learning objects;
- Bottom-up Search Processor: Bottom-up Search Processor searches through gene ontology, annotation, and FOAF network using bottom-up search mechanism basing on data prepared by Data Combination Module.

As shown in Figure 2.5, all the data in GAS Blog are stored in GO storage, annotation storage, and user data storage. Among them, GO storage and annotation is XML-based file format, and user data storage is in MySQL database. They are the core of GAS Blog and as a media for different modules to exchange data, communicate and interact with each other.

GO Creator, Annotation Processor, User Data Collector, and FOAF Information Collector are inputting modules of GAS Blog. They collect data inputted by users and store them into related storage for other modules to use.

GO & Annotation Connector is a media module between data storage and output modules; it prepares the data from storage for output purpose. Graphical FOAF Generator, Syndication Processor, GO Data Output Processor, and Learning Object Wrapper are output modules of GAS Blog. They make use of data prepared by GO & Annotation Connector and deliver them to users.

Detailed architecture, usage scenario, and development of most modules will be introduced in next chapter. Bottom-up Search Processor and Learning Object Wrapper will be discussed in Chapter 4.

Chapter Three
The Semantic Blog Support System Architecture

Collaborative Infrastructure for Sharing & Annotating Gene Learning Objects

Semantics Blogs are blogs enriched with semantic, machine-understandable metadata (Möller, Breslin, & Decker, 2005). It is an extension of regular blogs. As described in Chapter 2, current Gene annotation systems have the following disadvantages: they do not allow users to collaborate with each other; users can not communicate with each other; it is inefficient to publish information on them; it is difficult for users to find the latest annotations to existing genes; search mechanism is inefficient as well. To solve these disadvantages, a semantic collaborative blog system for gene learning objects called "Gene Annotation Semantic Blog" (GAS Blog) is developed alone with this thesis.

GAS Blog is built on a regular open source blog system – WordPress and added with some semantic features such as describing regular Gene Ontologies as SCORM Learning Objects for collaboration, FOAF semantics, Syndication & Aggregation semantics, and bottom-up searching mechanism.

This chapter will focus on GAS Blog architecture and introduce its semantic features mentioned above in details. System requirements and installation instruction of GAS will be explained in Appendix A.

3.1 GAS Support System Design Criteria

Design criteria are one of the important issues for a support system as Yao (2005) put it.Design criteria of GAS can be summarized as follows:

- System Users: GAS Support System is designed for gene ontology annotators and the people who wish to browse gene ontologies and their annotation as well as cooperate with others in a collaborative gene ontology support system.
- Searching Information in Support System: GAS Support System includes a gene navigation service to allow users navigate gene ontology, and annotation information through navigation command.
- System User Management: users can be divided into three categories: friends, users in the same group / research field, users from different group / research field. Users can view their friends' information through a graphical FOAF network, collaborate with users in the same user group, and view other users' work in GAS Support System.
- Capability of Giving Feedback in Support System: GAS Support System allows registered users to annotate or comment on existing gene ontologies and each gene ontology can have unlimited annotations or comments as research feedback in Support System.

3.2 GAS Support System Requirements & Design Policies

According to features GAS Blog is planning to achieve, GAS Blog system requirements can be summarized as below.

- Data should be accompanied with machine-understandable metadata.
- Users should be able to publish new blog entries and the content should be stored in the same schema structure.
- Users should be able to annotate to existing blog resources or entries.
- Users should be able to add others users as their friends.
- Users should be able to create new user groups or join existing groups created by others for group-wise collaboration.
- Other than group-wise collaboration, cross-groups collaboration should be enabled.
- Aggregation should be customized to users personal interest, instead of broadcast everything to everybody.

To meet those requirements, design policies are:

Integration of metadata and blog data. GAS Blog data should be stored along with metadata, which is in XML-based format files instead of in database as regular blog systems.

Annotations and XML sequencing. Annotation facility should be developed for collaboration through GAS Blog. For maintenance reason, annotation should also be stored separately with original blog resources / learning objects, and XML sequencing connects annotations with the blog resources which they were annotated to.

Combination of blog system and FOAF network. FOAF network metadata should be added to annotation schema to indicate annotator information and group-wise collaboration.

Extended syndication feed. Syndication feed has become a popular way to deliver web contents with metadata to subscribers. Standard syndication schema only contains the information for general purpose, and some important professional information might be missing during process of delivery. However, XML-based syndication feed is extendible. Extended syndication feeds brings more professional information than regular blog systems provide after embedding semantic metadata of that research field into them, such as embedding Gene Ontology annotation metadata in this thesis.

Narrowcast – customized aggregation. For a blog system, not all subscribers are interested in every piece of information published on it. Broadcast may force users to receive the "junk" information and waste time on them. Narrowcast can be achieved by allowing subscribers customize categories of information on GAS Blog, and only deliver the information of their interests.

3.3 Selecting a Programming Framework for GAS Blog

After identifying system requirements of GAS Blog, a programming framework should be selected for implementation. Since GAS Blog is aimed to be a web application, only server side scripting languages programming framework are considerable. Currently, dominant languages in this field are ASP, JSP, and PHP.

3.3.1 ASP

Active Server Pages (ASP) is Microsoft's first server-side script engine for dynamically-generated web pages. It was initially marketed as an add-on to Internet Information Services (IIS) via the Windows NT 4.0 Option Pack, but has been included as a free component of Windows Server since the initial release of Windows 2000 Server. Programming ASP websites is made easier by various built-in objects. Each object corresponds to a group of frequently-used functions useful for creating dynamic web pages. Mixing traditional ASP and Microsoft's .NET technology, ASP .NET allows web application to be more intelligent and complicated.[14]

Most ASP pages are written in VBScript, but any other Active Scripting engine can be selected such as Jscript (Microsoft's implementation of ECMAScript) and PerlScript (a derivative of Perl), and other third-party installable Active Scripting.[15]

Although other Active Scripting engines strengthen ASP, the limitations of ASP are:

Cross system compatibility: ASP can be only run on server (IIS) with Windows operation systems, which build a barrier for Linux or UNIX servers.

Server ownership issues: ASP can be extended with many Active Scripting engines; however, this is only helpful to organizations that have their own servers. Most server hosting service providers only provide basic ASP with VBScript, and do not permit users to install Active Scripting engines for security concerns. Therefore, organizations that do not have their own servers can not benefit from the extension of active scripting engines.

3.3.2 JSP

JavaServer Pages (JSP) is a technology for developing web pages that include dynamic content. Unlike a plain HTML page, which contains static content that always remains the same, a JSP page can change its content based on any number of variable items, including the identity of the user, the user's browser type, information provided by the user, and selections made by the user (Bergsten, 2003).

---

[14] http://en.wikipedia.org/wiki/Active_Server_Pages
[15] http://en.wikipedia.org/wiki/Active_Server_Pages

Unlike ASP introduced above, JSP can run on server with various operation systems. However, the disadvantage of JSP is that it is a heavy weight framework, and costly on hosting. A detractor of the juggernaut might describe it as being for people with more money than sense.[16]

3.3.3 PHP

PHP is a widely-used general-purpose scripting language that is especially suited for web development and can be embedded into HTML. It generally runs on a web server, taking PHP code as its input and creating web pages as output. It can be deployed on most web servers and on almost every operating system and platform free of charge.[17]

Like JSP, PHP is system independent as well; it runs on servers of different operations systems. The big difference between JSP and PHP is that PHP is a light weight open source framework. Implementation of PHP is more direct-forward than JSP and there are a lot free or low cost hosting service available for organisations who do not have their own server and do not want to spend a fortune on it.

3.3.4 Comparison of Programming Framework

According to the brief description of each framework, cross system compatibility and especially server ownership issues make ASP not a good choice for GAS Blog. On the first thought, JSP meets the requirements for developing GAS Blog; however, the hosting service cost should also be concerned and heavy weight framework may cost more development effort than light weight framework – PHP. After all the consideration, PHP is the best choice for GAS Blog. Comparison of these programming frameworks is summarized into the following table.

	ASP	JSP	PSP
Cross-System	No	Yes	Yes
Framework Size	Heavy-weight	Heavy-weight	Light-weight
Cost	High	High	Low
License	Freeware	Freeware	Open Source

Table 3.1 Programming Framework Comparison Table

3.4 Selecting a Weblog Platform for GAS Blog

There are two ways to build a weblog system: developed from scratch or modify from an existing open source blog system. The latter is a wiser option since the former takes more time and efforts on development.

[16] http://training.gbdirect.co.uk/courses/php/comparison_php_versus_perl_vs_asp_jsp_vs_vbscript_web_scripting.html
[17] http://en.wikipedia.org/wiki/PHP

There are hundreds of blog platform on markets now. However, blog systems that can be used to build GAS Blog **MUST** meet the following requirements:

- Open source. GAS Blog is not a regular blog, although it is built on one. There are many features to be added, and only open source projects are allowed to be modified.
- Written in PHP. As introduced in earlier section of this chapter, GAS Blog should be developed in PHP, which means the weblog platform needs to be written in the same web scripting language or programming framework.
- Lightweight blogging system. A lightweight blogging system requires less development efforts. As mentioned earlier in this thesis, GAS Blog is a blog system for gene annotation with semantic features. Therefore, the blogging system it build on does not have to be the most powerful system or a complete website solution, but serves GAS Blog development requirements and design policies well.

Currently, there are hundreds of weblogs or blogging systems on market. Among them, the most famous and popular ones are: Blogger, MSN Space, Movable Type, Expression Engine, WordPress, etc.

3.4.1 Blogger & MSN Spaces

Blogger and MSN Spaces are both free blog service through the Internet. Blogger and MSN Spaces can be discussed together here because they have the same reason for not being able to be used to build GAS Blog. Although they are provided by different organizations, they are similar in essence. They are both free blogs but only free for users to use. People can register to them and publish information through them conveniently. The fatal limitation of them is that there is no way to access the source code of them legally and modify the code to meet the requirements of building GAS Blog.

3.4.2 Movable Type

Movable Type is a free weblog publishing system[18]. It is mostly written in Perl with some PHP. As a weblog platform, it has all the features as other regular blog systems, blogging, entries management, weblog theme template, archiving, commenting, plug-in support, etc. On the other hand, it is system-independent that can be installed on servers with various operation systems installed.

Since Movable Type is written in both Perl and PHP, communication between two languages may cause some problems, and standard PHP does not support the interaction with Perl code, under this situation, additional libraries need to be installed. As discussed in the section of ASP, web hosting service providers, especially free or low cost providers,

[18] http://en.wikipedia.org/wiki/Movable_Type

not always have additional libraries installed. For organizations without their own servers, this may cause other problems when they looking for a web hosting service in the future.

3.4.3 ExpressionEngine / pMachine

ExpressionEngine is a Content Management System (CMS). It is available in a free "Core Version", and in both "Personal" and "Commercial" versions after paying a one-time fee.[19] As a CMS, ExpressionEngine is an excellent and powerful entire website solution including blog, forum, and wiki, however, forum and wiki is only provided to personal and commercial users with purchasing.

ExpressionEngine is great, but this does not mean it is the best choice for building GAS Blog. As discussed in earlier section, GAS Blog is intended to be a blog system, not a complete website, and powerful platform ExpressionEngine may be too heavy and too complicate for GAS Blog, which require more development efforts during the implementation.

3.4.4 WordPress

WordPress, written in PHP and supported by MySQL database, is a state-of-the-art semantic personal publishing platform with a focus on web standards, and usability[20]. As a weblog platform, it also has all the features as other regular blog systems, blogging, entries management, weblog theme template, archiving, commenting, plug-in support, etc.

From the view of users, it is very similar to Movable Type. Actually they are comparably similar in most fields except Movable is partially written in PHP and WordPress is built on pure PHP. Different from ExpressionEngine, WordPress is a blog-only lightweight system, which would save some development efforts comparing to ExpressionEngine.

So far, WordPress meets all the requirements for selecting a blog platform for GAS Blog: open source, written in PHP, and a lightweight system. It is easy to install and configure. Other than that, WordPress is popular with detailed development documentations and free supports from many other developers all over the world. These two advantages are especially important for modifying open source during the implementation of GAS Blog. Therefore, it is the best choice to be the weblog platform for building GAS Blog.

3.4.5 Comparison of Popular Weblog Platforms

According to the description of each weblog platforms, no legal access to source code makes Blogger and MSN Space not good choices for GAS Blog. As for Moveable Type, it is a similar platform to WordPress in most fields, however, being coded in Perl makes it not a good choice for system owners who do not have their own servers due to its requirement for installing additional libraries and standard PHP does not support Perl. ExpressionEngine is an excellent platform as itself; however, it is not lightweight enough

---

[19] http://en.wikipedia.org/wiki/ExpressionEngine
[20] http://wordpress.org/

since it is a content management system rather than a blog-only platform. To build GAS Blog on ExpressionEngine, efforts on implementation would cost more than on a blog purpose only system such as WordPress. Comparison of these weblog platforms is summarized into the following table.

	Web Scripting Language	License	Comments
Blogger	N.A.	Free	Free to use No access to source code
MSN Space	N.A.	Free	Free to use No access to source code
Movable Type	Mostly Perl + Some PHP	Open Source	
ExpressionEngine / pMachine	PHP	Open Source & Partially Free	Free with core features More features for purchasing
WordPress	PHP	Open Source	

Table 3.2 Weblog Platforms Comparison Table

3.5 Selecting Learning Object Metadata Standard

GAS Blog is mainly designed to be a web application for Gene Ontology annotation. In GAS Blog system, each blog post/entry is considered as a learning object for web collaboration. Learning Objects Model is essential for treating blog entries as learning objects. Currently, the popular Learning Object Metadata standards available are SCORM and CanCore.

SCORM references a number of specifications and guidelines to create a multi-dimensional reference model. Significantly, this reference model includes a "content aggregation model", and has been developed in the context of military and training applications. The CanCore metadata profile addresses only one of the many specifications referenced by SCORM – namely, the IMS metadata specification. CanCore, moreover, has been developed in the context of public and continuing education needs and requirements. (Friesen, 2004)

Currently, CanCore metadata is being used to describe and classify content that would be identified in the SCORM content aggregation model as "Raw Media." In compliance with SCORM, the CanCore element set includes all 11 elements that SCORM identifies as mandatory for Raw Media materials. (Friesen, 2004)

In summary, SCORM is more flexiable and has a wider range of usage than CanCore, which is helpful for describing various type of resources as Learning Object and future upgrade. Moreover, SCORM is widely used all over the world, and CanCore is mostly used in Canada. Due to these consideration, SCORM will be used to wrape/describe gene ontologies as Learning Objects in GAS Blog. SCORM will be introduced in later section of this chapter.

3.6 Adding Semantics into GAS Blog

There are some levels of semantics added on top of blog platform – WordPress which was selected from earlier section in this chapter, to make GAS Blog intelligent and semantic: Gene Ontology semantics, SCORM semantics, FOAF semantics and syndication and aggregation semantics.

Before adding these semantics into GAS Blog system to make it semantical, there are two more questions need to be answered:

1. Where to publish semantics.
2. What format they should be published in.

3.6.1 Where to Publish Semantics

Before adding semantics into GAS Blog, the first question is not how to add but where to add. Let's start from the reason why semantics should be added to GAS Blog.

Semantics should be added to GAS Blog for increasing the machine readability of information transferred over network. For this reason, information carrier should be in a format which is convenient to exchange through Internet, be easy to extend for specific data about genes and annotations, and be a widely accepted standard on Internet, so there is no additional software to install on clients/computers for the data delivery. Requirements of semantics carrier can be summarized as follows.

- A convenient format to exchange data on Internet.
- A format which is easy to extend.
- A format with widely accepted standard on Internet.

Currently, syndication feeds are the major method to deliver the latest blog posts or comments to subscribers. XML-based feeds such as RSS and Atom meet all the semantics carrier requirements. They are originally designed for exchanging data on Internet, free to extend to deliver any information with related schemas, and they are primary standards in this field.

3.6.2 Selecting a Format to Publish Semantics

After deciding where to publish semantics in GAS Blog, a format should also be selected. Syndication feeds can be divided into three popular formats with their own schema: RSS 1.0, RSS 2.0, and Atom. Although these three formats all meet the requirements of semantics carrier described in former section, there are still some differences among them.

RSS 1.0, standing for Rich Site Summary, is a RDF format[21]. It is a lightweight multipurpose extensible metadata description and syndication format (Dornfest, 2000).

RSS 2.0, standing for Really Simple Syndication, is based on XML format. It is a simplified syndication feed format comparing to RSS 1.0[22]. Building upon previous versions of RSS, RSS 2.0 is backward compatible with previous versions (Lewin, 2003). The greatest change from RSS 1.0 to RSS 2.0 is the ability to extend the format using namespaces. RSS 2.0 supports namespaces, a standardized approach to add elements. Feeds can contain new elements if they are defined in a namespace (Lewin, 2003).

Atom is an XML language used for web feeds, while the Atom Publishing Protocol (short AtomPub or APP) is a simple HTTP-based protocol for creating and updating web resources.[23]

3.6.2.1 Comparison of Syndication Feeds' Format

Among these three feed formats, RSS 1.0 and Atom is RDF-XML style, and RSS 2.0 is pure XML, which is much simpler on syntax than the other two. However, RSS 2.0 is not RDF-XML style, RDF metadata still can be embedded into it and deliver through network, which makes RSS 2.0 more flexible than RSS 1.0 and Atom.

Another reason for choosing RSS 2.0 is that there is an existing learning objects metadata in RSS 2.0 formats but not a schema in the other two formats available on market yet. GAS Blog will wrap genes and their annotation information into Learning Objects, embed Learning Object Metadata into syndication feed, and deliver them through Internet. Therefore, Learning Object Metadata is crucial in consideration of which formats should be used in GAS Blog.

Because of the reasons discussed above, GAS Blog embeds semantical information into RSS 2.0, and deliver them during process of syndication. After deciding where to publish semantics and which format to publish semantics, the next question is how to add them to GAS Blog.

3.6.3 Adding Gene Ontology Semantics to GAS Blog

The primary purpose of GAS Blog is for gene annotation. Gene Ontology semantics is the first semantical features being added to the system.

Gene Ontology (GO) is a controlled vocabulary which is used to describe the biology of a gene product in any organism[24]. It is built up by blocks of information, and the blocks are called terms. Each term is an entry to Gene Ontology. In GAS Blog, Gene Ontology is displayed on term basis, and each term is treated as a blog entry.

---

[21] http://en.wikipedia.org/wiki/RSS_(file_format)
[22] http://en.wikipedia.org/wiki/RSS_(file_format)
[23] http://en.wikipedia.org/wiki/Atom_(standard)
[24] http://www.yeastgenome.org/help/gotutorial.html

3.6.3.1 Gene Ontology Term

Gene Ontology term is consisted by the following information: id: a unique numerical identifier of the form. For example: "MA: 0000001" is the id for a GO in category of Adult Mouse Anatomy; name: GO term name; synonym: other names of the GO term; and relationship. GO terms are linked together by relationship. There are five different kind of relationship among GO terms:

- is_a: The is_a relationship is a simple class-subclass relationship, where A is_a B means that A is a subclass of B; for example, nuclear chromosome is_a chromosome.[25]
- part_of: The part_of relationship is slightly more complex; C part_of D means that whenever C is present, it is always a part of D, but C does not always have to be present. An example would be periplasmic flagellum part_of periplasmic space.[26]
- regulates, positively_regulates and negatively_regulates: The regulates, positively_regulates and negatively_regulates relationships describe interactions between biological processes and other biological processes, molecular functions or biological qualities. When a biological process E regulates a function or a process F, it modulates the occurrence of F. If F is a biological quality, then E modulates the value of F. An example of the regulation of a biological process would be the term regulation of transcription. When regulation of transcription occurs, it always alters the rate, extent or frequency at which a gene is transcribed.[27]

3.6.3.2 Gene Ontology Format

There are two major types of Gene Ontology format:

1. OBO, a plain text format.
2. XML based formats such as GO RDF-XML format, OBO_XML format and OWL format.

Between the formats above, plain text format obo is deprecated. It is difficult to query through, troublesome to find out the latest change to the file, and it is not a standard format for storing or displaying data on web pages. Based on these reasons, GAS Blog uses a XML based format Gene Ontology called OBO_XML which is generated by Perl script from obo format.

[25] http://www.geneontology.org/GO.doc.shtml
[26] http://www.geneontology.org/GO.doc.shtml
[27] http://www.geneontology.org/GO.doc.shtml

In OBO_XML file, GO terms are defined as a XML node <term> with following children: id – a unique identity for the term; name – name of term; def – term definition; relationship – term relationship with other terms; namespace – a namespace refers to the file in which the term should be stored. A sample term is described as follow:

```
<term>
  <id>CARO:0000014</id>
  <name>cell component</name>
  <def>
    <defstr>Anatomical structure that is a direct part of the
cell.</defstr>
    <dbxref>
      <acc>MAH</acc>
      <dbname>CARO</dbname>
    </dbxref>
  </def>
  <is_a>CARO:0000003</is_a>
  <relationship>
    <type>part_of</type>
    <to>CARO:0000013</to>
  </relationship>
  <namespace>caro</namespace>
</term>
```

3.6.4 Adding Gene Annotation Semantics to GAS Blog

Annotation is the process of assigning GO terms to gene products (GeneOntology.org, 2007). Each annotation of Gene Ontology must include an evidence code to indicate how the annotation to a particular term is supported. The available evidence codes are: (Guide to GO Evidence Codes)

1. Experimental Evidence Codes:

   EXP: Inferred from Experiment

   IDA: Inferred from Direct Assay

   IPI: Inferred from Physical Interaction

   IMP: Inferred from Mutant Phenotype

   IGI: Inferred from Genetic Interaction

   IEP: Inferred from Expression Pattern

2. Computational Analysis Evidence Codes

   ISS: Inferred from Sequence or Structural Similarity

   ISO: Inferred from Sequence Orthology

ISA: Inferred from Sequence Alignment

ISM: Inferred from Sequence Model

IGC: Inferred from Genomic Context

RCA: inferred from Reviewed Computational Analysis

3. Author Statement Evidence Codes

   TAS: Traceable Author Statement

   NAS: Non-traceable Author Statement

4. Curator Statement Evidence Codes

   IC: Inferred by Curator

   ND: No biological Data available

5. Automatically-assigned Evidence Codes

   IEA: Inferred from Electronic Annotation

6. Obsolete Evidence Codes

   NR: Not Recorded

For easy maintenance of original GO files, annotation information is stored in separated XML files with same file base name and different extension. This is also called sequential XML. To connect a GO annotation with GO term assigned to it, GAS Blog system uses XPointer to identify the original GO file location and id of GO term that annotation is assigned to.

An annotation node contains attributes XPointer type and href to point to a GO term, and the following children: evidence_code: evidence code of annotation; content: annotation content; foaf: FOAF network information of annotator; timestamp: time of annotation. A sample GO annotation is given below.

```
<annotation type="simple" href="http://localhost/wordpress/wp-
content/owl/caro.obo_xml#CARO:0000000">
  <evidence_code>TAS</evidence_code>
  <content>a sample annotation</content>
  <foaf>
    <author id="1">admin</author>
    <groups>
      <group id="6">Group 1</group>
    </groups>
  </foaf>
  <timestamp time="1210639289">Mon May 12 20:41:29 EDT
2008</timestamp>
```

```
</annotation>
```

3.6.5 Adding SCORM Semantics to GAS Blog

SCORM, or Shareable Content Object Reference Model, is a compilation of technical specifications for web-based e-learning[28]. One of the primary purposes of the SCORM standards is to define interoperability between learning content and learning management systems. Through SCORM conformance, content packages and learning managements systems together achieve such interoperability.[29]

GAS Blog uses SCORM standard to syndicate Gene Ontology terms which are wrapped as Learning Objects. In SCORM, Learning Objects are described by manifest file. Manifest file is a XML based file that describes the learning objects' package and its contents. It includes the information about identifier, version, schema, resources and organization. Manifest node contains the following children to carry Learning Objects information: metadata – manifest schema and its version; organizations – activities of the Learning Object; Resources – Learning Object resources.

SCORM manifest standard is used in Gene Ontology Navigation for displaying search results. A sample metadata of SCORM manifest is listed as below.

```
<?xml version="1.0" encoding="UTF-8" ?>
<manifest identifier="MANIFEST_IDENTIFIER" version="1.0"
      xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
      xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_v1p3"
      xmlns:adlnav="http://www.adlnet.org/xsd/adlnav_v1p3"
      xmlns:adlseq="http://www.adlnet.org/xsd/adlseq_v1p3"
      xmlns:imsss="http://www.imsglobal.org/xsd/imsss"
      xmlns:lom="http://ltsc.ieee.org/xsd/LOM"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1
      imscp_v1p1.xsd http://www.adlnet.org/xsd/adlcp_v1p3
      adlcp_v1p3.xsd http://www.adlnet.org/xsd/adlnav_v1p3
      adlnav_v1p3.xsd http://www.adlnet.org/xsd/adlseq_v1p3
      adlseq_v1p3.xsd http://www.imsglobal.org/xsd/imsss
      imsss_v1p0.xsd http://ltsc.ieee.org/xsd/LOM lom.xsd"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <metadata>
    <schema>ADL SCORM</schema>
    <schemaversion>CAM 1.3</schemaversion>
  </metadata>
  <organizations default="ORG-CARO">
    <organization identifier="ORG-CARO" structure="hierarchical">
      <title>Activity Tree</title>
      <item identifier="ACT-CARO:0000000" identifierref="RES-
CARO:0000000">
```

[28] http://www.scormsoft.com/scorm/overview
[29] http://www.scormsoft.com/scorm/conformance

```
        <title>Anatomical Entity</title>
      </item>
    </organization>
  </organizations>
  <resources>
    <resource identifier="RES-CARO:0000000" type="webcontent"
adlcp:scormType="sco" href="http://localhost/wordpress/wp-
includes/single.php?category=/caro&ID=CARO:0000000">
      <metadata>
        <lom:lom>
          <lom:general>
            <lom:title>
              <lom:string language="en">Anatomical
Entity</lom:string>
            </lom:title>
            <lom:language>en</lom:language>
            <lom:description>
              <lom:string language="en">Biological entity that is
either an individual member of a biological species or constitutes
the structural organization of an individual member of a biological
species.</lom:string>
            </lom:description>
            <lom:keyword>
              <lom:string>TAS</lom:string>
            </lom:keyword>
            <lom:structure>
              <lom:source>LOMv1.0</lom:source>
              <lom:value>hierarchical</lom:value>
            </lom:structure>
            <lom:annotation>sample annotation</lom:annotation>
          </lom:general>
        </lom:lom>
      </metadata>
      <file href="http://localhost/wordpress/wp-
content/upload/1211166773.gif" />
    </resource>
  </resources>
</manifest>
```

In GAS Blog, each entry is a Learning Object. However, syndication feed standard does not carry any information specifically for Learning Objects since standard RSS only contains metadata of channels and does not carry Learning Object Metadata (LOM) which is necessary for other Learning Management Systems (LMS) to read Learning Objects.

To make RSS carry LOM, it should be extended with LOM schema. A RSS format for Learning Object Metadata (RSS-LOM) is defined by Stephen Downes (Downes, 2003). This format enables RSS to exchange LOM on network. This format (RSS-LOM) makes it possible to distribute learning objects to courses without having to depend on the content libraries provided by a learning management system; it also will allow authors to distribute learning objects without having to work through an intermediary such as a publisher (Harrsch, 2003).

RSS-LOM includes metadata in following categories:

- General Metadata: LO identifier, language, title, description, keyword, coverage, structure, and aggregation level (Harrsch, 2003);
- Lifecycle Metadata: RSS version, status, and contribute; Metadata such as Metadata identifier, schema, contribute, and language (Harrsch, 2003);
- Technical Metadata: RSS format, size, location, operation system and browser requirement, installation remarks, other platform requirement, and duration (Harrsch, 2003);
- LO educational Metadata: interactivity type, learning resource type, interactivity level, semantic density, intended end user role, context, typical age range, difficulty, typical learning time, description, language (Harrsch, 2003);
- LO Rights: cost, copyright and other restrictions, and description (Harrsch, 2003);
- Relation Metadata: relation (Harrsch, 2003);
- Annotation Metadata: annotation (Harrsch, 2003);
- Classification Metadata: LO classification, prerequisite, educational objective, accessibility restrictions, educational level, skill level, security level, and competency (Harrsch, 2003).

```
<dc:identifier>MA:0000001</dc:identifier><!-- post id-->
<dc:title>Mouse Anatomy</dc:title><!-- post title-->
<dc:language>en</dc:language>
<dc:description>mouse anatomy is usually found on
mouse</dc:description>
<dc:subject>/anatomy/adult_mouse_anatomy</dc:subject>
<dc:evidence_code></dc:evidence_code>
<lom-gen:structure>Atomic</lom-gen:structure>
<lom-gen:aggregationLevel>13</lom-gen:aggregationLevel>
<lom-life:version>beta</lom-life:version>
<lom-life:status>final</lom-life:status>
<dc:publisher>Lakehead University</dc:publisher>
<dc:editor>admin</dc:editor>
<lom-meta:metadataScheme resource="lom-meta;LOMv1.0"/>
<dc:format>text/html</dc:format>
<lom-tech:operatingSystem>Multi-OS</lom-tech:operatingSystem>
<lom-edu:interactivityType>Active</lom-edu:interactivityType>
<lom-edu:type>Exercise</lom-edu:type>
<lom-edu:interactivityLevel>Low</lom-edu:interactivityLevel>
<lom-edu:semanticDensity>High</lom-edu:semanticDensity>
<lom-edu:intendedEndUserRole>Manager</lom-edu:intendedEndUserRole>
<lom-edu:context>School</lom-edu:context>
```

```
<lom-edu:difficulty>Easy</lom-edu:difficulty>
<lom-edu:typicalLearningTime>PT1H20M</lom-edu:typicalLearningTime>
<lom-edu:language>en</lom-edu:language>
<lom-rights:copyrightAndOtherRestrictions>Lakehead University</lom-
rights:copyrightAndOtherRestrictions>
<dc:rights>RightsBroker:RightsModel</dc:rights>
```

3.6.6 Adding FOAF Semantics to GAS Blog

Friend of a Friend (FOAF) is a machine-readable ontology describing persons, their activities and their relations to other people and objects. Anyone can use FOAF to describe him or herself. FOAF allows groups of people to describe social networks without the need for a centralised database.[30]

In GAS Blog, FOAF is embedded into annotation XML fragment as a child to carry annotators' information. Each FOAF node contains the information of annotator/author id, annotator/author name, and group(s) the annotator/author joined. A sample FOAF fragment used to describe annotator of a certain GO term is given as below.

```
<foaf>
  <author id="1">admin</author>
  <groups>
    <group id="6">Group 1</group>
  </groups>
</foaf>
```

With similar idea on integrating Learning Object metadata to RSS, RSS-FOAF is introduced by Johannes Ernst to allow RSS carry FOAF metadata (Ernst, 2005). The following tags are extended by RSS-FOAF:

- rss-foaf:type – this tag indicates that an RSS item should be treated as representing a Person or other entity with whom the individual exporting the extended RSS feed has a relationship with (Ernst, n.d.);
- rss-foaf:group – this tag indicates that an RSS item should be treated as representing a social group, as seen from the perspective of the individual exporting the extended RSS feed (Ernst, n.d.);
- rss-foaf:rel – this tag may be contained by RSS items that represent a Person or other entity with whom the individual has a relationship with (Ernst, n.d.).

```
<rss-foaf:type name="admin"/>
<rss-foaf:group name="/anatomy/adult_mouse_anatomy"/>
<rss-foaf:rel to="user1"/>
```

[30] http://en.wikipedia.org/wiki/FOAF_(software)

3.6.7 Syndication & Aggregation Semantics of GAS Blog

As mentioned in Chapter 1, RSS is widely used to syndicate blog information through Internet. Users can subscribe to RSS feed and receive notice of news. In GAS Blog, syndication can be considered as the output of the system. It integrates metadata of Gene Ontologies, SCORM Learning Object, and FOAF together, and delivers them to end clients to aggregate in format of RSS 2.0.

Most regular blog systems use broadcast as the way of syndication. They send everything to everybody who subscribed, without considering whether the subscribers like it or not. Instead of broadcasting, GAS Blog uses a more semantical way to do the job – narrowcasting. GAS Blog allows subscribers to customize their interested categories of gene. After saving their interests to the system, GAS Blog would narrowcast information about new annotation or newly found Gene Ontologies to subscribers who show their interests to the gene categories with the integrated metadata mentioned in earlier section of this chapter.

Narrowcasting is a better way for syndication comparing to broadcasting for the following reasons. First of all, it decreases the unnecessary information transferring over network and saves subscribers' time and effort to classify and delete them. On the other hand, it is more intelligent and semantical than broadcasting.

A sample RSS generated by GAS Blog is given below.

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0"
      xmlns:content="http://purl.org/rss/1.0/modules/content/"
      xmlns:wfw="http://wellformedweb.org/CommentAPI/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:rss-foaf="http://rss-extensions.org/wiki/Rss-foaf"
      xmlns:rss-pm="http://rss-extensions.org/wiki/Rss-foaf"
      xmlns:lom-gen="http://kmr.nada.kth.se/el/ims/schemas/lom-
      general"
      xmlns:lom-life="http://kmr.nada.kth.se/el/ims/schemas/lom-
      lifecycle"
      xmlns:lom-meta="http://kmr.nada.kth.se/el/ims/schemas/lom-
      metadata"
      xmlns:lom-tech="http://kmr.nada.kth.se/el/ims/schemas/lom-
      technical"
      xmlns:lom-edu="http://kmr.nada.kth.se/el/ims/schemas/lom-
      educational"
      xmlns:lom-rights="http://kmr.nada.kth.se/el/ims/schemas/lom-
      rights">
  <channel>
    <title>LU Gene Annotation Semantic Blog</title>
    <link>http://localhost/wordpress</link>
    <description>Wei Yuan's Thesis</description>
    <pubDate>Tue, 18 Mar 2008 03:11:45 +0000</pubDate>
    <generator>http://wordpress.org/?v=2.3.3</generator>
    <language>en</language>
    <item>
```

```
      <title>Mouse Anatomy by admin at Wed Jun 11 15:43:41 EDT
2008</title>
      <link>http://localhost/wordpress/wp-
includes/single.php?category=/anatomy/adult_mouse_anatomy&#38;ID=MA:0
000001&#35;annotation-1213213421</link>
      <annotations>http://localhost/wordpress/wp-
includes/single.php?category=/anatomy/adult_mouse_anatomy&#38;ID=MA:0
000001&#35;annotation-1213213421</annotations>
      <dc:annotator>admin</dc:annotator>
      <!-- FOAF section-->
      <rss-foaf:type name="admin"/>
      <rss-foaf:group name="/anatomy/adult_mouse_anatomy"/>
      <rss-foaf:rel to="user1"/>
      <!-- learning object section -->
      <dc:identifier>MA:0000001</dc:identifier><!-- post id-->
      <dc:title>Mouse Anatomy</dc:title><!-- post title-->
      <dc:language>en</dc:language>
      <dc:description>annotation from group 1</dc:description>
      <dc:subject>/anatomy/adult_mouse_anatomy</dc:subject>
      <dc:evidence_code>TAS</dc:evidence_code>
      <lom-gen:structure>Atomic</lom-gen:structure>
      <lom-gen:aggregationLevel>13</lom-gen:aggregationLevel>
      <lom-life:version>beta</lom-life:version>
      <lom-life:status>final</lom-life:status>
      <dc:publisher>Lakehead University</dc:publisher>
      <dc:editor>admin</dc:editor>
      <lom-meta:metadataScheme resource="lom-meta;LOMv1.0"/>
      <dc:format>text/html</dc:format>
      <lom-tech:operatingSystem>Multi-OS</lom-tech:operatingSystem>
      <lom-edu:interactivityType>Active</lom-edu:interactivityType>
      <lom-edu:type>Exercise</lom-edu:type>
      <lom-edu:interactivityLevel>Low</lom-edu:interactivityLevel>
      <lom-edu:semanticDensity>High</lom-edu:semanticDensity>
      <lom-edu:intendedEndUserRole>Manager</lom-
edu:intendedEndUserRole>
      <lom-edu:context>School</lom-edu:context>
      <lom-edu:difficulty>Easy</lom-edu:difficulty>
      <lom-edu:typicalLearningTime>PT1H20M</lom-
edu:typicalLearningTime>
      <lom-edu:language>en</lom-edu:language>
      <lom-rights:copyrightAndOtherRestrictions>Lakehead
University</lom-rights:copyrightAndOtherRestrictions>
      <dc:rights>RightsBroker:RightsModel</dc:rights>
    </item>
  </channel>
</rss>
```

3.7 Overall Architecture

As mentioned at the beginning of this chapter, current gene annotation systems do not allow users to collaborate with each other; it is inefficient to publish information on them; it is difficult for users to find the latest annotations to existing genes; and search mechanism is inefficient as well.

Under this situation, GAS Blog is developed to solve the disadvantages of existing gene annotation systems. The purpose of GAS Blog system is to help biologist to share their research or finding.

As mentioned by the end of Chapter 2 and displayed in Figure 3.1, GAS Blog is composed of the following modules.

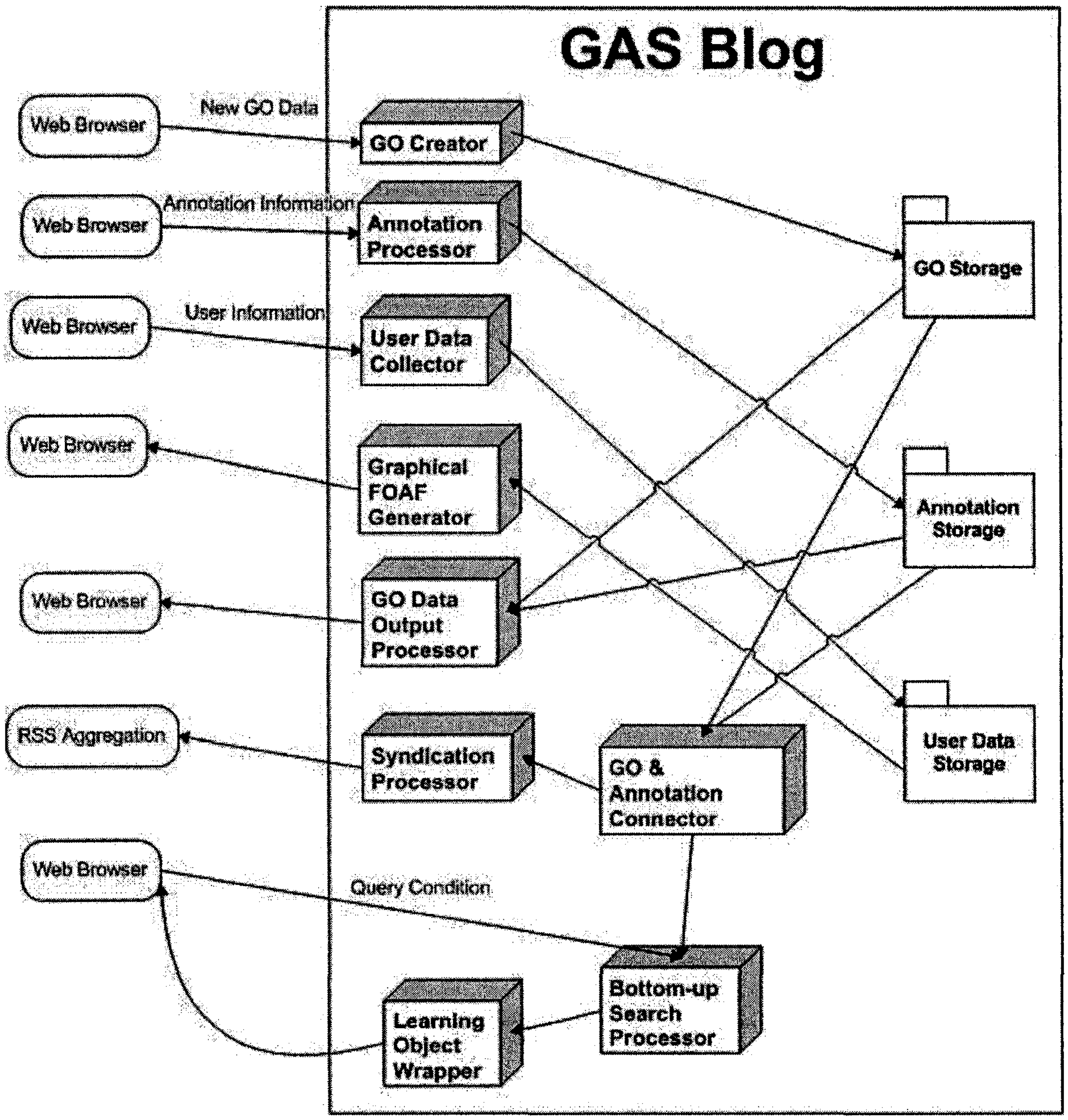


Figure 3.1 GAS Blog Overall Architecture

- GO Creator: GO Creator creates a new gene ontology according to data retrieved from users' web browser;

- Annotation Processor: Annotation Processor annotates to gene ontology according to data retrieved from users' web browser;
- User Data Collector: User Data Collector collects user information from users such as user name, email, interested gene ontology categories, groups willing to join, etc.;
- Graphical FOAF Generator: Graphical FOAF Generator generates graphical FOAF network information;
- GO & Annotation Connector: GO & Annotation Connector prepares data for Bottom-up search Processor by combining the information of Gene Ontology, annotation, and FOAF network;
- Syndication Processor: Syndication Processor gets the well-wrapped data from Learning Object Wrapper and generate syndication for RSS aggregation;
- GO Data Output Processor: GO Data Output Processor gets the data from Gene Ontology and annotation, and display them on web page;
- Learning Object Wrapper: Learning Object Wrapper gets the Gene Ontology and annotation data and wrap them as Learning Objects;
- Bottom-up Search Processor: Bottom-up Search Processor searches through Gene Ontology, annotation, and FOAF network using bottom-up search mechanism basing on data prepared by Data Combination Module.

As shown in Figure 3.1, all the data in GAS Blog are stored in GO storage, annotation storage, and user data storage. Among them, GO storage and annotation is XML-based file format, and user data storage is in MySQL database. They are the core of GAS Blog and the media for different modules to exchange data, communicate and interact with each other.

GO Creator, Annotation Processor, User Data Collector, and FOAF Information Collector are inputting modules of GAS Blog. They collect data inputted by users and store them into related storage for other modules to use.

GO & Annotation Connector is a media module between data storage and output modules; it prepares the data from storage for output purpose. Graphical FOAF Generator, Syndication Processor, GO Data Output Processor, and Learning Object Wrapper are output modules of GAS Blog. They make use of data prepared by GO & Annotation Connector and deliver them to users.

Among these modules, Learning Object Wrapper and Bottom-up Search Processor will be explained stand-alone in next chapter.

3.8 Overall GAS Architecture Class/Module UML

Figure 3.2 shows an overall GAS architecture in class/module UML diagram. It illustrates the major classes with their methods for implementing every module introduced in last section.

Implementation details of each class/module in Figure 3.2 will be explained in later sections of this chapter.

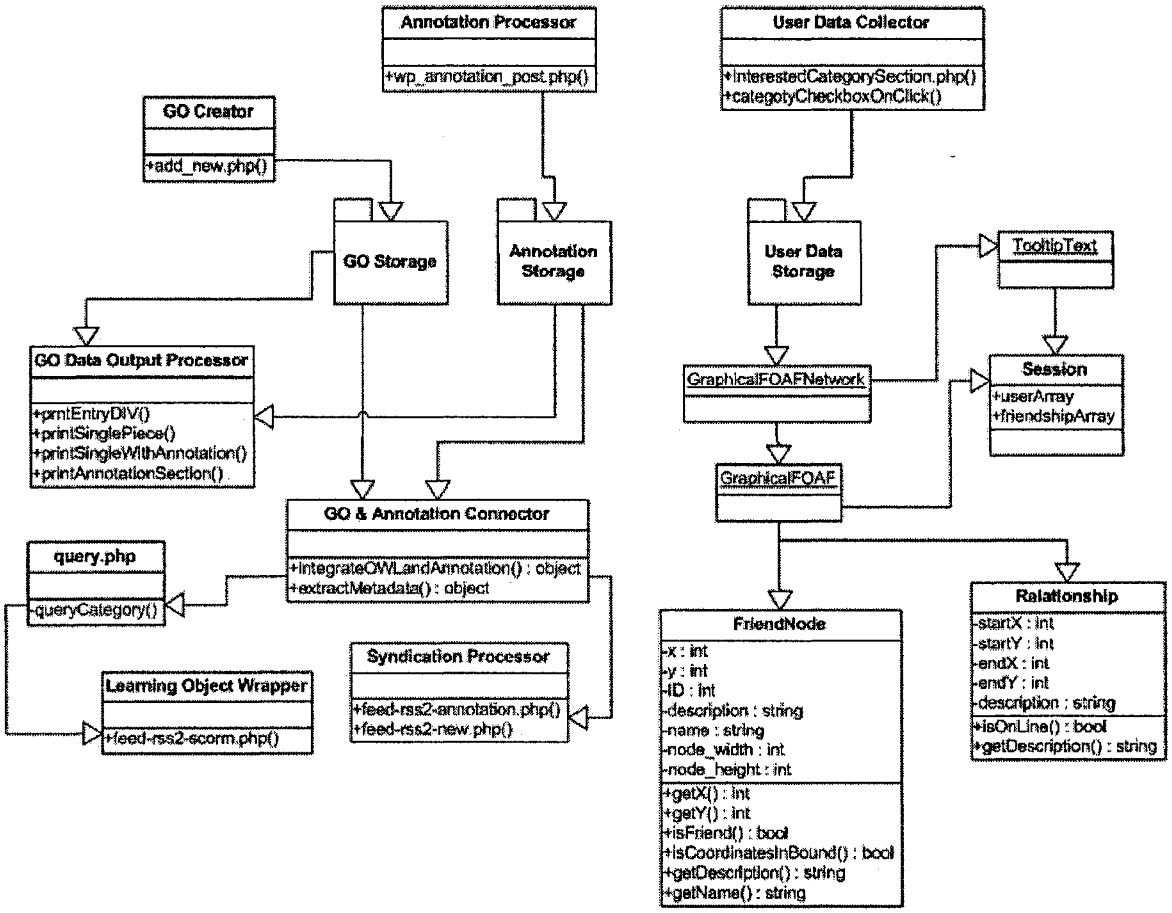


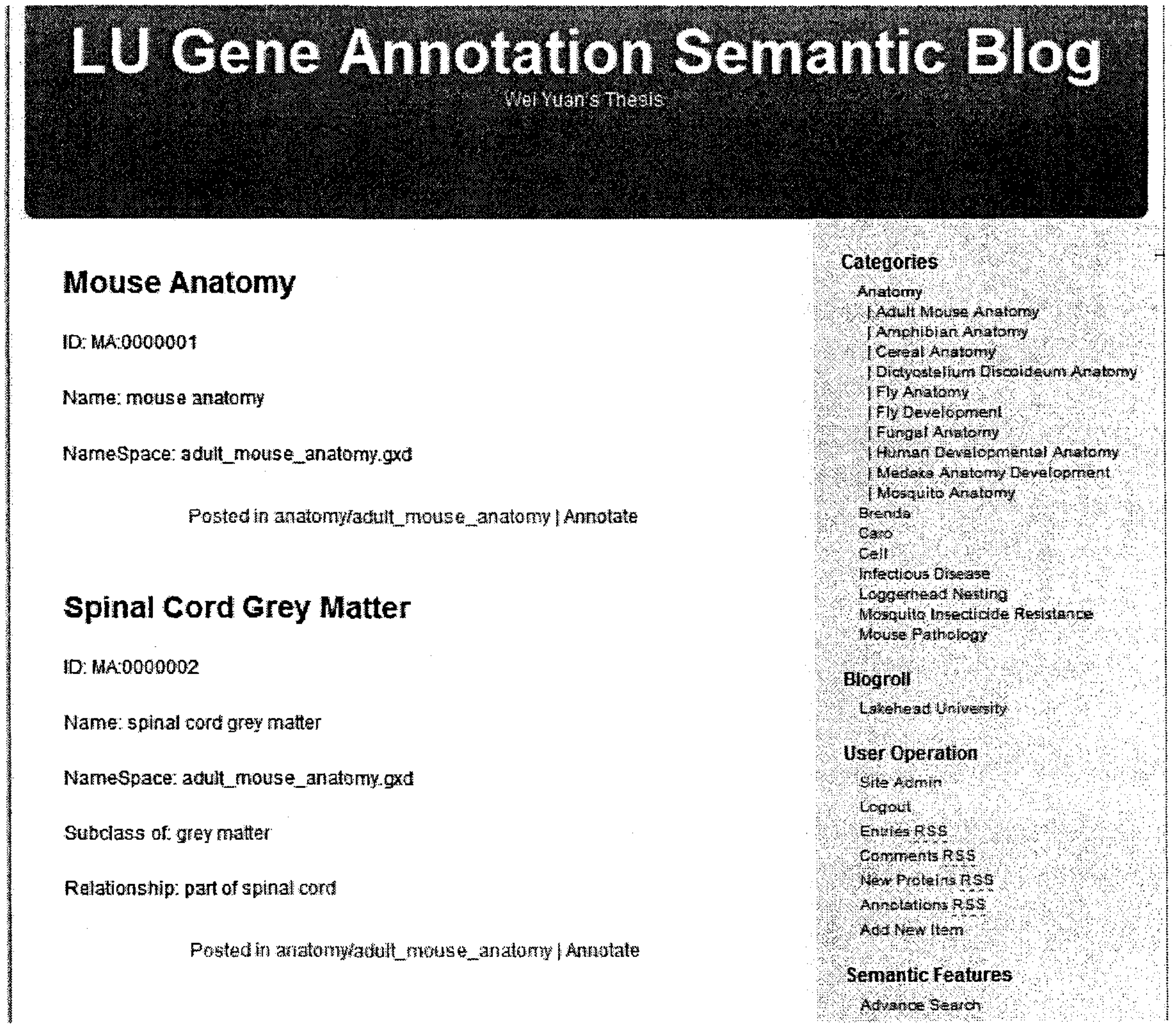
Figure 3.2 Overall GAS Architecture Class/Module UML

3.9 GAS Blog Usage Scenario

For better description of GAS Blog architecture, this chapter starts with introducing GAS Blog usage scenario to deliver a clear image on what GAS Blog can do and how to operate it. The implementation and develop information of will be discussed in following sections of this chapter on module basis.

3.9.1 Displaying Gene Ontology Term

GAS Blog loads Gene Ontology from OBO_XML file into web pages for blog users to view in their browsers. Gene Ontology is categorised on a list, and users can choose a category to view by clicking the link of the category.



LU Gene Annotation Semantic Blog

Wei Yuan's Thesis

Mouse Anatomy

ID: MA:0000001

Name: mouse anatomy

NameSpace: adult_mouse_anatomy.gxd

Posted in anatomy/adult_mouse_anatomy | Annotate

Spinal Cord Grey Matter

ID: MA:0000002

Name: spinal cord grey matter

NameSpace: adult_mouse_anatomy.gxd

Subclass of: grey matter

Relationship: part of spinal cord

Posted in anatomy/adult_mouse_anatomy | Annotate

Categories

Anatomy
| Adult Mouse Anatomy
| Amphibian Anatomy
| Cereal Anatomy
| Dictyostelium Discoideum Anatomy
| Fly Anatomy
| Fly Development
| Fungal Anatomy
| Human Developmental Anatomy
| Medaka Anatomy Development
| Mosquito Anatomy
Brenda
Caro
Cell
Infectious Disease
Loggerhead Nesting
Mosquito Insecticide Resistance
Mouse Pathology

Blogroll

Lakehead University

User Operation

Site Admin
Logout
Entries RSS
Comments RSS
New Proteins RSS
Annotations RSS
Add New Item

Semantic Features

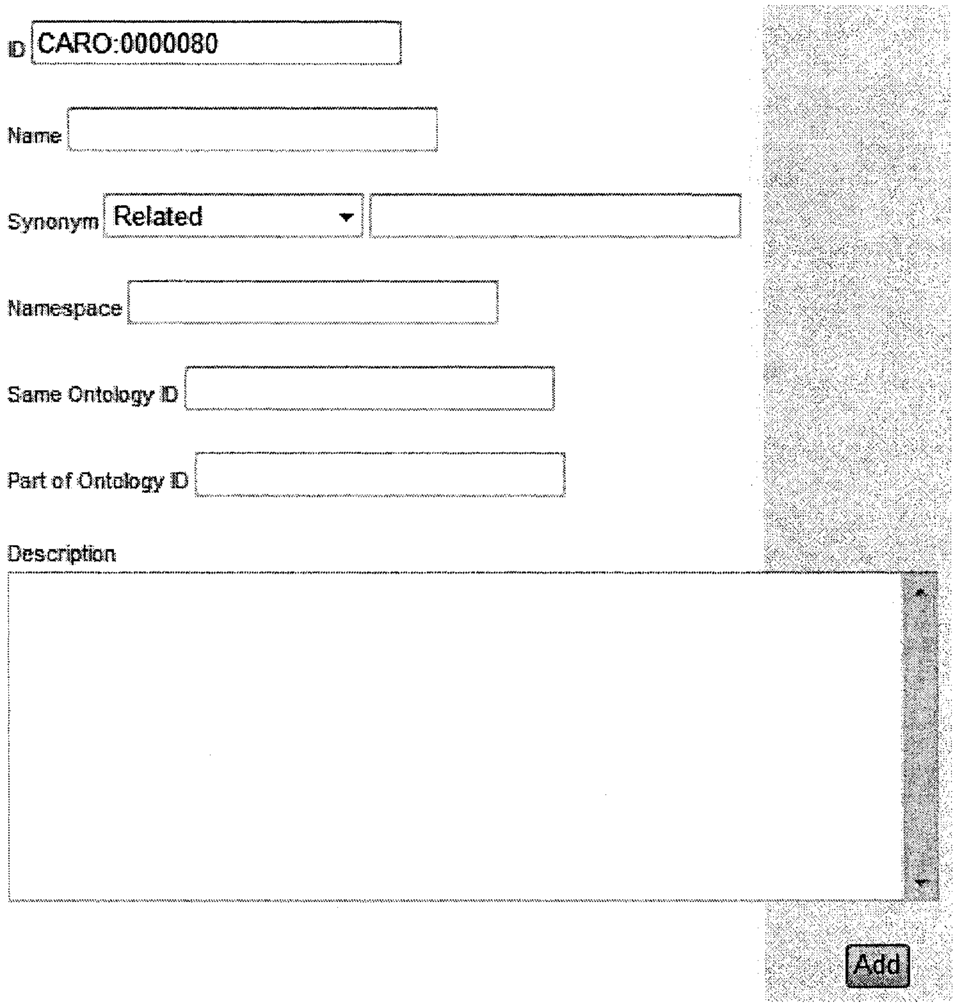
Advance Search

Figure 3.3 Gene Ontology Term Displaying & Browsing Page

3.9.2 Adding New Gene Ontology Term

Other than browsing existing GO terms, GAS Blog also allows registered users to add new found Gene Ontology terms. After filling the form for adding new GO term, blog system construct a new obo_xml node for the term, append it to the original GO file, and also have a backup in a separated obo_xml file. The reason to have a backup is for syndication of new GO terms through RSS. In regular GO term, there is no information about when terms are created, and RSS syndication relies on time to notice subscribed users. Backup of newly added GO terms recorded this information for easier syndication.

On the following form being used to add new GO term, ID is not editable and generated automatically by blog system to make sure that the ID is unique.


ID CARO:0000080

Name

Synonym Related

Namespace

Same Ontology ID

Part of Ontology ID

Description

Figure 3.4 Add New Gene Ontology Form

3.9.3 Annotating Gene Ontology

GAS Blog allows registered users to annotate on Gene Ontology terms. After login the blog system successfully, user has to choose a GO term to annotate, fill an annotation form and submit annotation. Users can also upload an image with annotation.

In the text field of "Evidence Code", GAS Blog pops out the possible evidence code dynamically according to the first few letters users typed in, and users can choose any one of them by clicking the popped-out evidence code, or users can also use an evidence code that is not in the list by keeping typing in the text field and ignoring the tips provided by GAS Blog.

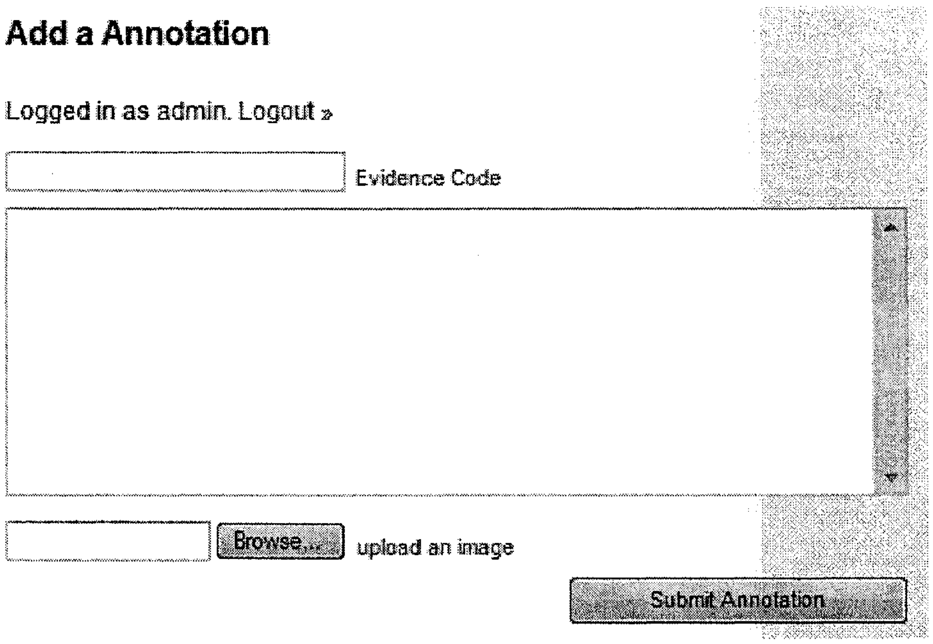


Figure 3.5 Annotation Form

3.9.4 Creating User Group

In GAS Blog, users can create groups so other users can join, share knowledge and collaborate with each other. To create a new group, users need to go to page "Site Admin" and select tab "Users" to edit user, under section "group", input a group name and click the link "Add a new group". If the group name is not used by other groups, a new group with inputted name can be created successfully.

Figure 3.6 Group Operation GUI – Adding New Group

3.9.5 Joining User Group

In GAS Blog, users can also join the existing groups. On the same section of "Creating User Group", all available groups over GAS Blog will be listed as checkboxes, and checked groups indicate that the user is already in that group. Users can join the groups they like by checking the group name, or quit the groups they already joined by un-checking the group name.

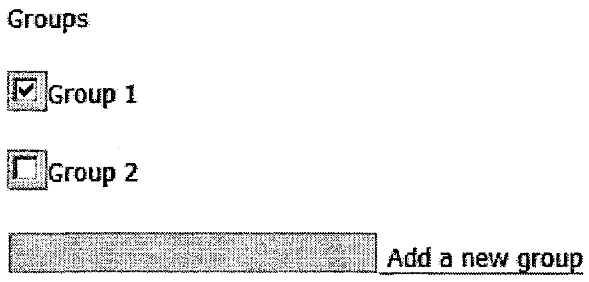


Figure 3.7 Group Operation GUI

3.9.6 Adding Friends

In GAS Blog, users can add other users as their friend in a FOAF network. To do this, users need to go to page "Site Admin" and choose tab "Users" to edit user, and there is a section for users to add friends as Figure 3.8. After inputting the friend's user name on GAS Blog, friendship between them will be created in the FOAF network.

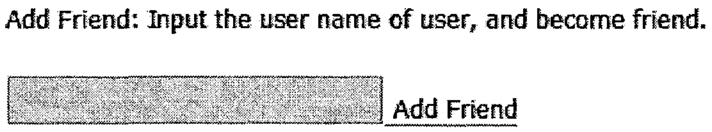


Figure 3.8 Friendship Management GUI

3.9.7 Viewing Graphical FOAF Network

GAS Blog treats registered users as a member of its FOAF network, in this way users can collaborate with each other more convenient than usual weblogs. The blog allows users to create groups or join existing ones.

Blog users can also view FOAF network graphically. Graphical FOAF Network displays all the friendship inside the blog, which makes the blog a FOAF Network. Registered users to the blog can make friends with other users by inputting the user name as Figure 3.8 shows.

A sample Graphical FOAF network is given below in Figure 3.9. Each person icon represents a registered user of the blog with their user name displayed beside the icon. Users in the same group are drawn in the same colour of frame around person icon. A tooltip will popup on the picture with information of name, group, and interested categories when mouse hover on a user icon. The straight line is used to indicate the friendships between two users. A tooltip will also popup to display the information of friendship when mouse is hovered on a line.

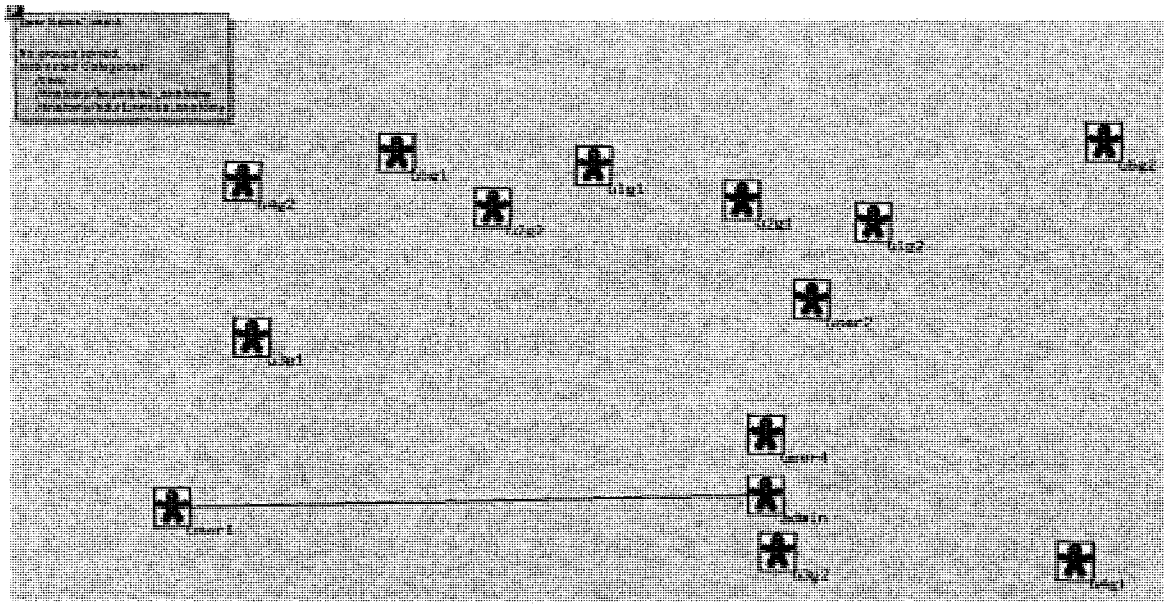
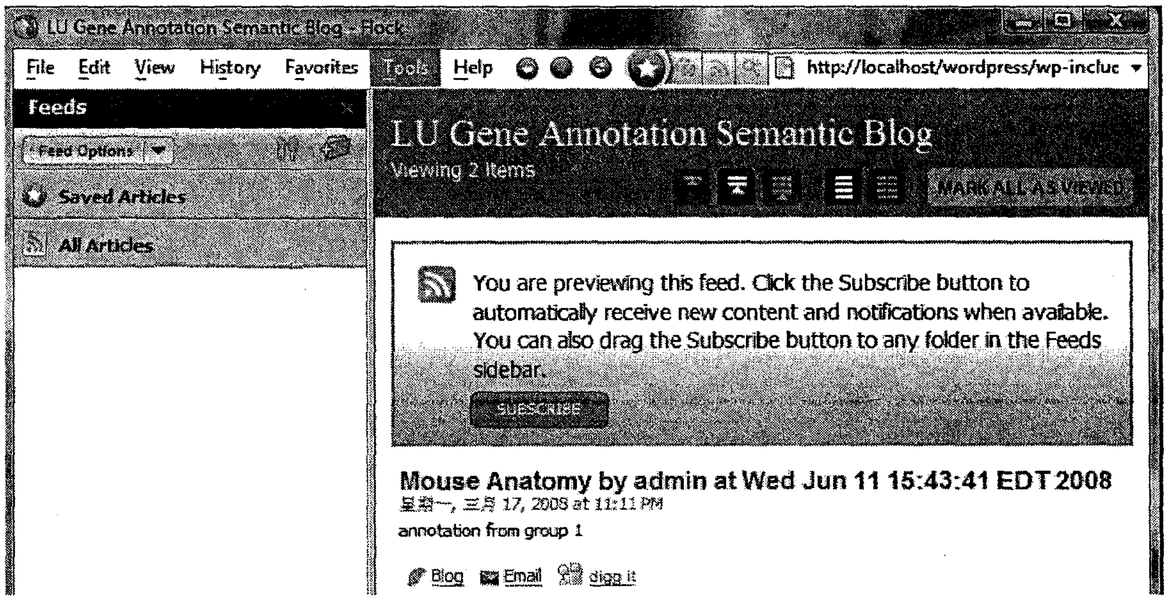


Figure 3.9 Graphical FOAF Network

3.9.8 GAS Blog Syndication & Aggregation

There are two syndication feeds in GAS Blog, “New Proteins RSS” and “Annotation RSS”. The former is for syndicating newly added gene ontology and the latter is for gene ontology annotation. These two syndication feeds are the same to subscribe to. To subscribe to any one of them, users have to login first. On the right side bar of main page of GAS Blog, click the link of “New Proteins RSS” or “Annotation RSS”. Another way to subscribe to a feed is copying the link address and adding it to users’ aggregator.



LU Gene Annotation Semantic Blog
Viewing 2 Items

You are previewing this feed. Click the Subscribe button to automatically receive new content and notifications when available. You can also drag the Subscribe button to any folder in the Feeds sidebar.

Mouse Anatomy by admin at Wed Jun 11 15:43:41 EDT 2008
星期一, 三月 17, 2008 at 11:11 PM
annotation from group 1

Figure 3.10 Syndication Feeds in Aggregator

3.10 GAS Blog Detailed Architecture

After introducing GAS Blog usage scenario, this section discusses development and implementation of each module. In GAS Blog, users can browse gene ontology terms, add new gene ontology terms, annotate to existing gene ontology terms, syndicate new gene ontology terms or annotation of existing ones, view FOAF user network graphically, and navigate through gene ontology terms which will be introduced separately in next chapter.

3.10.1 Browsing Gene Ontology

In GAS Blog, all the Gene Ontology data is stored in XML based "obo_xml" file. GAS Blog system parses data from GO file and displays them as blog entry.

As the diagram below, the GAS system categorizes all Gene Ontology and lists the categories for users to choose. After selecting a category to view, GAS Blog parse GO terms saved in obo_xml file of the selected category and display terms on web as blog entries so that users can browse GO terms in web browsers.

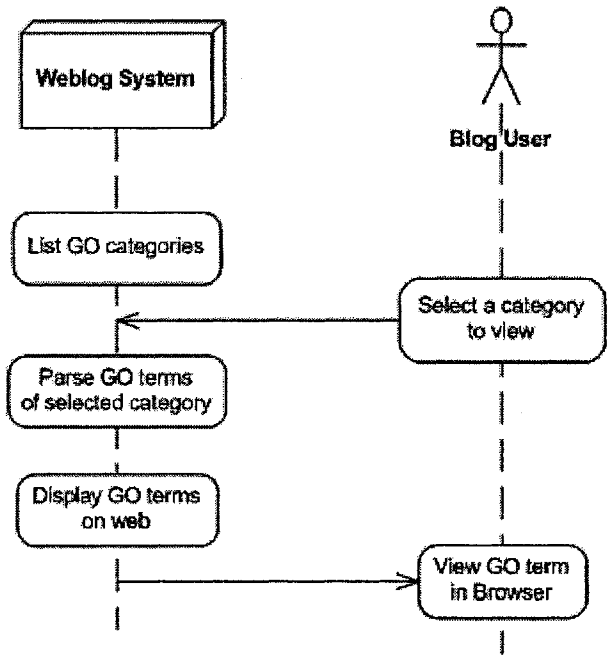


Figure 3.11 Sequential Diagram of Parsing GO Terms

3.10.2 Adding New Gene Ontology Term

Information of newly added GO terms is appended to GO file with other terms and backup in a separated obo_xml file with timestamp for future use of syndicating newly terms.

As Figure 3.12, registered users have to select a category to add the new GO term. Blog generates a unique ID for the new GO term according to the ID naming rule. After filling the form of adding new GO term, GAS Blog constructs the new GO term in obo_xml format, appends it into the GO file of selected category, and creates a backup for the term as described in earlier section in this chapter.

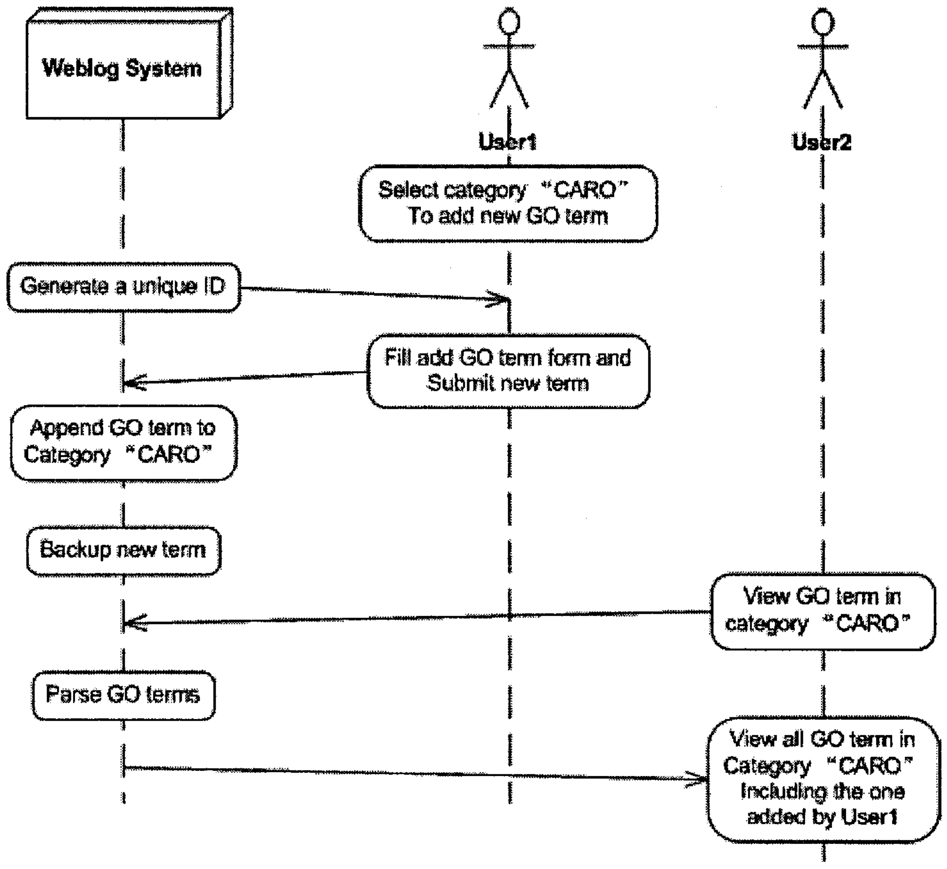

Weblog System
User1
User2
Select category "CARO" To add new GO term
Generate a unique ID
Fill add GO term form and Submit new term
Append GO term to Category "CARO"
Backup new term
View GO term in category "CARO"
Parse GO terms
View all GO term in Category "CARO" Including the one added by User1

Figure 3.12 Sequential Diagram of Adding New GO Term

3.10.3 Annotating Gene Ontology Term

As Figure 3.13, registered users have to select the category and term before annotating. After filling the form for annotating term, GAS Blog system writes annotation information in the format described earlier in this chapter. Annotation can be viewed by other users if they browse the same term.

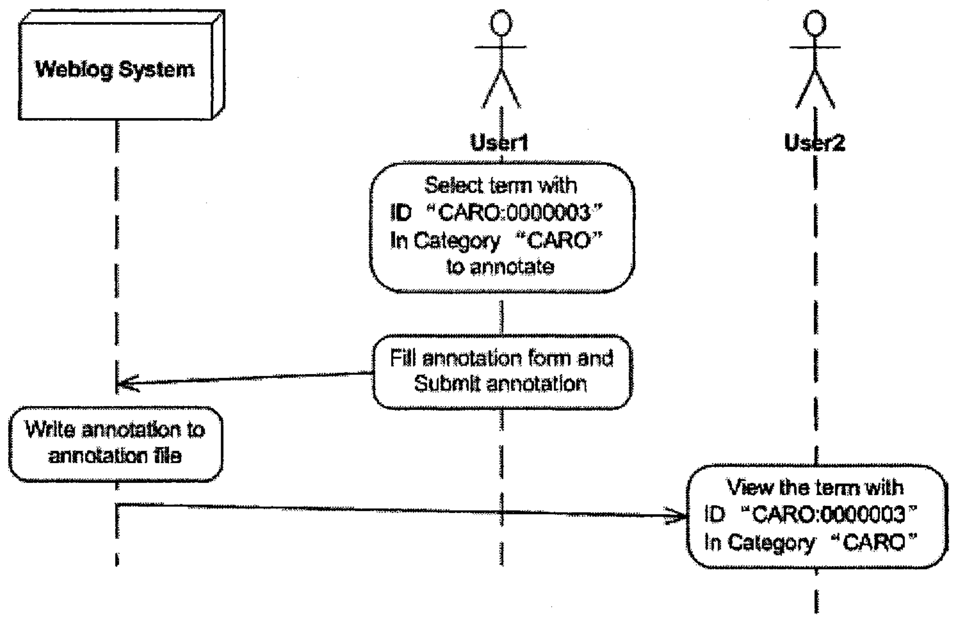

Weblog System
User1
User2
Select term with ID "CARO:0000003" In Category "CARO" to annotate
Fill annotation form and Submit annotation
Write annotation to annotation file
View the term with ID "CARO:0000003" In Category "CARO"

Figure 3.13 Sequential Diagram of Annotation

3.10.4 Syndication of Newly Added Gene Ontology Term

As described in earlier section of this chapter, narrowcasting is used to syndicate newly added GO terms to registered users who shows interest to their categories. Although registered users are from various groups, syndication of newly added GO term is based on users' interest rather than groups.

For example, U1G1, U2G1, and U3G1 are GAS users from "Group 1" and U1G2 is from "Group 2". Among them, U1G1, U2G1, and U1G2 showed interest in the same categories Adult Mouse Anatomy. At one time, U1G1 adds a new GO term to category Adult Mouse Anatomy. After U1G1 adding the new term, U3G1 adds interest to category Adult Mouse Anatomy. However, GAS Blog only notices U1G1, U2G1, and U1G2 about the new term through RSS because U3G1 adds interest after U1G1 adding the new GO term. At another time, U2G1 cancelled the interest in category Adult Mouse Anatomy, and U1G1 adds another new term to category Adult Mouse Anatomy. At this time, U1G1, U3G1, and U1G2 get the notice from GAS Blog, since U2G1 cancelled the interest in this category. Figure 3.14 shows the example in a sequential diagram.

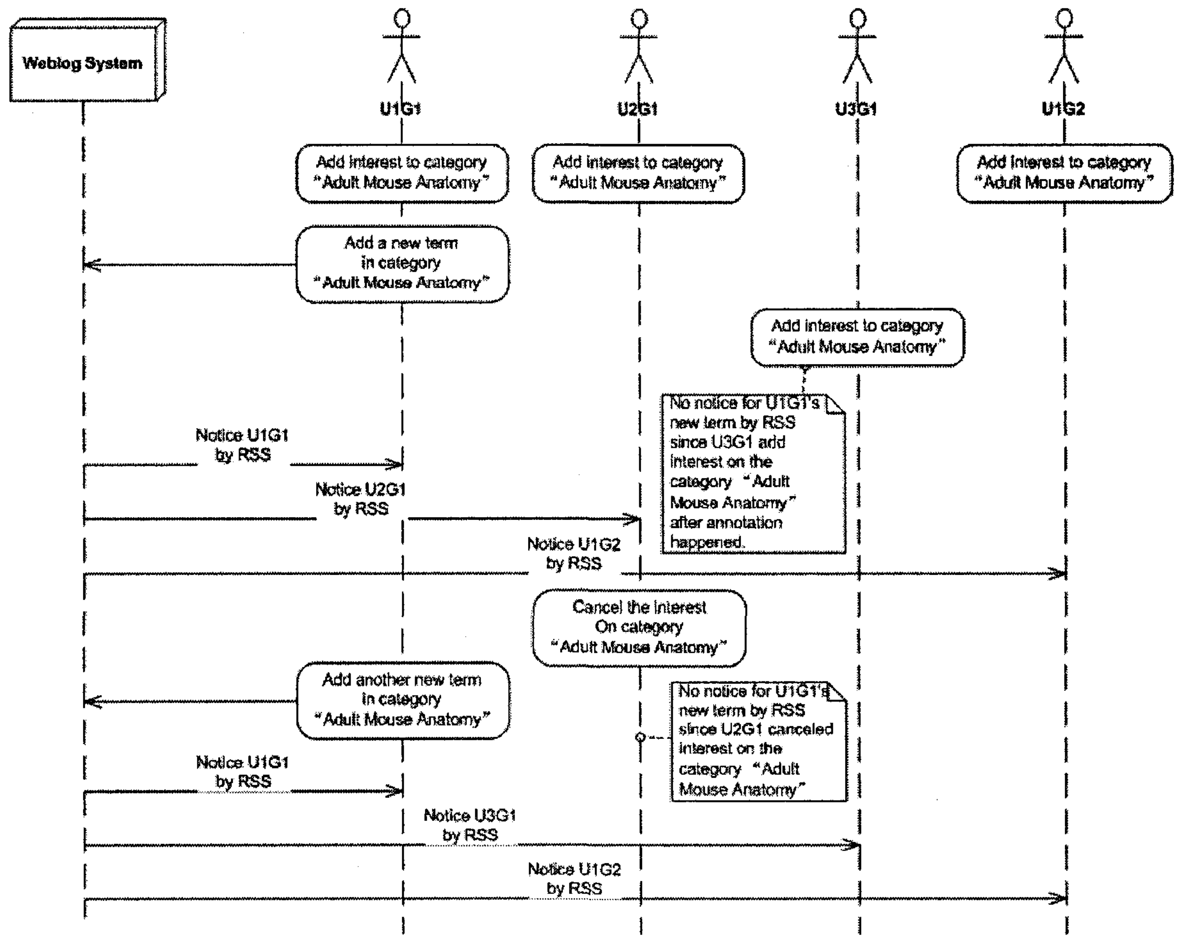

Weblog System
U1G1
U2G1
U3G1
U1G2
Add interest to category "Adult Mouse Anatomy"
Add interest to category "Adult Mouse Anatomy"
Add interest to category "Adult Mouse Anatomy"
Add a new term in category "Adult Mouse Anatomy"
Add interest to category "Adult Mouse Anatomy"
No notice for U1G1's new term by RSS since U3G1 add interest on the category "Adult Mouse Anatomy" after annotation happened.
Notice U1G1 by RSS
Notice U2G1 by RSS
Notice U1G2 by RSS
Cancel the interest On category "Adult Mouse Anatomy"
Add another new term in category "Adult Mouse Anatomy"
No notice for U1G1's new term by RSS since U2G1 canceled interest on the category "Adult Mouse Anatomy"
Notice U1G1 by RSS
Notice U3G1 by RSS
Notice U1G2 by RSS

Figure 3.14 Sequential Diagram of Syndicating Newly Added GO Terms

3.10.5 Syndication of Annotation

The process of syndicating annotation is similar to the syndication of newly added GO term. It narrow-casts to registered users who showed interest in category the annotation belongs to, no matter what groups users joined.

Figure 3.15 indicates the process of annotating to terms. U1G1, U2G1, and U3G1 are GAS users from "Group 1" and U1G2 is from "Group 2". Among them, U1G1, U2G1, and U1G2 showed interest in the same categories Adult Mouse Anatomy. At one time, U1G1 annotates to one of terms in category Adult Mouse Anatomy. After U1G1 annotating to the term, U3G1 adds interest to category Adult Mouse Anatomy. However, GAS system only notices U1G1, U2G1, and U1G2 through RSS because U3G1 adds interest after U1G1 annotating the GO term. At another time, U2G1 cancelled the interest in category Adult Mouse Anatomy, and U1G1 annotates to category Adult Mouse Anatomy again. At this time, U1G1, U3G1, and U1G2 get the notice from GAS system, since U2G1 cancelled the interest in this category.

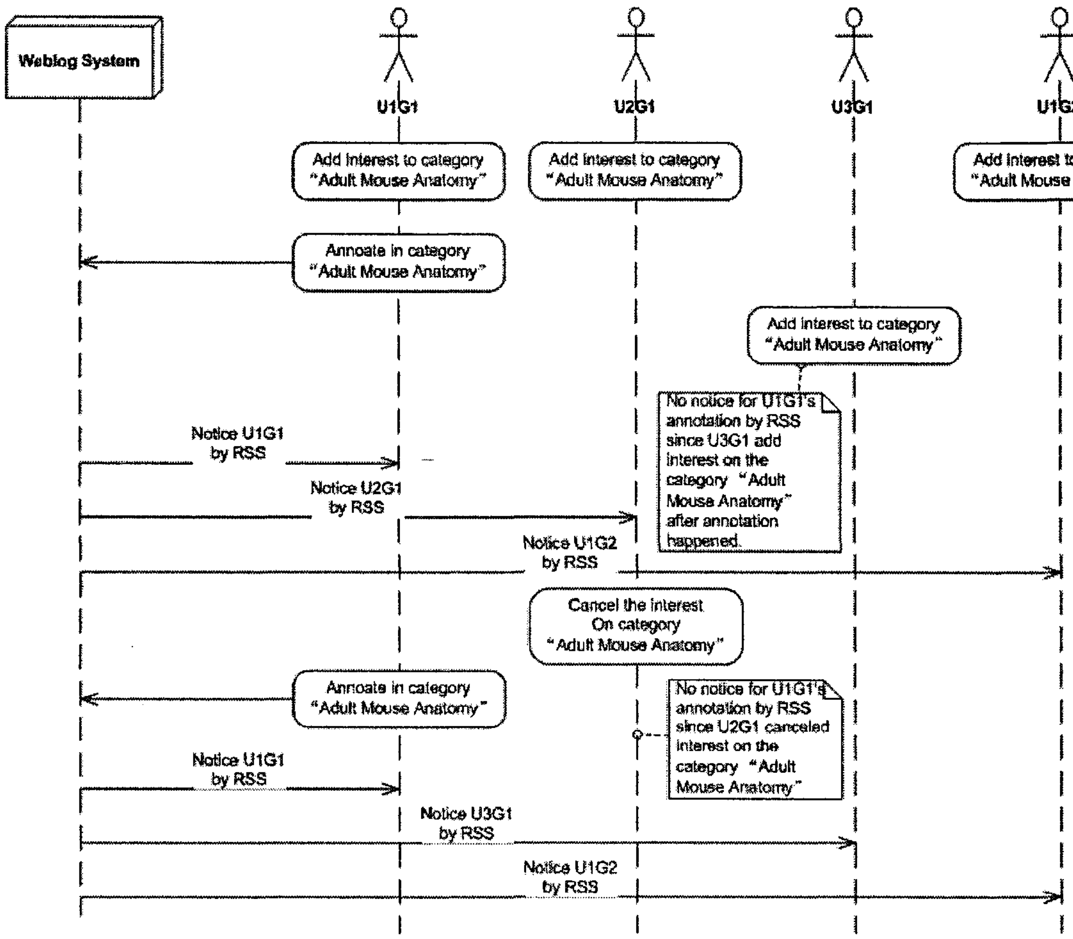

Weblog System
U1G1
U2G1
U3G1
U1G2
Add interest to category "Adult Mouse Anatomy"
Add interest to category "Adult Mouse Anatomy"
Add interest to category "Adult Mouse Anatomy"
Annoate in category "Adult Mouse Anatomy"
Add interest to category "Adult Mouse Anatomy"
No notice for U1G1's annotation by RSS since U3G1 add interest on the category "Adult Mouse Anatomy" after annotation happened.
Notice U1G1 by RSS
Notice U2G1 by RSS
Notice U1G2 by RSS
Cancel the interest On category "Adult Mouse Anatomy"
Annoate in category "Adult Mouse Anatomy"
No notice for U1G1's annotation by RSS since U2G1 canceled interest on the category "Adult Mouse Anatomy"
Notice U1G1 by RSS
Notice U3G1 by RSS
Notice U1G2 by RSS

Figure 3.15 Sequential Diagram of Syndicating Annotations

3.11 GAS Blog Module Implementation

This section talks about the implementation of each module which was mentioned in section of overall architecture, earlier of this chapter.

3.11.1 GO Creator

Gene Ontology Creator is a module that creates newly found Gene Ontology term according to Gene Ontology data inputted by users. It implements the usage scenario of "adding new Gene Ontology term" discussed in earlier section of this chapter.

This module provides users a graphical user interface of a form as Figure 3.4 to collect the data from users' input. Since all Gene Ontology terms is stored in XML-based format files, GO Creator composes the data getting from user interface into XML fragment following the same schema structure described in section 3.5.3 and save to the end of related Gene Ontology file.

Meanwhile, GO Creator also saves the newly added term separately as backup in separate directory with similar schema structure. The only difference between regular ontology schema and the backup schema is that the backup has one more namespace for time that the term is added. In this way, GAS Blog can easily tell existing gene ontology terms and newly added ones apart, and track in time sequence.

The pseudo code is as followed to reflect the logic description described above.

```
<?php
  ... ...
  $sameas=$_POST['sameas'];
  $partof=$_POST['partof'];
  $description=$_POST['description'];
  $file=$_POST['file'];
  $xml=simplexml_load_file(ABSPATH."/wp-content/owl$file");
  $new_path=ABSPATH."/wp-content/new_protein/$file";
  if(file_exists($new_path))$new_xml=simplexml_load_file($new_path);
  else{
    $new_xml=new SimpleXMLElement("<obo
xmlns:xlink=\"http://www.w3.org/1999/xlink\"></obo>");
    $directory=dirname($new_path);
    if(!is_dir($directory))mkdir($directory);
  }
  $term=$xml->addChild('term');
  $new_term=$new_xml->addChild('term');
  $id=$_POST['id'];
  $term->addChild('id',$id);
  $new_term->addChild('id',$id);
  ... ...
  global $user_ID,$userdata;
  $new_xml->asXML($new_path);
  if($xml->asXML(ABSPATH."/wp-content/owl$file"))echo "Add Item
Successfully!";
  else echo "Add Item Failed!";
  $category=str_replace(GO_EXTENSION,"",$file);
  wp_redirect(get_option('home')."/wp-
includes/single.php?category=$category&ID=$id");
?>
```

3.11.2 Annotation Processor

Annotation Processor provides a friendly form interface as Figure 3.5. As mentioned in section 3.7.3, text field of “Evidence Code” can provide users with possible evidence codes according to the first few letters inputted in it. This is implemented by AJAX. GAS Blog compare the text in filed of “Evidence Code” with all the possible evidence codes described in section 3.5.4, and only lists out the one that begin with the text in the field of “Evidence Code” to help users choose the proper evidence code and, on the other hand, decrease the opportunity of typo.

Similar to GO Creator, Annotation Processor also composes XML fragment for newly added annotation according to the schema described in section 3.5.4. As mentioned earlier of this chapter, annotation data is not a part of Gene Ontology term schema, and they are saved in separate directory for convenient maintenance. Therefore, there should be a way to connect the annotation with its owner gene ontology term. XPointer is used in GAS Blog to do this task.

XML Pointer Language (XPointer) allows hyperlinks in a XML to point to specific parts of the same or another XML document[31]. In this way, referencing a Gene Ontology term's address in its annotation's namespace allows the annotation points to its owner Gene Ontology term. This referencing address in annotation namespace will be used by GO & Annotation Connector to link Gene Ontology term and annotation together. Details about connecting them will be addressed in later section of this chapter.

```
<?php
  ... ...
  $category =$_POST['category'];
  $evidence_code =$_POST['ec_input'];
  $user = wp_get_current_user();
  //deal with uploaded image
  $path_info = pathinfo($_FILES['image']['name']);
  $save_path="wp-content/upload/".time().".".$path_info['extension'];
  $is_attachment=($_FILES['image']['name']!="");
  if($is_attachment)
    if(!move_uploaded_file($_FILES['image']['tmp_name'],
$save_path)){
      print "Upload failed!";
      exit;
    }
  $path=ABSPATH."/wp-content/annotation$category".
ANNOTATION_EXTENSION;
  if(file_exists($path))$xml=simplexml_load_file($path);
  else{
    $xml=new SimpleXMLElement("<annotations
xmlns:xlink=\"http://www.w3.org/1999/xlink\"></annotations>");
    $directory=dirname($path);
    if(!is_dir($directory))mkdir($directory);
  }
  $annotation=$xml->addChild("annotation");
  $annotation->addAttribute("xlink:type","simple");
  $annotation->addAttribute("xlink:href",get_option('home')."/wp-
content/owl$category".GO_EXTENSION."#$comment_post_ID");
  $annotation->addChild('evidence_code',$evidence_code);
  ... ...
  if($is_attachment)
    $annotation->addChild('attachment',basename($save_path));
  $doc = new DOMDocument('1.0');
  $doc->formatOutput = true;
  $domnode = dom_import_simplexml($xml);
  $domnode = $doc->importNode($domnode, true);
  $domnode = $doc->appendChild($domnode);
  $doc->save($path);
  wp_redirect(get_option('home')."/wp-includes/single.php?category=
```

[31] http://www.w3schools.com/xlink/xlink_intro.asp

```
$category&ID=$comment_post_ID");
?>
```

3.11.3 GO & Annotation Connector

GO & Annotation Connector is a module to connect Gene Ontology terms with their annotations since they are stored separately in GAS Blog system. This process can also be called XML sequencing. This module relies on XPointer, with the reference address from annotation namespace; it finds out the owner Gene Ontology for each annotation, appends the annotation to its owner Gene Ontology terms, and stores the data in memory as a virtual XML file for other modules to use, for example, Syndication processor, GO Data Output processor, etc.

As mentioned before, XPointer enables XML node reference to another XML node with given address. Since annotation is stored separately with original GO terms, XPointer connects them together. In this way, annotation can be added to Gene Ontology without modifying the original Gene Ontology file.

Currently, PHP does not support XPointer while parsing XML. Under this circumstance, this thesis developed a module – GO & Annotation Connector to do so. While parsing a Gene Ontology file as XML, GAS Blog checks if its annotation file exists since annotation is stored in a separate file with same base name. If annotation file exists, the module will append annotation node as children to the GO term node according to the XPointer address; otherwise, the module will parse GO file as a single XML file. A virtual XML file will be generated in memory as the result of integration process in order to let other module use it.

```
<?php
  function integrateOWLandAnnotation($owl_path){
    $annotation_path=str_replace(ABSPATH."\wp-
content\owl",ABSPATH."\wp-content\annotation",$owl_path);
    $annotation_path=str_replace(basename($owl_path,GO_EXTENSION).
GO_EXTENSION,basename($owl_path,GO_EXTENSION).ANNOTATION_EXTENSION,
$annotation_path);
    if(file_exists($annotation_path)){
      $result_xml=new SimpleXMLElement($owl_path, NULL, TRUE);
      $annotation_xml=simplexml_load_file($annotation_path);
      $annotations=$annotation_xml->xpath("//annotation");
      foreach($annotations as $annotation){
        $href=$annotation->attributes()->href;
        $id=substr($href,strpos($href,'#')+1);
        $gos=$result_xml->xpath("//term[id='$id']");
        $go=$gos[0];
        $go_annotation=$go->addChild("annotation");
        $go_annotation->addChild("evidence_code",$annotation-
>evidence_code);
        $go_annotation->addChild("content",$annotation->content);
        $foaf=$go_annotation->addChild("foaf");
        $foaf->addChild("author",$annotation->foaf->author);
        $groups=$foaf->addChild("groups");
        foreach($annotation->foaf->groups as $group)$groups-
>addChild("group",$group->group);
        $go_annotation->addChild("attachment",$annotation-
>attachment);
        $category=str_replace(GO_EXTENSION,"",
str_replace(ABSPATH."\wp-content\owl","",$owl_path));
        $go->addChild("category",$category);
      }
      return $result_xml;
    }else return simplexml_load_file($owl_path);
  }
?>
```

3.11.4 User Data Collector

User Data Collector is a module that collects users' information such as user name, email address, user interested gene ontology categories, user groups joined, friendship with other users, etc., through a form interface and save in MySQL database for other module to use such as Graphical FOAF Generator.

This section takes adding friend information as an example for pseudo code, and adding the other user information to MySQL database is similar to this one.

```
<?php
  $user_ID=$_GET['user'];
  $friend_name=$_GET['friend'];
  ... ...
  $friends=$wpdb->get_results("select * from $wpdb->users where
user_login='$friend_name'");
  if(sizeof($friends)==0){
    echo "user name does not exist!";
    exit;
  }
  $friend_ID=$friends[0]->ID;
  if($friend_ID==$user_ID){
    echo "you can not add yourself as a friend!";
    exit;
  }
  $wpdb->query("SELECT wp_foaf.* FROM wp_foaf WHERE
friend1_ID=$user_ID AND friend2_ID=$friend_ID OR friend2_ID=$user_ID
AND friend1_ID=$friend_ID");
  if($wpdb->num_rows==0){
    $wpdb->query("INSERT INTO wp_foaf (friend1_ID,friend2_ID) VALUES
($user_ID,$friend_ID)");
    echo "Friend added successfully!";
    exit;
  }
  echo "$friend_name is already your friend!";
?>
```

3.11.5 Graphical FOAF Generator

Graphical FOAF Generator is a module providing users a visual image of the FOAF network all over GAS Blog as Figure 3.9. The module generates a dynamical image with the data from user data storage (MySQL database), and display it in a web page.

There are three classes in Graphical FAOF Generator: FriendNode, Relationship, and GraphicalFOAF. Their relationship is as below.

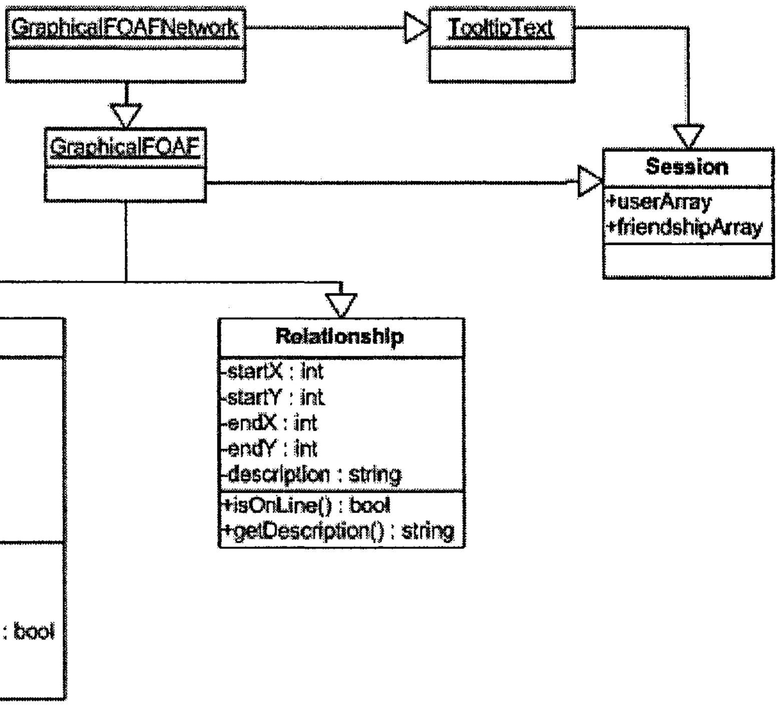


Figure 3.16 Sequential Diagram of Graphical FOAF Generator

GraphicalFOAFNetwork is the output web page which displays the dynamically generated image. It calls class of GraphicalFOAF to generate the image, and call class of TooltipText to display tooltip when it receives the event of users' mouse hovering on the image.

```
<html>
  <body>
    <script type="text/javascript" src="js/wz_tooltip.js"></script>
    <script>
      function showTooltip(){
        var xmlHttp;
        try{xmlHttp=new XMLHttpRequest();}
        catch (e){
          try{xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");}
          catch (e){
            try{xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");}
            catch (e){
              alert("Your browser does not support AJAX!");
```

```
                return false;
              }
            }
          }
          xmlHttp.onreadystatechange=function(){
            if(xmlHttp.readyState==4)Tip(xmlHttp.responseText, SHADOW,
true);
          }
          xmlHttp.open("GET","TooltipText.php?x="+ window.event.clientX
+"&y="+window.event.clientY,true);
          xmlHttp.send(null);
        }
      </script>
      <img src="GraphicalFOAF.php" onmousemove="showTooltip()"
onmouseclick="UnTip()"/>
   </body>
</html>
```

GraphicalFOAF is the core of generating Graphical FOAF image. It composes object for users and their relationship in GAS Blog system. All the user and relationship objects are stored in an array. These arrays will be stored in browser session after drawing the user nodes and their relationship onto image in order to let TooltipText to access the data.

```
<?php
$image_width=1200;
$image_height=630;
$image = imagecreatetruecolor($image_width, $image_height);
imagefilledrectangle($image, 0, 0, $image_width, $image_height,
imagecolorallocate($image, 245, 245, 245));
... ...
require_once('FriendNode.php');
$author_array=array();
$authors = $wpdb->get_results("SELECT ID, user_nicename from $wpdb-
>users");
//allocate color for groups
$group_colors=array();
$groups=$wpdb->get_results("select ID,name from wp_group");
//draw image explanation rectangle
imagefilledrectangle($image,0,0,90,sizeof($groups)*14+10,imagecoloral
locate($image,220,220,220));
$y=8;
foreach($groups as $group){
  $group_colors[$group-
>ID]=imagecolorallocate($image,rand(0,255),rand(0,255),rand(0,255));
  imagerectangle($image, 10, $y, 30, $y+10, $group_colors[$group-
>ID]);
  imagestring($image,3,33,$y-1,$group->name,$group_colors[$group-
>ID]);
  $y+=14;
}
imagesetthickness ($image ,2);
//draw user node
```

```
foreach((array)$authors as $author){
  $x=rand(90,$image_width-90);
  $y=rand(90,$image_height-90);
  $description="<div><h6>User Name: $author->user_nicename</h6>";
  $user_groups=$wpdb->get_results("SELECT wp_group.name,wp_group.ID
from wp_group left join user_group on wp_group.ID=user_group.group_ID
where user_group.user_ID=$author->ID");
  $color;
  if(sizeof($user_groups)==0)$description.="No groups joined.";
  else $description.="Groups:";
  foreach($user_groups as $group){
    $description.="<br>&nbsp;&nbsp;&nbsp;&nbsp;".$group->name;
    $color=$group_colors[$group->ID];
  }
  $user_categories=$wpdb->get_results("select distinct category,time
from user_interested_category where user_ID=$author->ID");
  if(sizeof($user_categories)==0)$description.="<br>No interested in
any categories";
  else $description.="<br>Interested Categories:";
  foreach($user_categories as $category){
    $description.="<br>&nbsp;&nbsp;&nbsp;&nbsp;".$category->category;
  }
  $author_array[$author->ID]=new FriendNode($author-
>user_nicename,$image,$x,$y,$author-
>ID,$description."</div>",$color);
}
$relationships=$wpdb->get_results("select * from wp_foaf");
require_once('Relationship.php');
$relationship_array=array();
foreach($relationships as $relationship){
  $relationship_array[$relationship->ID]=new
Relationship($image,$author_array[$relationship->friend1_ID]-
>getX(),$author_array[$relationship->friend1_ID]-
>getY(),$author_array[$relationship->friend2_ID]-
>getX(),$author_array[$relationship->friend2_ID]-
>getY(),imagecolorallocate($image,0,0,0),"Friendship between
".$author_array[$relationship->friend1_ID]->getName()."
and".$author_array[$relationship->friend2_ID]->getName());
}
header("Content-type: image/png");
imagepng($image);
imagedestroy($image);
session_start();
$_SESSION['authors'] = $author_array;
$_SESSION['relationship']=$relationship_array;
?>
```

FriendNode is an object for GAS Blog users. It contains the following information (attributes) of users.

- X: X coordinator of node position;
- Y: Y coordinator of node position;

- ID: user ID in GAS Blog;
- Description: the text displayed in node tooltip such as user group joined, interested gene ontology categories;
- Name: user name;
- Node_width: the width of user node on image;
- Node_height: the height of user node on image.

To ensure this information can be accessed outside of this class, it provides the some functions:

- getX: get the X coordinator of a user node on imagel
- getY: get the Y coordinator of a user node on image;
- isFriend (friend_ID): return a boolean type value to indicate whether the friend_ID parsed in is a friend with the current user object or not;
- isCoordinatesInBound (mouse_X, mouse_Y): this is a function to check if the coordinates of mouse is in range of user node so GraphicalFOAF module can decide whether display tooltip or not. User node tooltip will only be displayed when coordinator of mouse in bound of user node object on image;
- getDescription: get the description of user node in order to display in tooltip;
- getName; get the user name.

```
<?php
class FriendNode{
  var $x;
  var $y;
  var $ID;
  var $description;
  var $name;
  var $node_width;
  var $node_height;
  function FriendNode($name,$image,$x,$y,$ID,$description,$node_color
){
    if($image!=''){
      $node_image = @imagecreatefromjpeg("images/node.jpeg");
      $this->node_width=imagesx($node_image);
      $this->node_height=imagesy($node_image);
      $this->x=$x+$this->node_width/2;
      $this->y=$y+$this->node_height/2;
      $this->description=$description;
      imagecopy($image,$node_image,$x,$y,0,0,$this->node_width,$this-
>node_height);
```

```
      imagestring($image, 7,$x+$this->node_width-2,$y+$this-
>node_height-4, $name, $node_color);
      $this->name=$name;
      imagerectangle($image,$x,$y,$x+$this->node_width,$y+$this-
>node_height,$node_color);
    }
    $this->$ID=$ID;
  }

  function getX(){return $this->x;}

  function getY(){return $this->y;}

  function isFriend($friend_ID){
    $count=mysql_query("SELECT COUNT(*) as 'count' from wp_foaf where
friend1_ID=$ID and friend2_ID=$friend_ID or friend2_ID=$ID and
friend1_ID=$friend_ID");
    return $count->count!=0;
  }

  function isCoordinatesInBound($mouseX,$mouseY){
    if($mouseX>=$this->x&&$mouseX<=$this->x+$this-
>node_width&&$mouseY>=$this->y&&$mouseY<=$this->y+$this-
>node_height)return 1;
    return 0;
  }

  function getDescription(){return $this->description;}

  function getName(){return $this->name;}
}
?>
```

Relationship is an object of friendship among GAS Blog users. It contains the following information (attributes):

- startX: the start X-coordinator of relationship line on image;
- startY: the start Y-coordinator of relationship line on image;
- endX: the end X-coordinator of relationship line on image;
- endX: the end Y-coordinator of relationship line on image;
- description: the text displayed on tooltip;

As FriendNode object, it also provides some functions for accessing these data externally.

- isOnLine (mouse_X, mouse_Y): to check whether the mouse coordinator is on relationship line. GraphicalFOAF module only displays tooltip when the mouse hover on relationship line in the image;

- getDescription: get the relationship description for tooltip.

```
<?php
class Relationship{
  var $startX;
  var $startY;
  var $endX;
  var $endY;
  var $description;

  function
Relationship($image,$startX,$startY,$endX,$endY,$color,$description){
    if($image!=''){
      $this->startX=$startX;
      $this->startY=$startY;
      $this->endX=$endX;
      $this->endY=$endY;
      imageLine($image,$this->startX,$this->startY,$this->endX,$this-
>endY,$color);
      $this->description=$description;
    }
  }

  function isOnLine($x,$y){
    if($this->startX!=$this->endX){
      $x1=$this->startX;
      $y1=$this->startY;
      $x2=$this->endX;
      $y2=$this->endY;
      $a=($y1-$y2)/($x1-$x2);
      $b=($x1*$y2-$x2*$y1)/($x1-$x2);
      $standardY=$a*$x+$b;
      if($y>=$standardY-
40&&$y<=$standardY+40&&$x>=min($x1,$x2)&&$x<=max($x1,$x2)&&$y>=min($y
1,$y2)&&$y<=max($y1,$y2))return 1;
      return 0;
    }else{
      if($x<=$this->startX+40&&$x>=$this->startX-
40&&$x>=min($x1,$x2)&&$x<=max($x1,$x2)&&$y>=min($y1,$y2)&&$y<=max($y1
,$y2))return 1;
      return 0;
    }
  }

  function getDescription(){return $this->description;}
}
?>
```

Tooltiptext is a class to display tooltip on Graphical FOAF Network image. It retrieves array of FriendNode and Relationship object from session, and check what kind of tooltip should be displayed according to the mouse coordinator receiving from class GraphicalFOAFNetwork.

```
<?php
  require_once('FriendNode.php');
  require_once('Relationship.php');
  session_start();
  $authors=$_SESSION['authors'];
  $x=$_GET['x'];
  $y=$_GET['y'];
  foreach($authors as $author){
    if($author->isCoordinatesInBound($x,$y)){
      echo $author->getDescription();
      exit;
    }
  }
  $relationships=$_SESSION['relationship'];
  foreach($relationships as $relationship){
    if($relationship->isOnLine($x,$y)){
      echo $relationship->getDescription();
      exit;
    }
  }
?>
```

3.11.6 Syndication Processor

Syndication Processor is the module to generate dynamic syndication feeds of newly added Gene Ontology or existing Gene Ontology annotation according to users' interest on Gene Ontology categories. The schema of syndication feeds is introduced in section 3.5.7.

GAS Blog uses templates of syndication to build up main structure of syndication feeds, and fill the content according to the data prepared by module GO & Annotation Connector using script language PHP.

GAS Blog uses SimpleXMLElement, a class for XML operation, to load real or virtual XML files, and uses XPath to query useful information to fill in syndication feeds templates.

This section takes annotation syndication code as pseudo code as example; the syndication processor code for new found gene ontology is similar to this one.

```
<?php
header('Content-Type: text/xml; charset=UTF-8' , true);
$more = 1;
global $rss_user_ID;
?>
<?php echo '<?xml version="1.0"
encoding="'.get_option('blog_charset').'"?'.'>'; ?>
<?php
header('Content-Type: text/xml; charset=UTF-8' , true);
$more = 1;
```

```
global $rss_user_ID;
?>
<?php echo '<?xml version="1.0"
encoding="'.get_option('blog_charset').'"?'.'>'; ?>

<rss version="2.0"
      xmlns:content="http://purl.org/rss/1.0/modules/content/"
      xmlns:wfw="http://wellformedweb.org/CommentAPI/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:rss-foaf="http://rss-extensions.org/wiki/Rss-foaf"
      xmlns:rss-pm="http://rss-extensions.org/wiki/Rss-foaf"
      xmlns:lom-gen="http://kmr.nada.kth.se/el/ims/schemas/lom-
general"
      xmlns:lom-life="http://kmr.nada.kth.se/el/ims/schemas/lom-
lifecycle"
      xmlns:lom-meta="http://kmr.nada.kth.se/el/ims/schemas/lom-
metadata"
      xmlns:lom-tech="http://kmr.nada.kth.se/el/ims/schemas/lom-
technical"
      xmlns:lom-edu="http://kmr.nada.kth.se/el/ims/schemas/lom-
educational"
      xmlns:lom-rights="http://kmr.nada.kth.se/el/ims/schemas/lom-
rights"
  <?php do_action('rss2_ns'); ?>
>

<channel>
      <title><?php bloginfo_rss('name'); wp_title_rss(); ?></title>
      <link><?php bloginfo_rss('url') ?></link>
      <description><?php bloginfo_rss("description") ?></description>
      <pubDate><?php echo mysql2date('D, d M Y H:i:s +0000',
get_lastpostmodified('GMT'), false); ?></pubDate>
      <generator>http://wordpress.org/?v=<?php
bloginfo_rss('version'); ?></generator>
      <language><?php echo get_option('rss_language'); ?></language>
      <?php
    do_action('rss2_head');
    $interested_categories=$wpdb->get_results("select distinct
category,time from user_interested_category where
user_ID=$rss_user_ID");
    foreach($interested_categories as $category):
      $path=ABSPATH."/wp-content/annotation$category-
>category".ANNOTATION_EXTENSION;
      if(!file_exists($path))continue;
      $annotation_xml=simplexml_load_file($path);
      $annotations=$annotation_xml-
>xpath("//annotation[//@time>$category->time]");
      foreach($annotations as $annotation) :
  ?>
      <item>
            <title><?php
              $xml = simplexml_load_file(ABSPATH."/wp-
content/owl$category->category".GO_EXTENSION);
              $href=$annotation->attributes()->href;
      $id=substr($href,strpos($href,'#')+1);
      $terms = $xml->xpath("//term[id=\"$id\"]");
      $title=ucwords($terms[0]->name);
      $annotator=$annotation->foaf->author;
      echo $title." by ".$annotator." at ".$annotation->timestamp;
    ?></title>
            <link><?php
              $time=$annotation->timestamp->attributes()->time;
      echo get_option('home')."/wp-
includes/single.php?category=$category-
>category&#38;ID=$id&#35;annotation-$time";
    ?></link>
            <annotations><?php echo get_option('home')."/wp-
includes/single.php?category=$category-
>category&#38;ID=$id&#35;annotation-$time"; ?></annotations>
            <dc:annotator><?php
      echo $annotator;
    ?></dc:annotator>
            <!-- FOAF section-->
            <rss-foaf:type name="<?php //full name here, if user does
not have full name registered, use user name instead
      echo $annotator;
    ?>"/>
    <rss-foaf:group name="<?php //category name here right now
      echo $category->category;
    ?>"/>
    <?php
      $relationships=$wpdb->get_results("select user_nicename from
wp_users left join wp_foaf on wp_foaf.friend2_ID=wp_users.ID where
wp_foaf.friend1_ID=$rss_user_ID");
      foreach($relationships as $relationship):
    ?>
      <rss-foaf:rel to="<?php echo $relationship->user_nicename;
?>"/>
    <?php
      endforeach;
      $relationships=$wpdb->get_results("select user_nicename from
wp_users left join wp_foaf on wp_foaf.friend1_ID=wp_users.ID where
wp_foaf.friend2_ID=$rss_user_ID");
      foreach($relationships as $relationship):
    ?>
      <rss-foaf:rel to="<?php echo $relationship->user_nicename;
?>"/>
    <?php endforeach;?>
    <!-- learning object section -->
    <dc:identifier><?php echo $id ?></dc:identifier><!-- post id-->
    <dc:title><?php echo $title; ?></dc:title><!-- post title-->
      <dc:language>en</dc:language>
      <dc:description><?php echo $annotation->content;
?></dc:description>
            <dc:subject><?php //category name here
      echo $category->category;
    ?></dc:subject>
    <dc:evidence_code><?php //category name here
      echo $annotation->evidence_code;
    ?></dc:evidence_code>
    <lom-gen:structure>Atomic</lom-gen:structure>
    <lom-gen:aggregationLevel>13</lom-gen:aggregationLevel>
      <lom-life:version>beta</lom-life:version>
      <lom-life:status>final</lom-life:status>
    <dc:publisher>Lakehead University</dc:publisher>
    <dc:editor><?php //full name here, if user does not have full
name registered, use user name instead
              echo $annotator;
    ?></dc:editor>
    <lom-meta:metadataScheme resource="lom-meta;LOMv1.0"/>
    <dc:format>text/html</dc:format>
      <lom-tech:operatingSystem>Multi-OS</lom-tech:operatingSystem>
    <lom-edu:interactivityType>Active</lom-edu:interactivityType>
    <lom-edu:type>Exercise</lom-edu:type>
    <lom-edu:interactivityLevel>Low</lom-edu:interactivityLevel>
      <lom-edu:semanticDensity>High</lom-edu:semanticDensity>
    <lom-edu:intendedEndUserRole>Manager</lom-
edu:intendedEndUserRole>
    <lom-edu:context>School</lom-edu:context>
    <lom-edu:difficulty>Easy</lom-edu:difficulty>
    <lom-edu:typicalLearningTime>PT1H20M</lom-
edu:typicalLearningTime>
    <lom-edu:language>en</lom-edu:language>
    <lom-rights:copyrightAndOtherRestrictions>Lakehead
University</lom-rights:copyrightAndOtherRestrictions>
    <dc:rights>RightsBroker:RightsModel</dc:rights>
      </item>
      <?php endforeach;endforeach;?>
      </channel>
</rss>
```

3.11.7 GO Data Output Processor

GO Data Output Processor is another output module as Syndication Processor. Difference between GO Data Output Processor and Syndication Processor is that the former is for machine clients and the latter is for human users. GO Data Output Processor translates virtual XML data prepared by GO & Annotation Connector into web page text and display them in a well-formatted way; therefore human users can read it from their browser easily.

Similar to Syndication Processor, GO Data Output Processor uses SimpleXMLElement to load XML files and query with XPath to get the information needs to be displayed on web pages.

```
<?php
function printEntryDIV($term,$xml,$category){
?>
<div class="entry">
  <p>ID: <?php echo $term->id;?></p>
  <p>Name: <?php echo $term->name;?></p>
  <?php if($term->synonym):?>
    <p><?php echo ucfirst($term->synonym->attributes()->scope);?>
Synonym: <?php echo $term->synonym->synonym_text;?></p>
  <?php endif;?>
  <p>NameSpace: <?php echo $term->namespace;?></p>
  <?php if($term->is_a):?>
    <p>Subclass of: <a href="<?php
      $synonym_terms=$xml->xpath("//term[id=\"$term->is_a\"]");
      $synonym_term=$synonym_terms[0];
      echo get_option('home')."\wp-
includes\single.php?category=$category&ID=$term->is_a";
    ?>"><?php echo $synonym_term->name;?></a></p>
  <?php endif;?>
  <?php if($term->relationship):?>
    <p>Relationship: <?php
      $relationship=$term->relationship;
      $to_terms=$xml->xpath("//term[id=\"$relationship->to\"]");
      $to_term=$to_terms[0];
      if(substr_count($relationship->type,":")>0)echo
str_replace("_"," ",substr($relationship->type,strpos($relationship-
>type,":")+1))." ";
      else echo str_replace("_"," ",$relationship->type)." ";
    ?><a href="<?php echo get_option('home')."\wp-
includes\single.php?category=$category&ID=$relationship-
>to";?>"><?php echo $to_term->name;?></a></p>
  <?php endif;?>
  <?php if($term->def):?>
    <p>Definition: <?php echo $term->def->defstr;?></p>
  <?php endif;?>
</div>
<?php
}
```

Chapter Four
Navigation Gene Ontology Using GAS

Other than the features introduced in last chapter, there is another useful and important feature of GAS Blog – Gene Ontology Navigation (GON). Gene Ontology Navigation is a new generation of Gene Ontology querying service based on Web 2.0 and semantic web technology. Unlike the traditional Gene Ontology search engines such as AmiGO, GON is guided with FOAF and metadata for more accurate search result. In GAS Blog, GON can be divided into two sub-navigation modules, one is Collaborative Navigation, and the other is Generic Navigation.

This chapter starts from introducing what Web 2.0 navigation and semantic web navigation is and describing Collaborative Navigation and Generic Navigation respectively.

4.1 Navigation based on Web 2.0 & Semantic Web

Traditionally, Gene Ontology query like AmiGO is completely based on database searching primitives. After users fill a search form, search engine goes through database and return all the matched results. Among these returned results, users can also go through category step by step to narrow down the amount of results and get what they are looking for. This search process can be considered as a blind search that is guided by a top-down search mechanism.

Unlike usual methods of searching, new generation Gene Ontology Navigation engine is based on Web 2.0 technologies and semantic web. Instead of querying through database tables, it relies on metadata extracted dynamically from Gene Ontologies and their annotations (Fiaidhi, Mohammed, JAAM, & HASNAH, 2003). Other than that, GON uses some tag magnet and spelling correction to improve search result accuracy and relevance.

There are several key differences between the traditional Gene Ontology search engine and GON.

Query Method: Traditional GO search engines do the query database by SQL statement and GON perform query by metadata which is more flexible and accurate on query.

Dictionary Support: Dictionary is another important part in Gene Ontology Navigation engine. It allows GON engine extracts metadata of meaningful word, find semantically matched information as navigation result instead of information matched on spelling only.

Flexible Metadata Matching: Since traditional GO search engines are based on database, the title in each database table is fixed and can not be modified after the creation; however, metadata does not have this limitation, it is dynamic and semantic. With support of dictionary, navigation command does not have to use fixed keyword as traditional GO search engine, all synonyms of background keyword is acceptable to GON engine.

Typo Handling: When users annotate to a Gene Ontology term, GAS Blog provides possible evidence code to users to decrease the chance of typo. Traditional GO search engines do not have this feature.

Metadata Extraction: Metadata is important for people and computer to understand annotation resource and perform query. GON extracts metadata dynamically as keyword to each annotation according to an algorithm and support of a dictionary. The reason of extracting metadata is to accommodate the new annotations and tag made by users. This is even more reliable than using a global ontology for searching. Traditional GO search engine does not have metadata along with annotation at all.

The following table shows the comparison mentioned above.

	Traditional GO Search Engine	GON
Query Method	Database	Metadata
Metadata Matching	Fixed	Flexible
Typo Handling	No	Yes
Metadata Extraction	No	Yes
Dictionary Support	No	Yes

Table 4.1 Gene Ontology Search Engine Comparison Table

4.2 Gene Ontology Navigation

Gene Ontology Navigation in GAS Blog system is consisted by two sub-navigation services: Collaborative Navigation and Generic Navigation. The former is to search Gene Ontology and annotation information inside a group and the latter is for search Gene Ontology and annotation information cross all GAS Blog user groups.

However, these two sub-navigation services have user interface in common. This section will introduce the common part first and explain the difference on each one later.

4.2.1.1 Common User Interface

The user interface for both collaborative and generic Gene Ontology Navigation is the same, and they follow the same query grammar and rules. Figure 4.1 below is the user interface form for navigation.

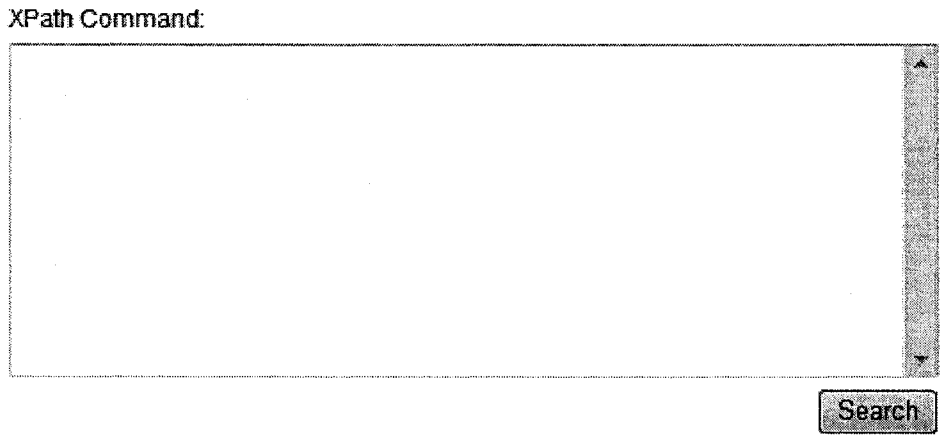


Figure 4.1 Navigation Form

There is a simple XPath grammar of navigation command for users to fill in the form shown in Figure 4.1. Using this grammar, navigation services can search according to specific conditions, fuzzy conditions, or combined conditions. Table 4.2 lists some sample navigation command below.

//term	select all GO terms in all categories.
//term[id="MA:0000001"]	select GO term with id "MA:0000001"
//term[name="brenda source tissue ontology"]	select GO term with name "brenda source tissue ontology"
//term[namespace="caro"]	select GO term with namespace "caro"
//term[annotation]	select GO term with annotation
//term[annotation/foaf/author="admin"]	select GO term with annotation from user "admin"
//term[annotation/foaf/groups/group="Group 1"]	select GO term with annotation from "Group 1"
//term[annotation/evidence_code="TAS"]	select GO term with annotation evidence code "TAS"
//term[contains(id, 'MA')]	select GO term with id contains "MA"
//term[contains(name,"ana")]	select GO term with name contains "ana"

//term[contains(annotation/foaf/groups/group,"oup")]	select GO term with group name contains "oup"

Table 4.2 Gene Ontology Navigation Command Table

Moreover, XPath search command conditions can be combined in format of: //term[namespace="caro"][id="MA:0000001"] or use "and" or "or" to combine several conditions together like //term[namespace="caro" or id="MA:0000001"] for more complicated navigation requirement.

4.2.1.2 Gene Ontology Navigation Dictionary

Gene Ontology Navigation relies on a dictionary for the features of flexible metadata matching, semantic query, and dynamical metadata extraction. The internal dictionary used by Gene Ontology Navigation engine is WordNet. WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet's structure makes it a useful tool for computational linguistics and natural language processing. (Miller, Fellbaum, Tengi, Wakefield, Langone, & Haskell)

Flexible metadata matching allows users use synonym word of standard background keyword as query range. Unlike SQL database query statement, keyword/title of each table is fixed and users can not use other word with similar meaning to perform the query; however, flexible metadata matching ensures that (Fiaidhi, Passi, & Mohammed, 2004). For example, navigation command "//term[annotation/foaf/author="admin"]" is a standard command for searching GO term annotated by author "admin". However, instead of calling annotator "author", some people might want to use the word "writer" which is a synonym of "author". With flexible metadata matching technology, GON engine matches the word "writer" to "author" in background and perform the command properly.

Semantic query is similar to flexible metadata. Unlike flexible metadata is to match synonym metadata, semantic query match the result with synonyms. For example, navigation command "//term[contains(annotation/description,"home")]" is used to query the GO terms whose annotation description contains word "home"; however, there might be some annotators prefer using the word "family" and they are similar on meaning. At this time, traditional GO search engines which based on SQL database query are not able to find out the annotation whose description used the word "family" because they are different on spelling. GON engines can recognize them after checking the dictionary to make sure that they are synonyms and return it as matched result.

Dynamical metadata extraction is used to extract metadata dynamically from given annotation description after looking up the dictionary and extract the meaningful word as metadata.

4.2.1.3 Gene Ontology Navigation Implementation

Gene Ontology Navigation is consisted of three modules: Category Redirection, Collaborative Navigation, and Generic Navigation. Among them, Category Redirection is the module shared by Collaborative and Generic Navigation module.

- Category Redirection: Category Redirection is a module to narrow down the query result into a certain gene ontology category in order to decrease the amount of un-useful information.
- Navigation Processor: Navigation Processor is used to fill the matched navigation query result redirected from Category Redirection module into SCORM manifest format based on navigation types – Collaborative or Generic.
- Metadata Extractor: Metadata Extractor is a module to extract metadata dynamically from Gene Ontology annotation description with support of dictionary.

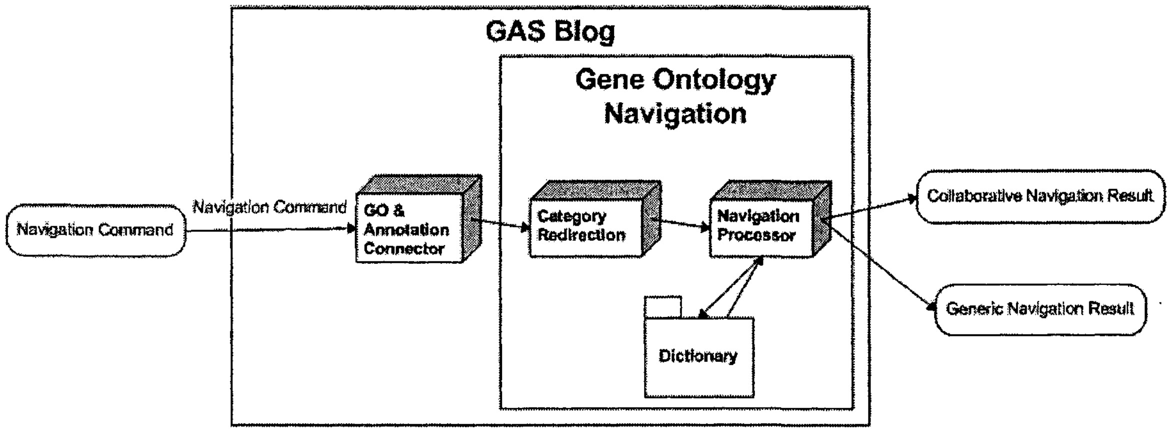


Figure 4.2 Gene Ontology Navigation Architecture

4.2.1.3.1 Category Redirection Implementation

Category Redirection module accepts navigation command from users' input and perform a pre-search for matched Gene Ontology information based on integrated Gene Ontology and annotation data transferred from GO & Annotation Connector. The pre-search just count and display the number of Gene Ontology terms matched the query command, and lead users go through the search output module – Navigation Processor.

The following pseudo code reflects the logic of Category Redirection described above.

```
  function queryCategory($path,$query_command){
    if(is_dir($path)){
      $dir_handle = @opendir($path) or die("Unable to open
$path<br>");
      while($file = readdir($dir_handle)){
        if($file == "." || $file == "..")continue;
        if(is_file($path."/".$file)){
          $category=str_replace(ABSPATH."\wp-
content\owl","",$path)."/".basename($file, GO_EXTENSION);
          $xml = integrateOWLandAnnotation("$path/$file");
          $terms =$xml->xpath($query_command);
          if(sizeof($terms)>0){
            $display_category=basename($file,GO_EXTENSION);
            ?>
            <li><a href="query_result.php?category=<?php echo
$category;?>&type=<?php echo $_POST['type']; ?>" name="<?php echo
$category;?>" value="<?php echo $display_category;?>"><?php echo
ucwords(str_replace("_"," ",$display_category));?> (<?php echo
sizeof($terms);?>)</a></li>
            <div id="<?php echo $display_category;?>" name="<?php
echo $display_category;?>"></div>
            <?php
          }
        }else if(is_dir($path."/".$file))
          queryCategory($path."/".$file,$query_command);
      }
      closedir($dir_handle);
    }
  }
```

4.2.1.3.2 Navigation Processor Implementation

Navigation Processor Implementation for Collaborative and Generic Navigation is similar. A condition statement is used in Navigation Processor to handle the different output SCORM manifest in code. For Collaborative Navigation, GON appends some XPath command by the end of query command collected from users' input to limit the range of query in the users' group. As for Generic Navigation, GON perform query command from users' input.

Following pseudo code reflects the logic mentioned above.

```
<?php
  require_once("../wp-blog-header.php");
  header('Content-Type: text/xml; charset=UTF-8' , true);
  echo '<?xml version="1.0"
encoding="'.get_option('blog_charset').'"?'.'>';
  $query_type=$_GET['type'];
  global $user_ID;
  session_start();
  $command=$_SESSION['command'];
  $user_groups=array();
  if($query_type=="groupwise"){
    $groups=$wpdb->get_results("select wp_group.name from wp_group
left join user_group on user_group.group_ID=wp_group.ID where
user_group.user_ID=$user_ID");
    if(sizeof($groups)>0){
      $group_command="[";
      foreach($groups as $group){
        $group_command.="annotation/foaf/groups/group=\"$group-
>name\" or";
        array_push($user_groups,$group->name);
      }
      $group_command=substr($group_command, 0, -3);
      $group_command.="]";
      $command.=$group_command;
    }
  }
  $category=$_GET['category'];
  require_once(ABSPATH."wp-
content\plugins\SemanticFeatures\IntegrationFunction.php");
  $xml=integrateOWLandAnnotation(ABSPATH."\wp-
content\owl".$category.GO_EXTENSION);
  $terms=$xml->xpath($command);
?>
```

4.2.1.3.3 Metadata Extractor Implementation

Metadata Extractor dynamically extracts metadata from annotation descriptions of Gene Ontology terms with support of dictionary WordNet.

After receiving annotation description from Navigation Processor, Metadata Extractor checks each single word of the desciption in dictionary and if it is a noun, Metadata Extractor will consider it as a valid metadata. After extracting all the metadata from annotation description, Metadata Extractor send all the metadata back to Navigation Processor to continue the navigation task.

The following pseudo code reflects the logic mentioned above.

```
  function extractMetadata($string_array){
    $metadatas=array();
    foreach($string_array as $string){
      $words=explode(" ", $string);
      foreach($words as $word){
        $word_array=str_split($word);
        if(ord($word_array[0])<65||ord($word_array[0])>122)
          $word_array[0]="";
        if(ord($word_array[sizeof($word_array)-1])<65||
ord($word_array[sizeof($word_array)-1])>122)
          $word_array[sizeof($word_array)-1]="";
        $word=implode("",$word_array);
        $result=mysql_query("select distinct wordnet30.synset.pos
from wordnet30.synset left join wordnet30.sense on
wordnet30.sense.synsetid=wordnet30.synset.synsetid left join
wordnet30.word on wordnet30.sense.wordid=wordnet30.word.wordid where
wordnet30.word.lemma like '$word' and wordnet30.synset.pos='n'");
        $num_rows = mysql_num_rows($result);
        if($num_rows!=0&&!in_array($word,$metadatas))
          array_push($metadatas,$word);
      }
    }
    for($i=0;$i<sizeof($metadatas);$i++){
      $meta1=$metadatas[$i];
      for($j=$i+1;$j<sizeof($metadatas);$j++){
        $result=mysql_query("select w2.lemma from wordnet30.sense,
(select wordnet30.sense.synsetid from wordnet30.word as w1 left join
wordnet30.sense on w1.wordid=sense.wordid where w1.lemma='$meta1')as
tmp, wordnet30.word as w2 where wordnet30.sense.synsetid=tmp.synsetid
and w2.wordid=wordnet30.sense.wordid and w2.lemma='$metadatas[$j]'");
        $num_rows = mysql_num_rows($result);
        if($num_rows!=0){
          $metadatas[$i]=$metadatas[$i]." / ".$metadatas[$j];
          if($debug=="debug")echo "<br>----metadata \"$meta1\" is
synonym to metadata \"$metadatas[$j]\"<br>";
          if($j==sizeof($metadatas)-1)array_pop($metadatas);
          else $metadatas[$j]=array_pop($metadatas);
        }
      }
    }
    return $metadatas;
  }
```

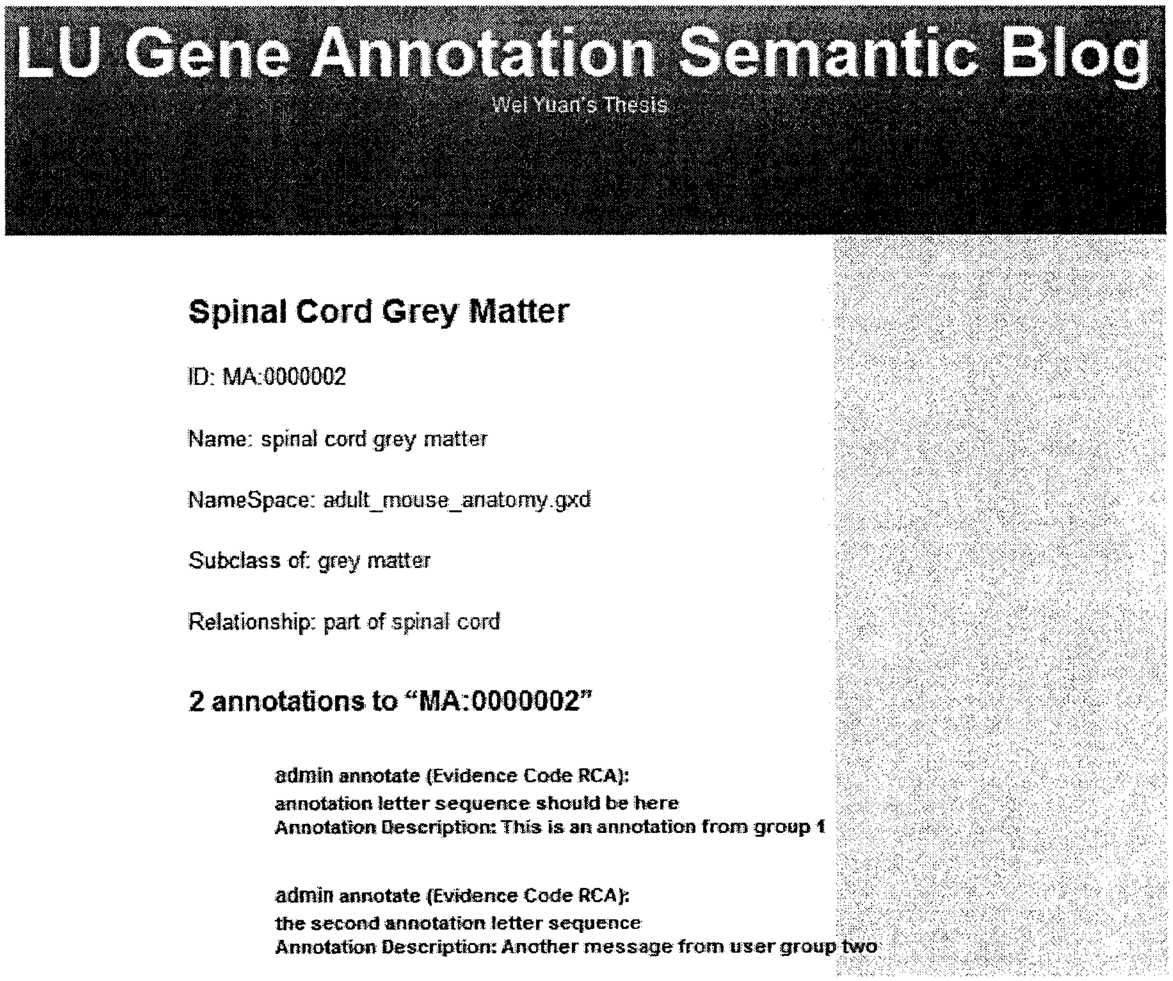
4.2.2 Collaborative Navigation

Collaborative Navigation is a group-wise navigation. It allows users to search Gene Ontologies or their annotation based on a specific navigation condition only in the groups they joined under the guide and assistance of FOAF feature of GAS Blog.

4.2.2.1 Collaborative Navigation Usage Scenario

Collaborative Navigation is only available to registered GAS Blog users, and users have to login before using this feature. To use Collaborative Navigation for browsing Gene Ontology, users can find the link of "Group-wise Metadata Search" on the first page of GAS Blog system. This link is only visible to users logged in GAS Blog and it will lead users to a simple form to Collaborative Navigation as Figure 4.2.

Here is a realistic example of Collaborative Navigation usage scenario. As Figure 4.3 displays, Gene Ontology for Spinal Cord Grey Matter with ID "MA: 0000002" has two annotations, one from Group 1, and the other from Group 2.



Spinal Cord Grey Matter

ID: MA:0000002

Name: spinal cord grey matter

NameSpace: adult_mouse_anatomy.gxd

Subclass of: grey matter

Relationship: part of spinal cord

2 annotations to "MA:0000002"

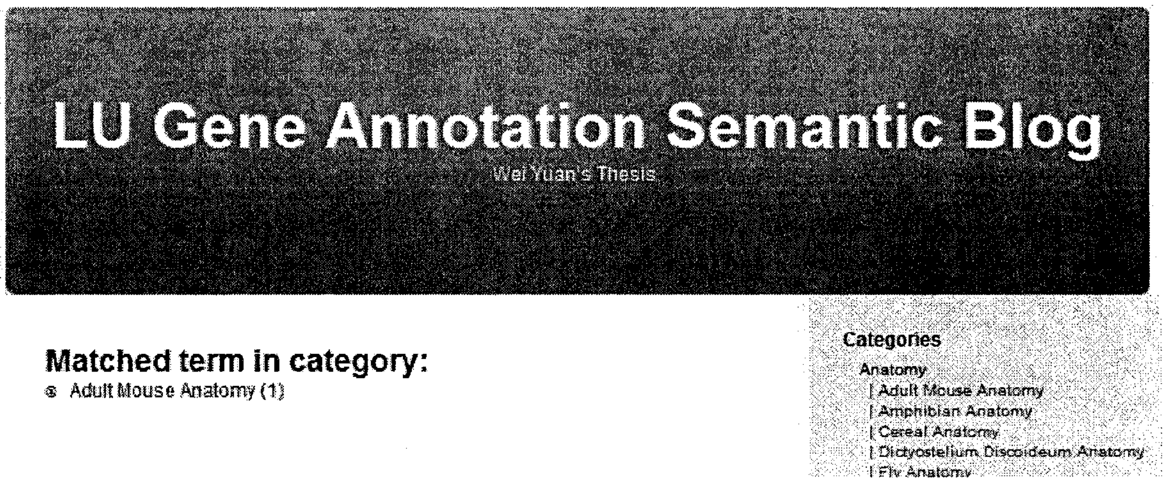
admin annotate (Evidence Code RCA):
annotation letter sequence should be here
Annotation Description: This is an annotation from group 1

admin annotate (Evidence Code RCA):
the second annotation letter sequence
Annotation Description: Another message from user group two

Figure 4.3 Spinal Cord Grey Matter Term & Annotation

In this example, navigation query command is "//term[annotation]" since there is only one annotated Gene Ontology in GAS Blog by now. In the future, users can use more specific and complicated navigation command to narrow down the result.

After clicking search button in Figure 4.2, a redirection web page shows and displays all categories with matched Gene Ontology term or annotation information and users can click one to get navigation result in that category to make the result more accurate and relevant as well as decrease un-useful results amount. On the other hand, beside the category name, it also indicates the number of results in that category. This redirection web page is displayed in Figure 4.4 below.


LU Gene Annotation Semantic Blog

Wei Yuan's Thesis

Matched term in category:

- Adult Mouse Anatomy (1)

Categories

Anatomy
| Adult Mouse Anatomy
| Amphibian Anatomy
| Cereal Anatomy
| Dictyostelium Discoideum Anatomy
| Fly Anatomy

Figure 4.4 Category Redirection Page

If users are willing to view the navigation result in category "Adult Mouse Anatomy", link "Adult Mouse Anatomy" will bring the result to users in format of SCORM Learning Objects manifest as mentioned in Chapter 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest identifier="MANIFEST_IDENTIFIER" version="1.0"
xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_v1p3"
xmlns:adlnav="http://www.adlnet.org/xsd/adlnav_v1p3"
xmlns:adlseq="http://www.adlnet.org/xsd/adlseq_v1p3"
xmlns:imsss="http://www.imsglobal.org/xsd/imsss"
xmlns:lom="http://ltsc.ieee.org/xsd/LOM"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1
imscp_v1p1.xsd
http://www.adlnet.org/xsd/adlcp_v1p3 adlcp_v1p3.xsd
http://www.adlnet.org/xsd/adlnav_v1p3 adlnav_v1p3.xsd
http://www.adlnet.org/xsd/adlseq_v1p3 adlseq_v1p3.xsd
http://www.imsglobal.org/xsd/imsss imsss_v1p0.xsd
http://ltsc.ieee.org/xsd/LOM lom.xsd"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <metadata>
    <schema>ADL SCORM</schema>
    <schemaversion>CAM 1.3</schemaversion>
  </metadata>
  <organizations default="ORG-ANATOMY-ADULT MOUSE ANATOMY">
```

```
    <organization identifier="ORG-ANATOMY-ADULT_MOUSE_ANATOMY"
structure="hierarchical">
      <title>Activity Tree</title>
      <item identifier="ACT-MA:0000002" identifierref="RES-
MA:0000002">
        <title>Spinal Cord Grey Matter</title>
      </item>
    </organization>
  </organizations>
  <resources>
    <resource identifier="RES-MA:0000002" type="webcontent"
adlcp:scormType="sco" href="http://localhost/wordpress/wp-
includes/single.php?category=/anatomy/adult_mouse_anatomy&#38;ID=MA:0
000002">
      <metadata>
        <lom:lom>
          <lom:general>
            <lom:title>
              <lom:string language="en">Spinal Cord Grey
Matter</lom:string>
            </lom:title>
            <lom:language>en</lom:language>
            <lom:description>
              <lom:string language="en"></lom:string>
            </lom:description>
            <lom:keyword>
              <lom:string>an</lom:string>
              <lom:string>annotation</lom:string>
              <lom:string>group</lom:string>
            </lom:keyword>
            <lom:structure>
              <lom:source>LOMv1.0</lom:source>
              <lom:value>hierarchical</lom:value>
            </lom:structure>
            <lom:annotation>annotation letter sequence should be
here</lom:annotation>
          </lom:general>
        </lom:lom>
      </metadata>
    </resource>
  </resources>
</manifest>
```

4.2.3 Generic Navigation

By contrast to Collaborative Navigation, Generic Navigation is a cross-group search. It allows users to search Gene Ontologies and their annotation based on a specific navigation condition all over GAS Blog under the guide and assistance of FOAF feature of GAS Blog.

4.2.3.1 Generic Navigation Usage Scenario

To use Generic Navigation, users do not have to login as Collaborative Navigation. There is a link on the front page of GAS Blog, and it will lead users to the navigation form as Figure 4.1 after clicking on it.

Here is a realistic Generic Navigation usage scenario. To show the difference between Collaborative Navigation and Generic Navigation, this example is based on same Gene Ontology and its annotation as the example in section 4.2.2.1.

As Figure 4.3 displays, gene ontology for Spinal Cord Grey Matter with ID "MA: 0000002" has two annotations, one from Group 1, and the other from Group 2. And the navigation command is the same as the example of Collaborative Navigation – "//term[annotation]".

After clicking search button in Figure 4.2, the same redirection page shows and displays all categories with matched Gene Ontology term or annotation information and the result amount under each category as Figure 4.4.

The following result only show the different part of output SCORM manifest navigation result and the other part is the same with example usage scenario in section of Collaborative Navigation.

```
... ...
<lom:general>
  <lom:title>
    <lom:string language="en">Spinal Cord Grey Matter</lom:string>
  </lom:title>
  <lom:language>en</lom:language>
  <lom:description>
    <lom:string language="en"></lom:string>
  </lom:description>
  <lom:keyword>
    <lom:string>an</lom:string>
    <lom:string>annotation</lom:string>
    <lom:string>group</lom:string>
    <lom:string>message</lom:string>
    <lom:string>user</lom:string>
    <lom:string>two</lom:string>
  </lom:keyword>
  <lom:structure>
    <lom:source>LOMv1.0</lom:source>
    <lom:value>hierarchical</lom:value>
  </lom:structure>
  <lom:annotation>annotation letter sequence should be
here</lom:annotation>
  <lom:annotation>the second annotation letter
sequence</lom:annotation>
</lom:general>
... ...
```

4.3 Gene Ontology Navigation Performance Evaluation

Gene Ontology Navigation is based on XPath query mechanism. To identify the advantages of adding semantics in the querying process, Generic Navigation and Collaborative Navigation is evaluated by comparing it to traditional XPath querying that is based on non-semantics. Generic Navigation and Collaborative Navigation are evaluated from following aspects:

- Number of Comparison: the amount of gene ontologies utilized using a given query command in order to find the matched results;
- Number of Result Returned: the number of returned navigation results;
- Effect of Annotations on Navigation: number of entries that are known to relate to annotated ontologies.

In our evaluation, the dataset for the Gene Ontology used include 28316 entries for all classification categories. In addition, there are two groups who attempted to annotate these entries where these are ten annotation made by each group. Table 4.3 lists the status of our initial dataset.

Total Ontology Amount	28316
Gene Ontologies with Annotation from User Group 1 in Category Adult Mouse Anatomy	5
Gene Ontologies with Annotation from User Group 1 in Category Fly Anatomy	5
Gene Ontologies with Annotation from User Group 2 in Category Fly Anatomy	5
Gene Ontologies with Annotation from User Group 2 in Category Adult Mouse Anatomy	5

Table 4.3 Gene Ontology Dataset

4.3.1 Number of Comparison

Navigate Available Gene Ontologies (No Condition)

For XPath without semantics, it goes through all available Gene Ontologies to match the navigation condition. Generic Navigation is aided by Gene Ontologies categories, therefore, it only goes through certain categories of Gene Ontologies and match the navigation condition. Collaborative Navigation only searches Gene Ontologies inside a certain group and also directed by Gene Ontology categories, therefore, Collaborative Navigation is expected to have the least number of comparison.

In this evaluation case, Generic Navigation and Collaborative Navigation is directed into category of Adult Mouse Anatomy, and user group of Collaborative Navigation is "Group 1". Navigation command used in this evaluation case is "//term".

As displayed in Figure 4.5, XPath goes through all 28316 gene ontologies while the Generic Navigation walks through only 2748 entries of Gene Ontologies in the category of "Adult Mouse Anatomy" and the Collaborative goes directly to 5 Gene Ontologies that are found to be annotated by users in same group.

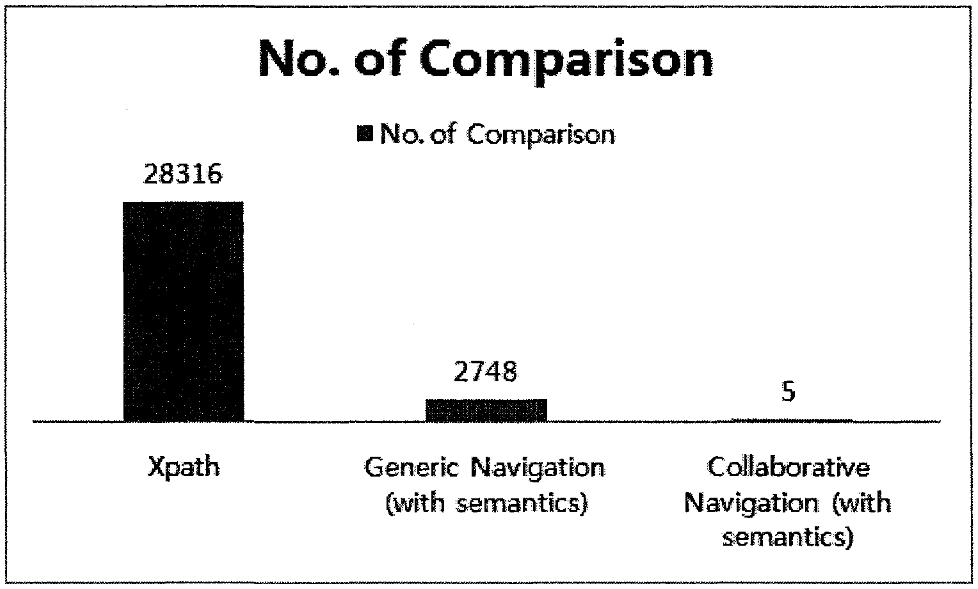


Figure 4.5 No. of Comparison (None Condition)

Moreover, we repeated the evaluation experiment with more complicated queries where we introduce more components or conditions at the query parts. Figures illustrate our findings.

Navigate Gene Ontologies whose name contains "neck" (One Condition)

In this evaluation case, Generic Navigation and Collaborative Navigation is directed into category of Fly Anatomy, and user group of Collaborative Navigation is "Group 1". Navigation command used in this evaluation case is "//term[contains(name,"neck")]".

As displayed in Figure 4.6, XPath goes through all 28316 Gene Ontologies while the Generic Navigation walks through only 6222 entries of Gene Ontologies in the category of "Fly Anatomy" and the Collaborative goes directly to 5 Gene Ontologies that are found to be annotated by users in same group.

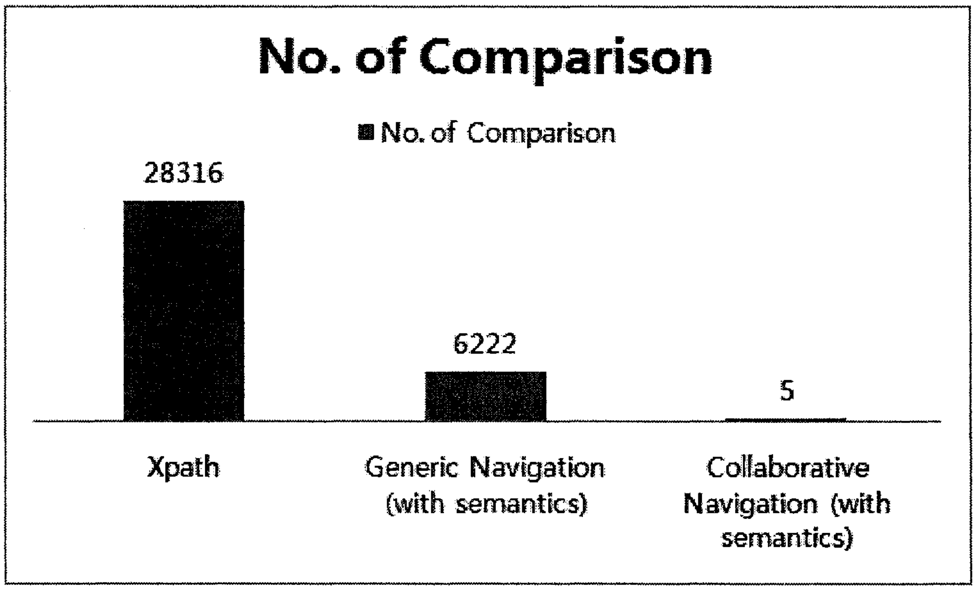


Figure 4.6 No. of Comparison (One Condition)

Comparing this evaluation case to last one, number of comparison for traditional XPath is fixed, it does not change according to navigation command. On the other hand, for Gene Ontology Navigation, number of comparison is dynamical, it changes according to guidance of metadata (Gene Ontology category).

Navigate Gene Ontologies whose name contains "neck" and id contains "MA" (Two Condition)

In this evaluation case, Generic Navigation and Collaborative Navigation is directed into category of Adult Mouse Anatomy, and user group of Collaborative Navigation is "Group 1". Navigation command used in this evaluation case is "//term[contains(name,"neck")][contains(id,"MA")]".

As displayed in Figure 4.7, XPath goes through all 28316 Gene Ontologies while the Generic Navigation walks through only 2748 entries of Gene Ontologies in the category of "Adult Mouse Anatomy" and the Collaborative goes directly to 5 Gene Ontologies that are found to be annotated by users in same group.

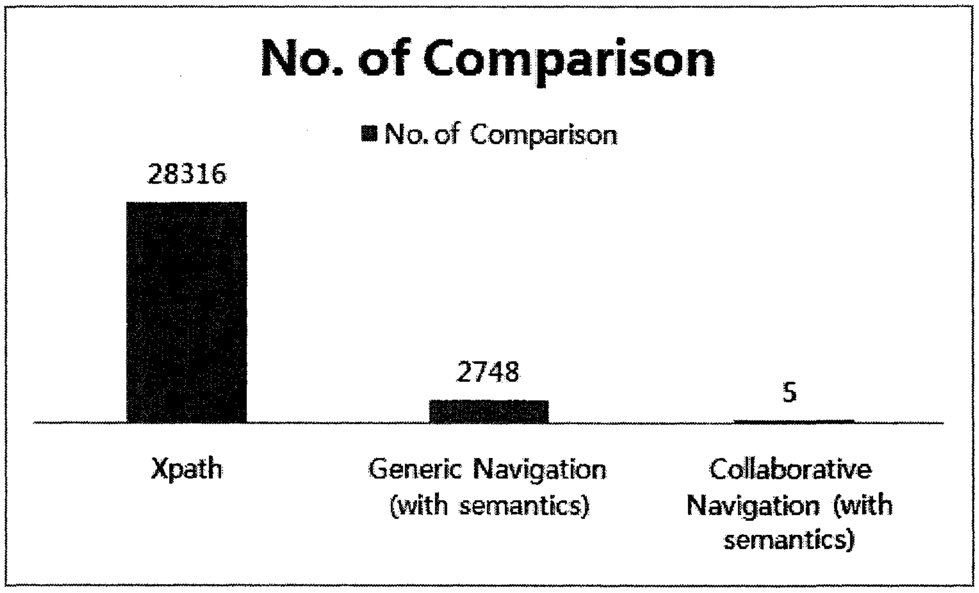


Figure 4.7 No. of Comparison (Two Conditions)

This evaluation case further illustrates the relationship between Gene Ontology category and number of comparison for Gene Ontology Navigation. Number of comparison for Gene Ontology Navigation is decided by Gene Ontology categories. The number of comparison for Gene Ontology Navigation is the number of entries in that category; it will not change by modifying the navigation command when Gene Ontology is the same. For traditional XPath, number of comparison is fixed all the time, and always the number of available entries of Gene Ontology from all categories.

From these figures, we found that by adding semantics we can reduce the number of comparison and save much of the navigation time even with the availability of annotation.

4.3.2 Number of Result Returned

Navigate Gene Ontologies whose name contains "neck" (One Condition)

We set our initial evaluation by navigating Gene Ontologies with an entry containing "neck". Same as last evaluation, Generic Navigation is guided by category of "Adult Mouse Anatomy", and user group of Collaborative Navigation is "Group 1". The navigation command used in this case is "//term[contains(name,"neck")]".

The result of this evaluation case is displayed in Figure 4.8. Traditional XPath found 30 results among Gene Ontologies from all categories while Generic Navigation returned 20 results after guided by category "Adult Mouse Anatomy", and Collaborative Navigation returned 2 results under guidance of category "Adult Mouse Anatomy" and user group "Group 1".

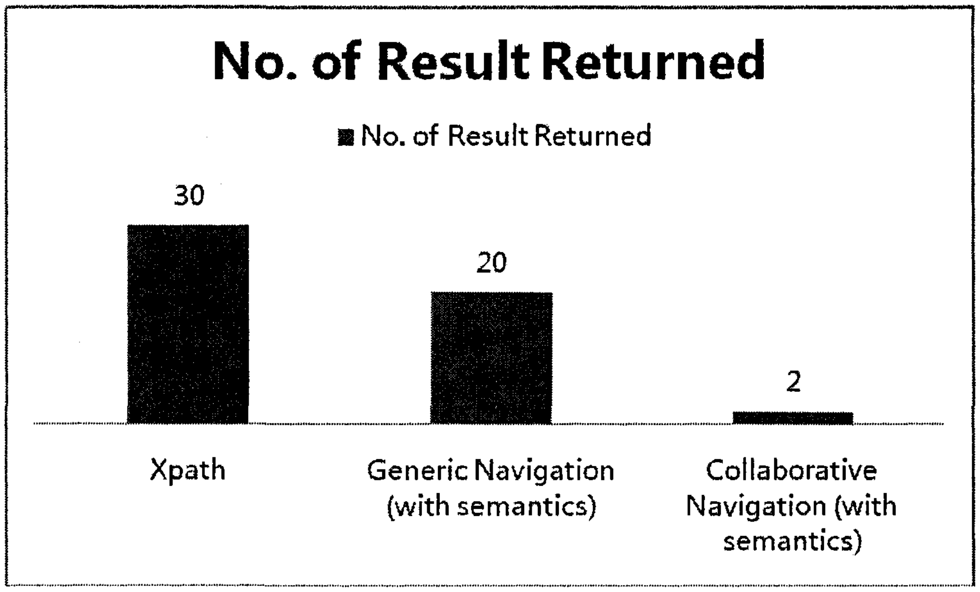


Figure 4.8 No. of Result Returned (One Condition)

This evaluation case is built on an assumption that the users who start the navigation are interested in Gene Ontology category "Adult Mouse Anatomy" and are from user group "Group 1". From the result of this evaluation case, XPath contains most un-useful information for users who only have interest in Gene Ontology of category "Adult Mouse Anatomy"; Generic Navigation narrows down the result to the specific category; Collaborative Navigation directly returns Gene Ontology which annotated by all user group members from "Group 1". Therefore, for users who collaborate with others in a single Gene Ontology field, Collaborative Navigation is most relevant and Generic Navigation helps users to browse Gene Ontology information from all user groups.

Navigate Gene Ontologies whose name contains "neck" and id contains "MA" (Two Conditions)

This evaluation case is to navigate Gene Ontologies whose name contains "neck" and id contains "MA". In this evaluation case, Generic Navigation is guided with category "Adult Mouse Anatomy" and Collaborative Navigation is directed by the same category and user group "Group 1". Navigation command used in this evaluation case is "//term[contains(name,"neck")][contains(id,"MA")]".

The result of this evaluation case is displayed in Figure 4.9. Navigation with XPath found 23 results in all categories, Generic Navigation directed by gene ontology category returned 20 results, and Collaborative Navigation guided by Gene Ontology category and user group returned 2 results.

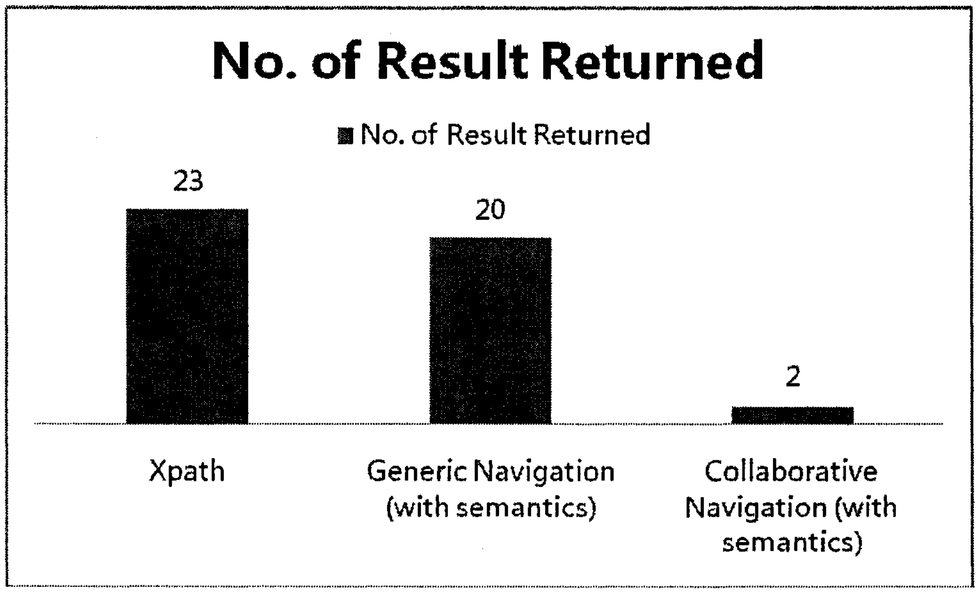


Figure 4.9 No. of Result Returned (Two Conditions)

Similarly as last evaluation case, more conditions in navigation command can decrease the amount of redundant information; however, there are still some unnecessary pieces of Gene Ontologies returned by XPath. This problem of XPath might not seem to be huge in the evaluation cases due to limited Gene Ontologies available at this time. However, there are millions of Gene Ontologies available in real life, and irrelevant information returned by XPath can take a lot time of users to filter them manually.

Navigate Gene Ontologies whose name contains "neck" and id contains "MA" and is part of another Gene Ontology (Three Conditions)

This evaluation case is to navigate Gene Ontologies whose name contains "neck" and id contains "MA" and is part of another Gene Ontology. In this evaluation case, Generic Navigation is guided with category "Adult Mouse Anatomy" and Collaborative Navigation is directed by the same category and user group "Group 1". Navigation command used in this evaluation case is "//term[contains(name,"neck")][contains(id,"MA")][contains(relationship/type,"part")]".

The result of this evaluation case is displayed in Figure 4.10. Navigation with XPath found 20 results in all categories, Generic Navigation directed by Gene Ontology category returned 17 results, and Collaborative Navigation guided by Gene Ontology category and user group returned one result.

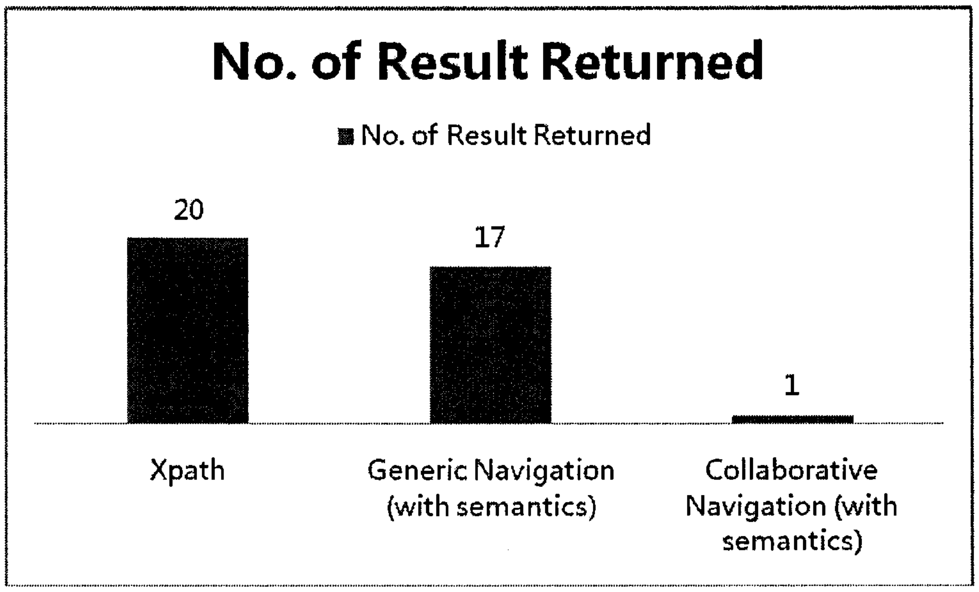


Figure 4.10 No. of Result Returned (Three Conditions)

From these figures, we found that by adding semantics we can reduce the number of result returned and save much of the navigation time for users to navigate to the Gene Ontologies that they are looking for.

4.3.3 Effect of Annotation on Navigation

Navigate Annotated Gene Ontologies

To analyse the effect on navigation when annotation is concerned, we dedicate this section for this purpose. As described in Chapter 3, Gene Ontology annotation information is stored separately from the original Gene Ontology data. Hence we expect that the traditional XPath does not find them as there is no support for the XPointer to reference external XML fragment and query through them. Under this circumstance, XPath returned nothing in this evaluation case. However, Generic Navigation and Collaborative Navigation are designed to support XPointer and connect annotation with Gene Ontologies. Thus, based on these semantics we can find entries even if they were as annotations.

We stated one evaluation case where the Generic Navigation is guided with category "Adult Mouse Anatomy" and Collaborative Navigation is directed by the same category and user group "Group 1". Navigation command used in this evaluation case is "//term[annotation]".

There are 10 entries of annotation from user group "Group 1" and "Group 2" in current GAS Blog system. Generic Navigation returns all of them as navigation result and Collaborative Navigation returns 5 results since they are annotated by users from user group "Group 1". Evaluation result is displayed in the chart below.

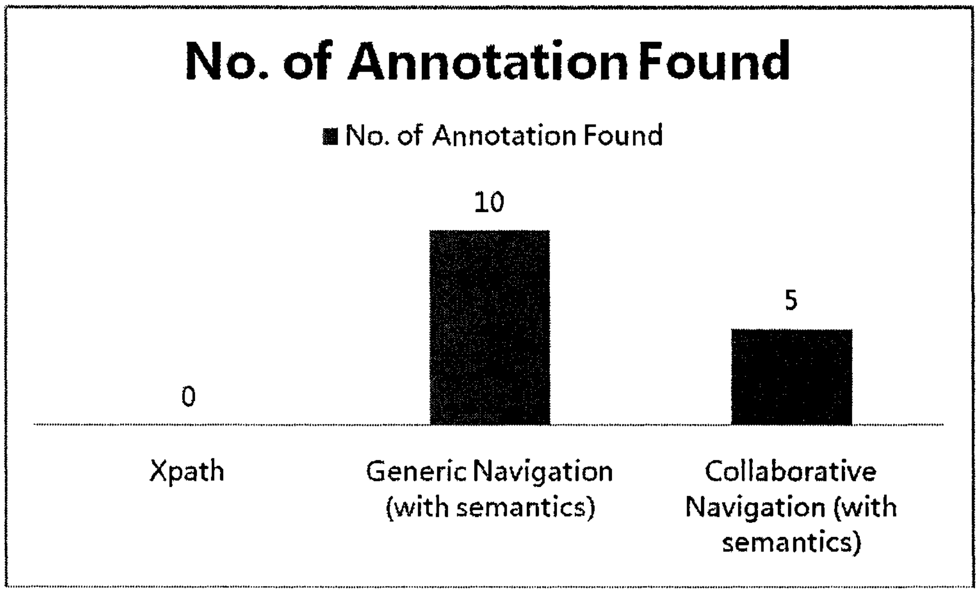


Figure 4.11 No. of Annotation Found (One Condition)

According to the result of this evaluation case, XPath is not able to query Gene Ontology annotation if they are not stored in the same file with Gene Ontologies in data management system. On the other hand, Generic Navigation and Collaborative Navigation can connect them together as they are stored in the location and complete the navigation task with relevant navigation results.

Navigate Gene Ontology Annotation with Content Contains "annotation"

This evaluation case is to navigate Gene Ontology whose annotation content contains "annotation". Similar to the former evaluation case, the Generic Navigation is guided with category "Adult Mouse Anatomy" and Collaborative Navigation is directed by the same category and user group "Group 1". Navigation command used in this evaluation case is "//term[annotation][contains(annotation/description,"annotation")]".

The result of this evaluation case is in Figure 4.12. Same reason as last evaluation case, traditional XPath can not return any result about annotation, and adding more conditions concerning to annotation does not do much help to that. As for Gene Ontology Navigation, more conditions about annotation in navigation command narrows down the annotation returned as expected. In this evaluation case, there are 4 annotation entries containing "annotation" in their content in both user group "Group 1" and "Group 2", and Generic Navigation returned them. Collaborative Navigation is directed by user group "Group 1", and Collaborative Navigation finds 2 annotation entries containing "annotation" in content in this group.

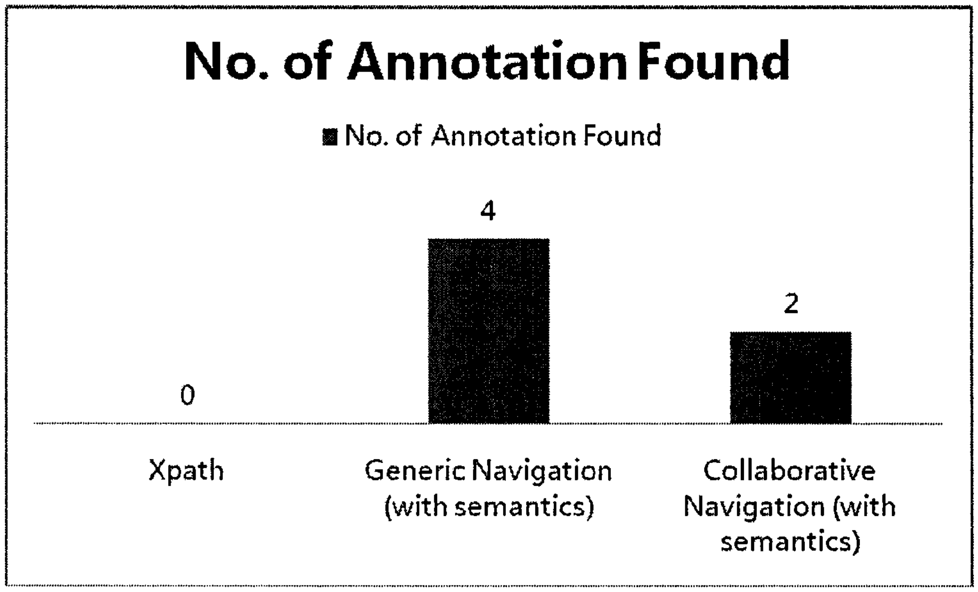


Figure 4.12 No. of Annotation Found (Two Condition)

From these figures above, we can conclude that traditional XPath can not query information concerning annotation, and the incapability of querying annotation for traditional XPath is not caused by navigation command. Generic Navigation and Collaborative Navigation is able to query annotation information as expected. The number of annotation found by Generic Navigation and Collaborative Navigation can be reduced and more accurate by adding conditions to navigation command.

It is unnecessary to keep evaluating Gene Ontology Navigation repeatedly by adding more conditions to navigation command because no matter how many conditions in navigation command, traditional XPath always return nothing and the number of annotation entries found by Gene Ontology Navigation would keep reducing as the navigation command become more complicated. The current evaluation cases already show the trend of that.

4.4 Summery

The core theory of Gene Ontology Navigation in GAS Blog is the use of metadata and they can accommodate annotations. Our metadata utilize a dictionary. Dictionary makes Gene Ontology Navigation engine more semantical during the process of extracting metadata and looking for synonyms for metadata. In this way, even users input different word in navigation command; GAS still can find semantical relevant information instead of spelling relevant one.

From the evaluation of Gene Ontology Navigation, Generic Navigation and Collaborative Navigation shows better overall performance comparing to traditional XPath query. After comparing to traditional XPath querying, Gene Ontology Navigation match the navigation command with less comparison objects to increase the speed and efficiency of querying process; smaller amount but more relevant navigation results returned; and better ability on navigating gene ontology annotations.

However, GAS is not a complete solution yet; the future work related to this feature will be addressed in next chapter.

Chapter Five
Conclusions & Future Research

This thesis intends to introduce a new gene ontology annotation system (GAS) based on Web 2.0 technologies with extended semantics capabilities that includes Gene Ontology semantics, SCORM semantics, FOAF semantics and RSS syndication and aggregation semantics.

Chapter 1 introduced Web 2.0 with popular applications over Internet and the essential technologies for Web 2.0. On the other hand, it also talked about e-learning objects on Web 2.0 as well as the features of e-learning objects on Web 2.0.

Chapter 2 described gene ontology annotation and analysed existing gene ontology annotation systems powered by traditional technologies. Also, Chapter 2 addressed the disadvantages for current gene ontology annotation systems, and the necessity of creating a new gene ontology annotation system as well as navigation system based on the new generation of web technology – Web 2.0.

Chapter 3 addressed the development requirements and design policies of our new GAS Blog based on the features GAS system that overcomes the shortcomings of the traditional annotation and navigation systems. Based on these requirements, our proposed design is based on the new Web 2.0 programming environment and tools such as WordPress and PHP. After deciding development language, platform and learning object metadata standard, semantic features of GAS Blog is introduced in details in this chapter including what semantic features GAS Blog have, where and how they are published, GAS Blog usage scenario, and implementation of each modules with pseudo code and UML graphs.

Chapter 4 described GAS usage in navigation using generic and collaborative semantics. This chapter provide some proof on the GAS affectivity. Gene ontology navigation is a new generation of gene ontology querying service based on web 2.0 and semantic web technology. Chapter 4 starts from introducing what web 2.0 and semantic web navigation is and describing Collaborative Navigation and Generic Navigation respectively. After introducing Collaborative Navigation and Generic Navigation, they are evaluated with some scenarios by comparing to traditional XPath querying. Current evaluation result shows that Gene Ontology Navigation is an improvement to traditional XPath querying service.

5.1 Conclusion

This thesis builds Gene Ontology annotation and navigation system (GAS). The following are the achievements and findings of this thesis.

1. GAS provides a user friendly interface to allow users to publish newly found gene ontologies or annotating to existing gene ontologies on web pages;

2. GAS creates a online collaborative system for users to cooperate with each other on their work as a social network with capabilities to add other users as friend, creating user groups, join existing user groups, viewing graphical FOAF network information, etc.

3. GAS allows users to syndicate gene ontologies and their annotation information from the gene ontology categories they interested in through RSS to their aggregators.

4. GAS includes a navigation system for allowing users navigate through gene ontologies and annotation efficiently and accurately.

5. Gene Ontology and annotation connector is invented for GAS system to reference external XPoiter XML fragment during navigation process.

6. Automatic metadata extractor is created with support of dictionary WordNet to add metadata to gene ontology annotations to decrease the chances that metadata added manually by users might not match the real content.

7. Gene ontology navigation result is wrapped in SCORM manifest or easier delivery over network and convenient to import into e-learning system based on SCORM.

8. Evaluation of semantic-directed Gene Ontology Navigation is provided in this thesis, and shows that it is an improvement to traditional XPath querying service from the aspects of number of comparison, number of records returned, and the effects of annotations on navigation.

5.2 Future Research

Due to the limited time, some work is left over for the future research.

- Syndication and aggregation of GAS Support System is in RSS standard. Atom feed standard can be supported by GAS Support System for users who prefer Atom feed standard on feed syndication.

- Metadata Integration is not a fully function module yet. GAS Support System should recognize the keyword in navigation command and not only match it with gene ontology and annotation data but also match its synonyms and return the navigation result.

- Navigation result is only wrapped in SCORM manifest standard in GAS Support System at this time. However, it should be extended to wrap the navigation result in some other popular e-learning standard such as CanCore.

- Security issue is not considered yet in current GAS Support System, and it should be concerned in the future research.

- Error handling is only applied to annotation evidence code in current GAS Support System; however, it should be extended to correct the typo word in annotation description automatically with support of dictionary.

- Metadata Extraction can be improved by improving the algorithm to ensure that metadata extracted dynamically can be more meaningful to both machine and human.

For a long-term development of GAS Support System, there are some suggestions might also improve the system in the future at a lower priority as future work mentioned above.

- Extending our GAS to include a reasoned that takes into account various type of semantics introduced as well as some rules that can be helpful in Gene navigation (Mohammed & Fiaidhi, 2008).

- GAS Support System may enrich the FOAF on grouping by generating a group-wise sub-system automatically when group members donate any knowledge to the main system. The user interface can looks like the main GAS Support System, however, this sub-system only publish the Gene Ontologies or annotation published by group members. This could help users with more convenient group collaboration.

- GAS Support System can also have the feature of aggregating information from users' blog on other system and reference the information on GAS. Currently, GAS Support System allows users to aggregate information; however, GAS Support System can also be a user to other support systems by aggregating information from other systems through syndication feed and reference them on GAS Support System automatically. It collects information from other systems and saves users time to visit several systems.

- GAS Support System can have a collaboration log for each user group to show the research / collaboration action each group members did with timestamp. So that a group member will be able to trace the idea evolution and history, which in turn benefits the future research (Yao J. , Supporting Research with Weblogs: A Study on Web-based Research Support Systems, 2006).

- GAS Support System may have accessing authority on each annotation. So that users can grant viewing permits to everybody or only users from the same group as annotators (Yao J. , Supporting Research with Weblogs: A Study on Web-based Research Support Systems, 2006).

- GAS Support System should allow users to report un-useful comments to spam or set up a black list for every registered user. Annotation or comments from users on back list can be hidden for them (Yao J. , Supporting Research with Weblogs: A Study on Web-based Research Support Systems, 2006).

- GAS Support System can have reference retrieve feature in the future. Sometime, users add comments by reference a hyperlink, and GAS system should retrieve a preview of referenced hyperlink so that other users can decide whether browse the hyperlink or not after reading the preview (Yao J. , Supporting Research with Weblogs: A Study on Web-based Research Support Systems, 2006).

- GAS Support System may develop a desktop application for users to publish gene ontologies or annotation without using browser and operate on web pages. The desktop application can also deal with semantics on users' local computer before transferring data to server. It reduces the task of GAS system server and release more server resource other users. (Möller, Breslin, & Decker, 2005)

- GAS Suport System can become more powerful by using multiple gene ontology engines for different formats of gene ontology and annotation file (Shakya, 2006).

- GAS Support System can be extended to share gene ontology and annotation and allow users to collaborate in a decentralized or peer to peer network (Fiaidhi & Mohammed, 2005)

References

Anderson, P. (2007). What is Web 2.0? Ideas, Technologies and Implications for Education. JISC Technology and Standards Watch.

Berglund, A., Boag, S., Chamberlin, D., Fernández, M. F., Kay, M., Robie, J., et al. (2007, January 23). XML Path Language (XPath) 2.0. Retrieved June 9, 2008, from W3C: http://www.w3.org/TR/xpath20/

Bergsten, H. (2003). JavaServer Pages (3rd Edition ed.).

Brust, M. R., & Rothkugel, S. (n.d.) On Anomalies in Annotation Systems. Luxembourg: Faculty of Science, Technology and Communication (FSTC), University of Luxembourg.

Collin, J., & Wang, H. J. (2007). Subspace: Secure CrossDomain Communication for Web mashups. International World Wide Web Conference Committee (IW3C2).

Davis, J. R., & Huttenlocher, D. P. (n.d.) Shared Annotation for Cooperative Learning. Xerox Corporation, Design Research Institute; Computer Science Department, Cornell University.

DeRose, S., Maler, E., & Daniel, R. J. (2002, December 19). XPointer xpointer() Scheme. Retrieved June 9, 2008, from W3C: http://www.w3.org/TR/xptr-xpointer/

Dornfest, R. (2000, August 25). Writing RSS 1.0. Retrieved June 23, 2008, from O'Reilly Network: http://www.oreillynet.com/pub/a/network/2000/08/25/magazine/rss_tut.html

Downes, S. (2003, May 10). RDF Site Summary 1.0 Modules: Learning Object Metadata. Retrieved January 22, 2008, from Stephen's Web: http://www.downes.ca/xml/RSS_LOM.htm

Ernst, J. (2005, September 27). Embedding FOAF information in RSS. Retrieved January 22, 2008, from Johannes Ernst's Blog: http://netmesh.info/jernst/Big_Picture/foaf-in-rss.html

Ernst, J. (n.d.). RSS-FOAF. Retrieved January 22, 2008, from rss-extensions.org: http://rss-extensions.org/wiki/RSS-FOAF

Fiaidhi, J., & Mohammed, S. (2005). Developing a Collaborative Virtual Learning Model for Peer to Peer Grid Environment. IEEE Learning Technology Journal , 7 (1), 49-53.

Fiaidhi, J., Passi, K., & Mohammed, S. (2004). Developing a Framework for Learning Objects Search Engine. The 2004 International Conference on Internet Computing (IC04), (pp. 610-616). Las Vegas.

Fiaidhi, J., Mohammed, S., JAAM, J., & HASNAH, A. (2003). A Standard Framework for Search Hosting via Ontology Based Query Expansion. Pakistan Journal of Information and Technology , 66-70.

Friesen, N. (2004, October 22). Frequently asked Questions. Retrieved June 19, 2008, from CanCore: http://www.cancore.ca/en/faq.html

GeneOntology.org. (2007, December 4). GO Annotation Guide. Retrieved June 21, 2008, from the Gene Ontology: http://www.geneontology.org/GO.annotation.shtml

Glass, G. (n.d.). Marginalia Web Annotation. Retrieved october 26, 2007, from Geof: http://www.geof.net/code/annotation

Guide to GO Evidence Codes. (n.d.). Retrieved June 1, 2008, from the Gene Ontology : http://www.geneontology.org/GO.evidence.shtml

Harrsch, M. (2003). RSS: The Next Killer App For Education. Illinois Computing Educators Computer Update Bulletin for Educators , 2003 (4), 10.

Heck, R. M., Luebk, S. M., & Obermark, C. H. (1999). Retrieved October 25, 2007, from http://www.math.grin.edu/~rebelsky/Blazers/Annotations/Summer1999/Papers/survey_paper.html

Keller, C. (2005). 7 Things You Should Know About Wikis.

Koivunen, M.-R. (2005, October 31). Annotea project. Retrieved October 28, 2007, from World Wide Web Consortium: http://www.w3.org/2001/Annotea/

Lewin, J. (2003, December 23). Content feeds with RSS 2.0. Retrieved June 23, 2008, from IBM: http://www.ibm.com/developerworks/xml/library/x-rss20/

Liu, H., Hu, Z., & Wu, C. (2005). A tool for navigation and visualization of Gene Ontology resources. Retrieved July 16, 2008, from BMC Bioinformatics, BioMed Central Ltd: http://gauss.dbb.georgetown.edu/liblab/dyngo.html

Meng, P. (2005). Podcasting & Vodcasting. University of Missouri.

Miller, G. A., Fellbaum, C., Tengi, R., Wakefield, P., Langone, H., & Haskell, B. R. (n.d.). About WordNet. Retrieved August 4, 2008, from WordNet: http://wordnet.princeton.edu/

Möller, K., Breslin, J., & Decker, S. (2005). semiBlog – Semantic Publishing of Desktop Data. National University of Ireland, DERI, Galway, Republic of Ireland.

Mohammed, S., & Fiaidhi, J. (2008). Developing an Ontology Extraction Agent for Biomedical Learning Social Network. NASTEC 2008 Conference. Montreal.

Morgan, E. L. (2005). Presentation from pre-conference workshop during the ALA Annual Meeting. Chicago.

Nature, G. (2000). AmiGO Help. Retrieved July 16, 2008, from the Gene Ontology: http://www.geneontology.org/amigo/help-front.shtml

Nature, G. (2000). Gene Ontology: tool for the unification of biology. Retrieved June 10, 2008, from The Gene Ontology Consortium: http://www.geneontology.org/GO.annotation.shtml

O'Reilly, T. (2006). Web 2.0 Compact Definition: Trying Again. Retrieved October 4, 2007

Othman, R. M., Deris, S., & Illias, R. M. (2006). Computational Method for Annotation of Protein Sequence According to Gene Ontology Terms. International Journal of Biomedical Sciences Volumne 1 Number 1 ISSN 1306-1216 , 186.

Pilgrim, M. (2002, December 18). What Is RSS. Retrieved June 9, 2008, from XML: http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html

Röscheisen, M., Christian, M., & Terry, W. (1994). Shared Web Annotations As A Platform for Third-Party Value-Added Information Providers: Architecture, Protocols, and Usage Examples. Technical Report CSDTR/DLTR, Stanford University.

Shakya, A. (2006). A Semantic Blogging Framework for Better Utilization of Information. Asian Institute of Technology, School of Engineering and Technology.

Stuart, A. (2005, January 31). COBrA User Interface. Retrieved July 16, 2008, from XSPAN: http://www.xspan.org/cobra/

Swick, R., Prud'hommeaux, E., Koivunen, M.-R., & Kahan, J. (2002, December 20). Annotea Protocols. Retrieved October 29, 2007, from World Wide Web Consortium: http://www.w3.org/2002/12/AnnoteaProtocol-20021219

Vasudevan, V., & Palmer, M. On Web Annotations: Promises and Pitfalls of Current Web Infrastructure. Object Services and Consulting Inc.

Wiley, D. A. (2000). Learning Object Design and Sequencing Theory. Brigham Young University.

Willison, S. P. (2005). Building a decentralised collaborative annotation system for the World-Wide Web. Department of Computer, University of Bath.

WuXian, ZhangLei, & YuYong. (2005). Exploring Social Annotations for the Semantic Web. International World Web Conference Committee (IW3C2).

Yao, J. (2005). Design of Web-based Support System. University of Regina, Department of Computer Science, Regina.

Yao, J. T., & Yao, Y. Y. (2003). Web-based Support Systems. University of Regina, Department of Computer Science, Regina.

Yee, K.-P. (n.d.). CritLink: Public Web Annotation. Retrieved October 30, 2007, from Ka-Ping Yee: http://zesty.ca/crit/

Yee, K.-P. (2002). CritLink: Advanced Hyperlinks Enable Public Annotation on the Web. Berkeley: University of California.

Yee, K.-P. (2000, January 24). What's a mediator? Retrieved October 30, 2007, from Ka-Ping Yee: http://zesty.ca/mediator.html

APPENDIX A
GAS SYSTEM REQUIREMENT & INSTALLATION

A.1. GAS System Requirement

- Web Server such as Microsoft Internet Information Services (IIS)
- PHP script language (version 4.3 or greater)
- MySQL database (version 4.0 or greater)

A.2. GAS System Installation Instruction

GAS is a web-based annotation system written in PHP. Therefore, a web server and PHP must be running prior to GAS installation.

A.2.1. Installation of Web Server

This section takes Microsoft IIS web server as an example of web server installation. IIS is integrated with Windows operation system already. To activate it, go to "Control Panel" and double click Programs and Features as Figure A.1.

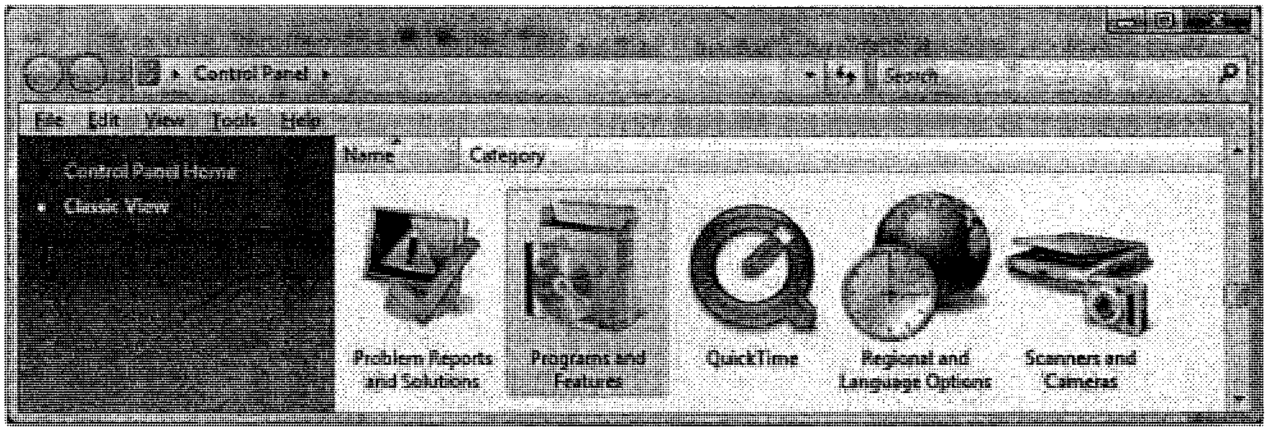
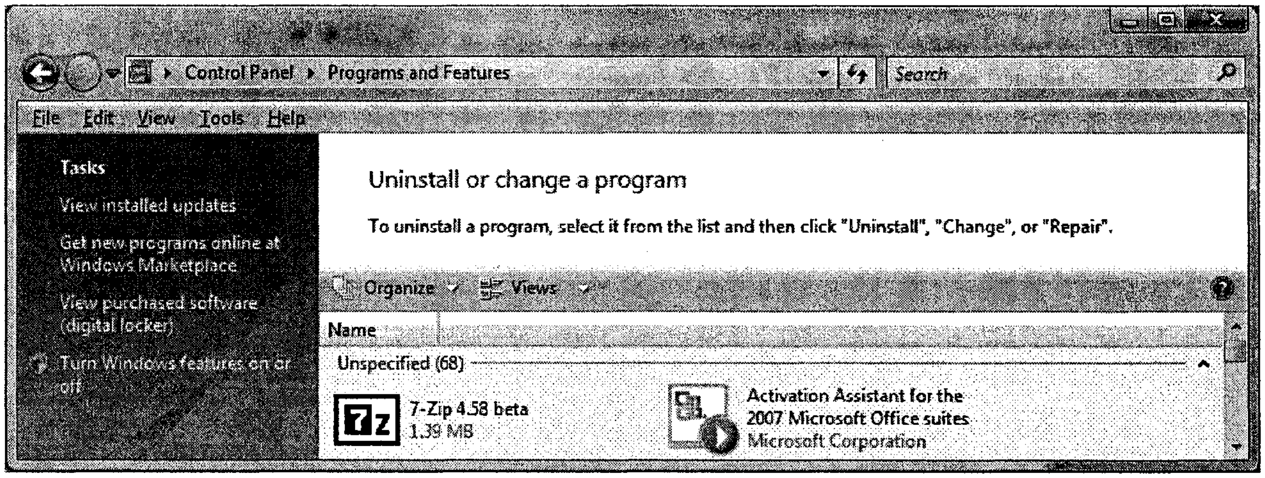


Figure A.1 Programs and Features on Control Panel

On Programs and Features window, click Turn "Windows features on or off" on the left sidebar as Figure A.2.

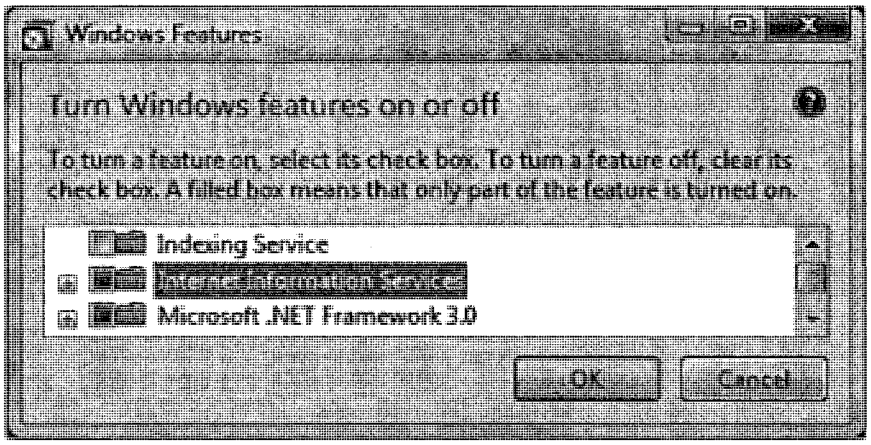

Control Panel ▸ Programs and Features
Search
File Edit View Tools Help
Tasks
View installed updates
Get new programs online at Windows Marketplace
View purchased software (digital locker)
Turn Windows features on or off
Organize Views
Name
Unspecified (68)
7-Zip 4.58 beta
1.39 MB
Activation Assistant for the 2007 Microsoft Office suites
Microsoft Corporation

Uninstall or change a program

To uninstall a program, select it from the list and then click "Uninstall", "Change", or "Repair".

Figure A.2 Programs and Features Window

After clicking "Turn Windows features on or off" on Figure 1.2, "Windows Feature" dialog comes out. In "Windows Features" dialog, expand the list of "Internet Information Services" as highlighted in Figure A.3.



Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

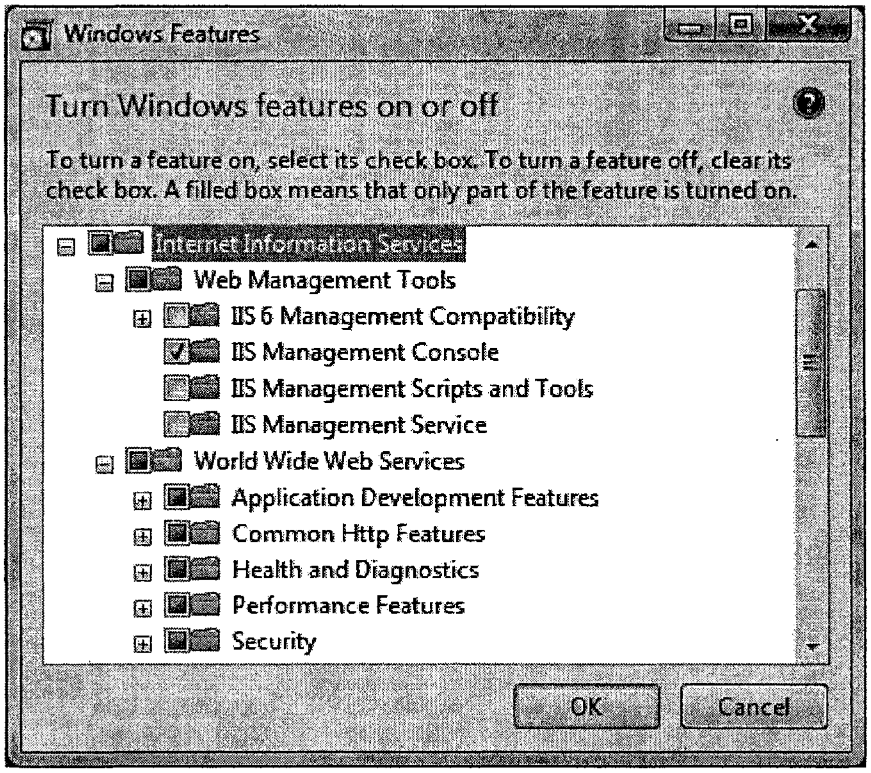
Figure A.3 Windows Features

After expanding the list as Figure 1.4, make sure that the following items under "Internet Information Services" are checked:

- "IIS Management Console" under "Web Management Tool"
- "CGI", "ISAPI Extensions", "ISAPI Filters" under "World Wide Web Services" -> "Application Development Features"
- "Default Document", "Directory Browsing", "Http Errors", "Static Content" under "Common Http Features"
- "Http Logging", "Request Monitor" under "Health and Diagnostics"

- "Static Content Compression" under "Performance Features"
- "Request Filtering" under "Security"

After checking all the items above, click OK button.



Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

```
⊟ ■ Internet Information Services
    ⊟ ■ Web Management Tools
        ⊞ ☐ IIS 6 Management Compatibility
          ✓ IIS Management Console
          ☐ IIS Management Scripts and Tools
          ☐ IIS Management Service
    ⊟ ■ World Wide Web Services
        ⊞ ■ Application Development Features
        ⊞ ■ Common Http Features
        ⊞ ■ Health and Diagnostics
        ⊞ ■ Performance Features
        ⊞ ■ Security
```

Figure A.4 Internet Information Services

Double click "Administrative Tools" on Control Panel as Figure 1.5.

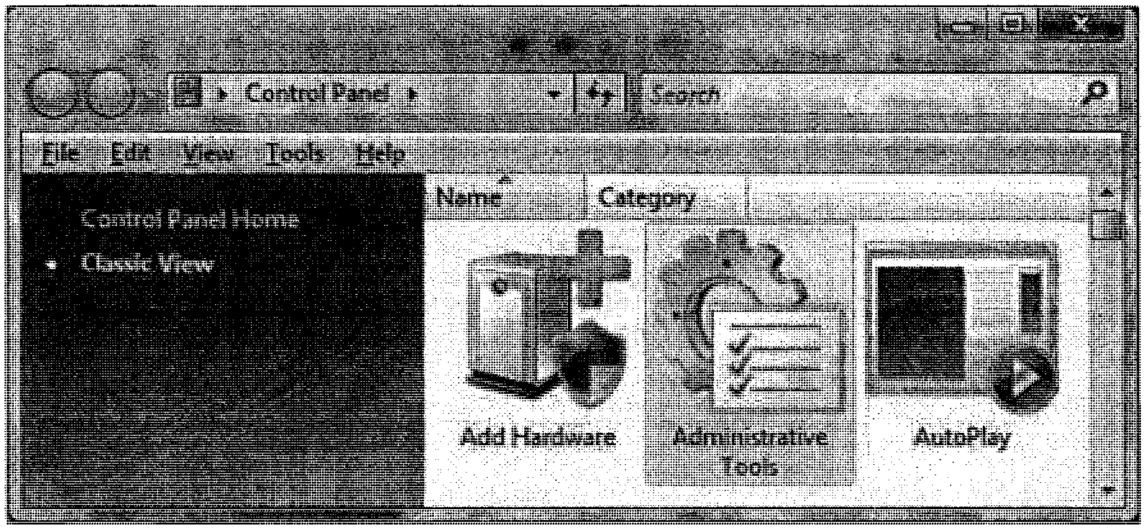
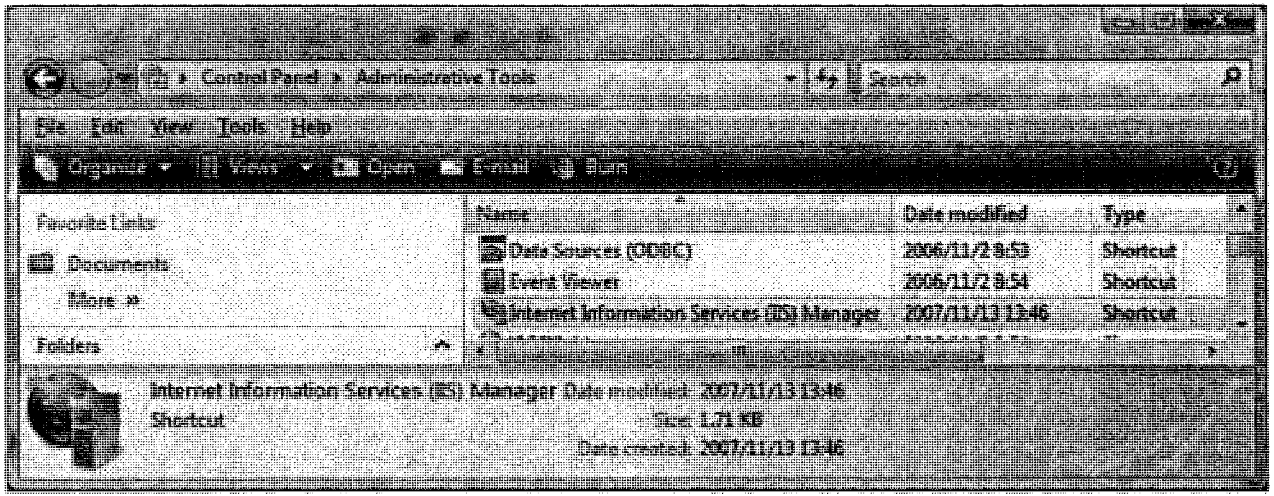


Figure A.5 Administrative Tools in Control panel

Double click “Internet Information Services (IIS) Manager” on “Administrative Tools” window as Figure A.6.


Control Panel ▸ Administrative Tools
Search
File Edit View Tools Help
Organize Views Open E-mail Burn
Favorite Links
Documents
More »
Folders

Name	Date modified	Type
Data Sources (ODBC)	2006/11/2 8:53	Shortcut
Event Viewer	2006/11/2 8:54	Shortcut
Internet Information Services (IIS) Manager	2007/11/13 13:46	Shortcut

Internet Information Services (IIS) Manager Date modified: 2007/11/13 13:46
Shortcut Size: 1.71 KB
Date created: 2007/11/13 13:46

Figure A.6 Administrative Tools

Click “Basic Settings” on right action side bar of “Internet Information Services (IIS) Manager” as Figure A.7.

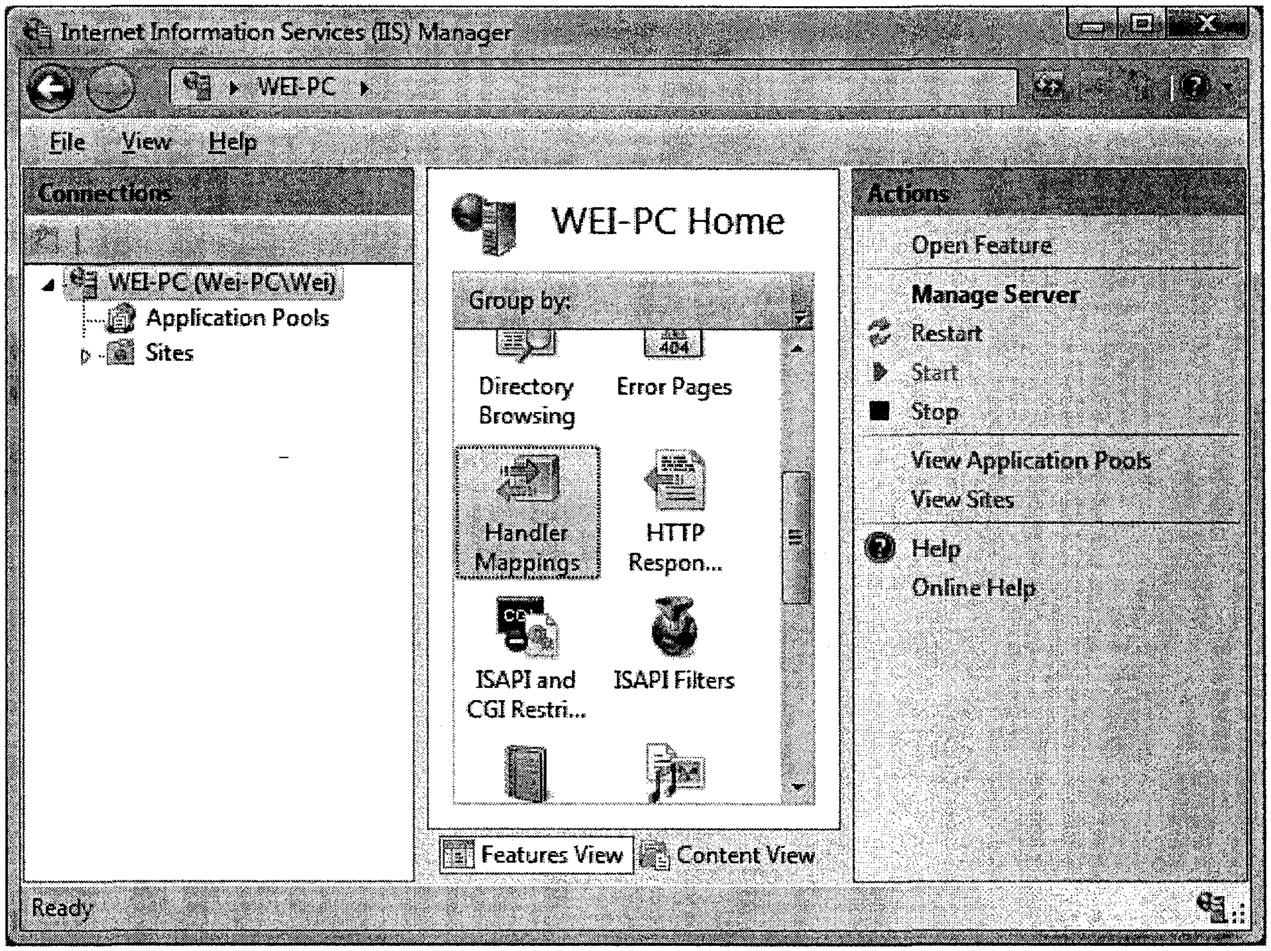
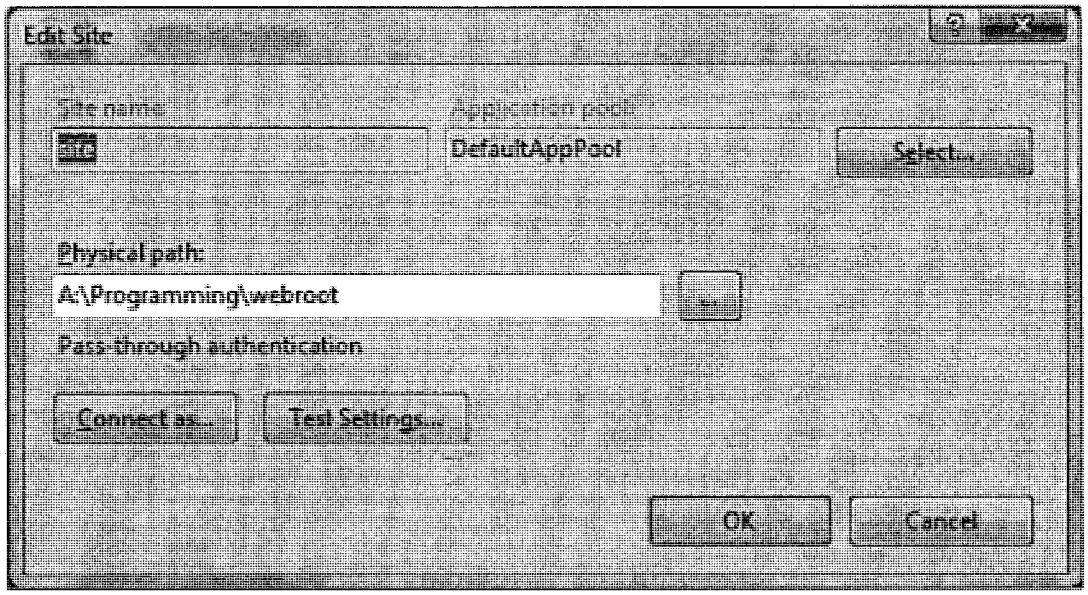


Figure A.7 Internet Information Services (IIS) Manager

“Edit Site” dialog (Figure 1.8) pops out after clicking “Basic Settings” on right action side bar of “Internet Information Services (IIS) Manager” as Figure 1.7. In “Edit Site” dialog, the “Physical path” can be changed to a folder you desired, or use the IIS default web root folder C:\inetpub\wwwroot if windows operation system is installed on drive C.



Site name: site

Application pool: DefaultAppPool

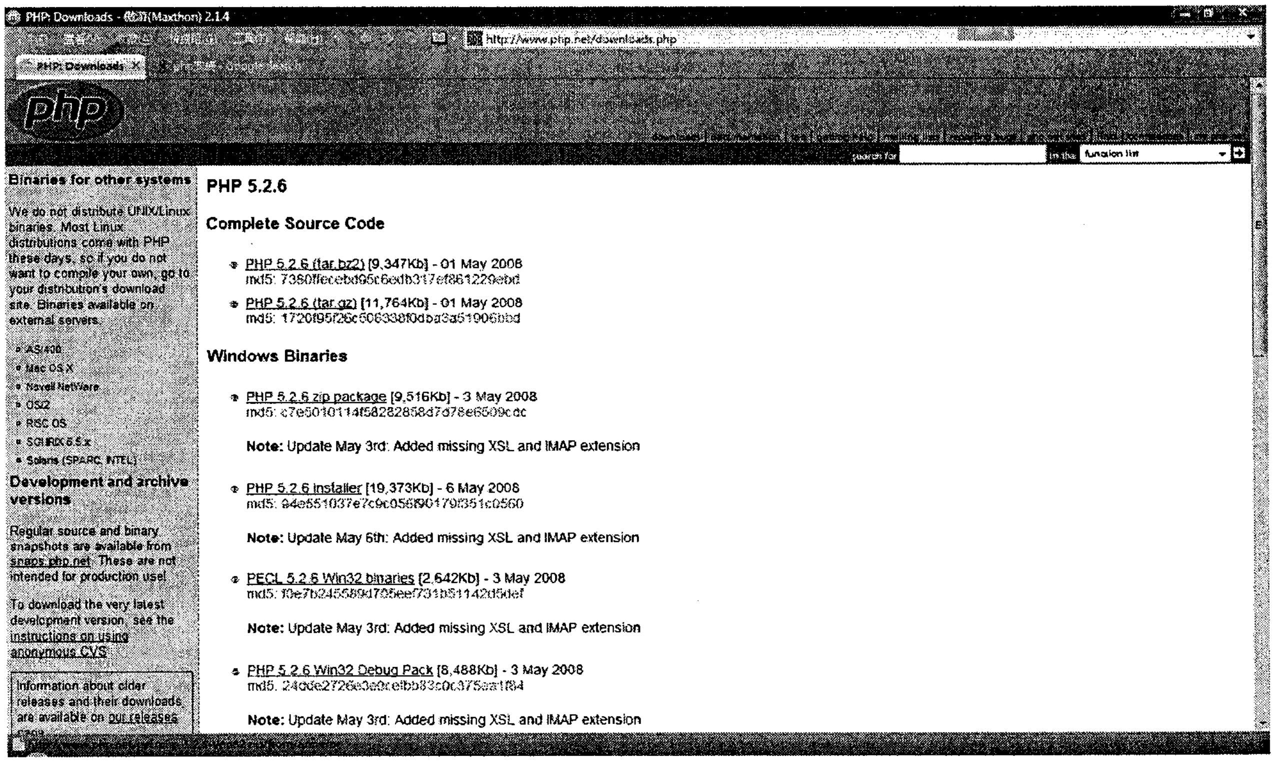
Physical path: A:\Programming\webroot

Pass-through authentication

Figure A.8 Internet Information Services (IIS) Manager

A.2.2. Installation of PHP

PHP installation package can be downloaded at web page http://www.php.net/downloads.php . Take Windows operation system as an example; download the latest "Windows Binary" zip file on the download web page as Figure A.9.


PHP: Downloads - (Maxthon) 2.1.4
http://www.php.net/downloads.php
PHP: Downloads
php

Binaries for other systems

We do not distribute UNIX/Linux binaries. Most Linux distributions come with PHP these days, so if you do not want to compile your own, go to your distribution's download site. Binaries available on external servers.

- AS/400
- Mac OS X
- Novell NetWare
- OS/2
- RISC OS
- SGI IRIX 6.5.x
- Solaris (SPARC, INTEL)

Development and archive versions

Regular source and binary snapshots are available from snaps.php.net. These are not intended for production use!

To download the very latest development version, see the instructions on using anonymous CVS.

Information about older releases and their downloads are available on our releases page.

PHP 5.2.6

Complete Source Code

- PHP 5.2.6 (tar.bz2) [9,347Kb] - 01 May 2008
  md5: 7380ffecebd95c6edb317ef861229ebd
- PHP 5.2.6 (tar.gz) [11,764Kb] - 01 May 2008
  md5: 1720f95f26c506338f0dba3a51906bbd

Windows Binaries

- PHP 5.2.6 zip package [9,516Kb] - 3 May 2008
  md5: c7e5010114f58282858d7d78e6509cdc

  Note: Update May 3rd: Added missing XSL and IMAP extension
- PHP 5.2.6 installer [19,373Kb] - 6 May 2008
  md5: 94e551037e7c9c056f90179f351c0560

  Note: Update May 6th: Added missing XSL and IMAP extension
- PECL 5.2.6 Win32 binaries [2,642Kb] - 3 May 2008
  md5: f0e7b245589d705ee731b51142d5def

  Note: Update May 3rd: Added missing XSL and IMAP extension
- PHP 5.2.6 Win32 Debug Pack [8,488Kb] - 3 May 2008
  md5: 24dde2726e3e0cefbb83c0c375ea1f84

  Note: Update May 3rd: Added missing XSL and IMAP extension

Figure A.9 Internet Information Services

Unzip the binary file downloaded from web page as Figure 2.1 to a folder which you wish to install PHP in.

Rename php.ini-dist under <PHP installation directory> to php.ini and copy it into folder c:\windows if windows operation system is installed in drive C. Open php.ini with notepad and search for “extension_dir”. Set the extension directory to “<PHP installation directory>\ext”.

Search for “; Windows Extensions” and set the extension section as follows.

```
;extension=php_bz2.dll
;extension=php_curl.dll
;extension=php_dba.dll
;extension=php_dbase.dll
;extension=php_exif.dll
;extension=php_fdf.dll
extension=php_gd2.dll
;extension=php_gettext.dll
;extension=php_gmp.dll
;extension=php_ifx.dll
;extension=php_imap.dll
;extension=php_interbase.dll
;extension=php_ldap.dll
extension=php_mbstring.dll
;extension=php_mcrypt.dll
;extension=php_mhash.dll
;extension=php_mime_magic.dll
;extension=php_ming.dll
;extension=php_msql.dll
;extension=php_mssql.dll
extension=php_mysql.dll
extension=php_mysqli.dll
;extension=php_oci8.dll
;extension=php_openssl.dll
;extension=php_pdo.dll
;extension=php_pdo_firebird.dll
;extension=php_pdo_mssql.dll
;extension=php_pdo_mysql.dll
;extension=php_pdo_oci.dll
;extension=php_pdo_oci8.dll
;extension=php_pdo_odbc.dll
;extension=php_pdo_pgsql.dll
;extension=php_pdo_sqlite.dll
```

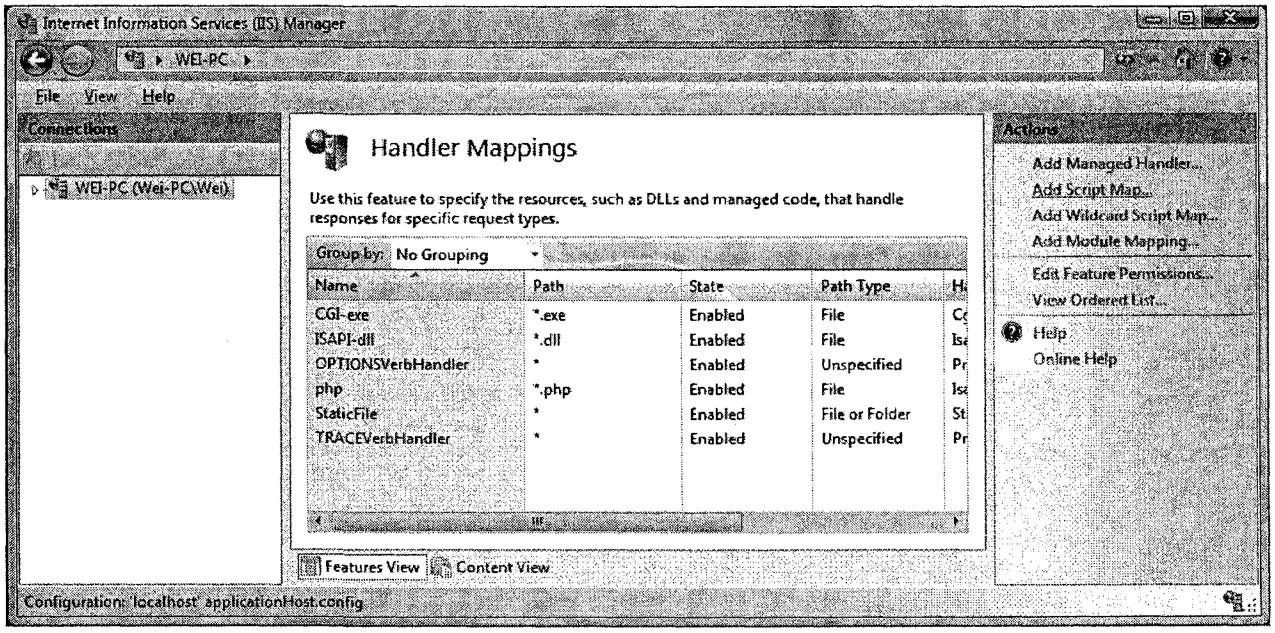
```
;extension=php_pgsql.dll
;extension=php_pspell.dll
;extension=php_shmop.dll
;extension=php_snmp.dll
;extension=php_soap.dll
;extension=php_sockets.dll
;extension=php_sqlite.dll
;extension=php_sybase_ct.dll
;extension=php_tidy.dll
;extension=php_xmlrpc.dll
extension=php_xsl.dll
;extension=php_zip.dll
```

Search for "upload_tmp_dir", and set a temporary folder uploading files, and search for "session.save_path" and set a folder to save session.

A.2.3. Configuration of PHP and Web Server

After installing PHP, some configuration for PHP on web server IIS is very necessary. Go to Control Panel and double click "Administrative Tools" as Figure 1.5. Double click "Internet Information Services (IIS) Manager" as Figure 1.6. Double click "Handler Mapping" on "Internet Information Services (IIS) Manager" window as Figure A.7.

Click "Add Script Map" in action side bar on the right of "Handler Mapping" window as Figure A.10.



Handler Mappings

Use this feature to specify the resources, such as DLLs and managed code, that handle responses for specific request types.

Group by: No Grouping

Name	Path	State	Path Type	H
CGI-exe	*.exe	Enabled	File	C
ISAPI-dll	*.dll	Enabled	File	Is
OPTIONSVerbHandler	*	Enabled	Unspecified	Pr
php	*.php	Enabled	File	Is
StaticFile	*	Enabled	File or Folder	St
TRACEVerbHandler	*	Enabled	Unspecified	Pr

Configuration: 'localhost' applicationHost.config

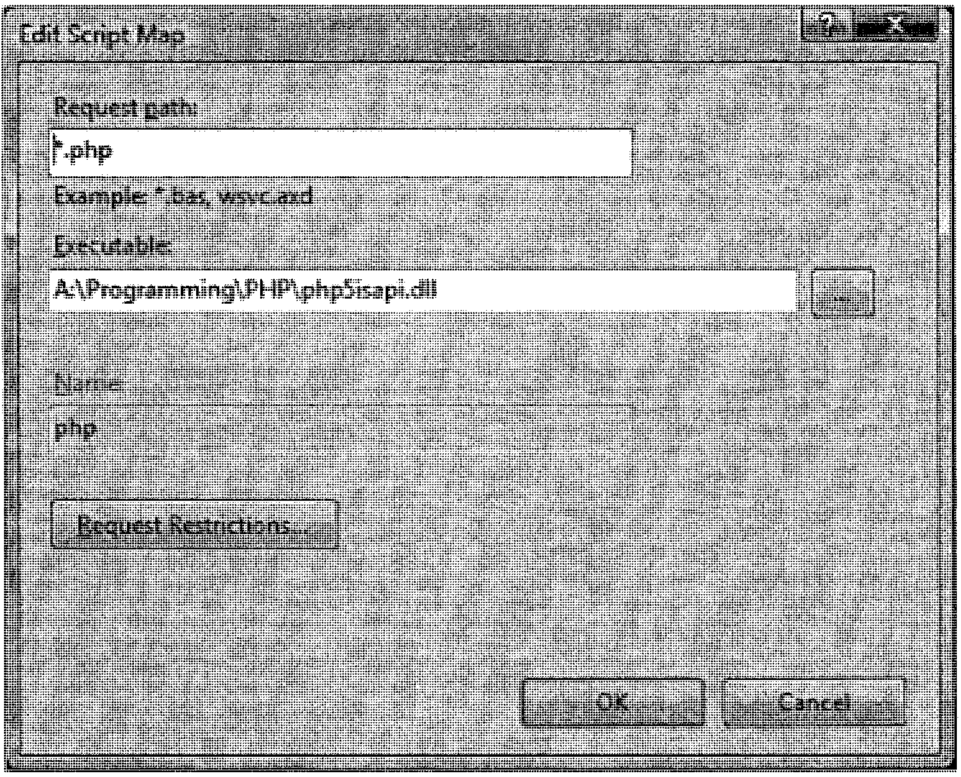
Figure A.10 Handler Mapping Window

“Edit Script Map” dialog pops out after clicking “Add Script Map” on the right side bar of “Handler Mapping” window as Figure A.10.

Fill in “Edit Script Map” dialog as Figure A.11.

- Type in “*.php” in textbox of “Request path”.
- Locate “php5isapi.dll” file with the browse button to the left of “Executable” textbox.
- Give a name to the script.

After filling the dialog, click OK button.


Edit Script Map

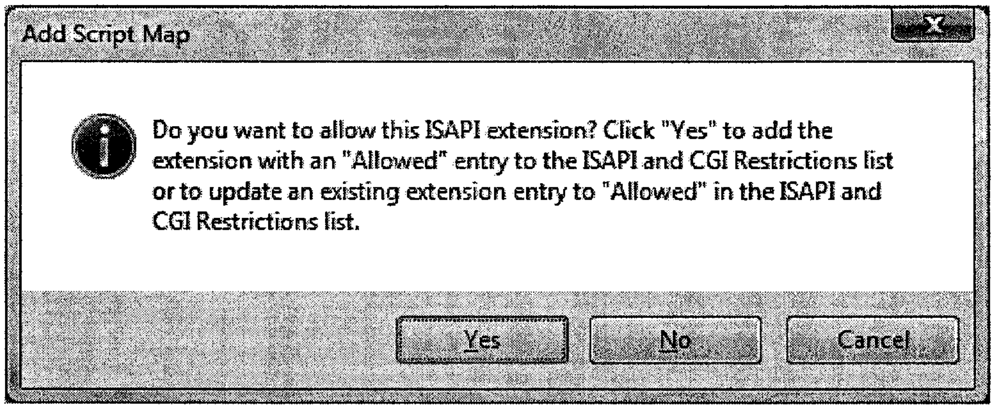
Request path:
*.php
Example: *.bas, wsvc.axd

Executable:
A:\Programming\PHP\php5isapi.dll

Name:
php

Figure A.11 Edit Script Map

After clicking OK button on “Edit Script Map” dialog as Figure A.11, a confirmation dialog pops out as Figure A.12 and click “Yes” on it to register the script on IIS.



Add Script Map

Do you want to allow this ISAPI extension? Click "Yes" to add the extension with an "Allowed" entry to the ISAPI and CGI Restrictions list or to update an existing extension entry to "Allowed" in the ISAPI and CGI Restrictions list.

Figure A.12 Edit Script Map

A.2.4. Test PHP on Web Server IIS

Create a file with content "<?php phpinfo();?>" and save as "test.php" under web root directory which specified in Figure 1.8. Open a web browser (Internet Explorer, Firefox, etc.), and type http://localhost/test.php in address bar. If a web page as Figure A.13 shows up, it proves that PHP works find with Web Server IIS.

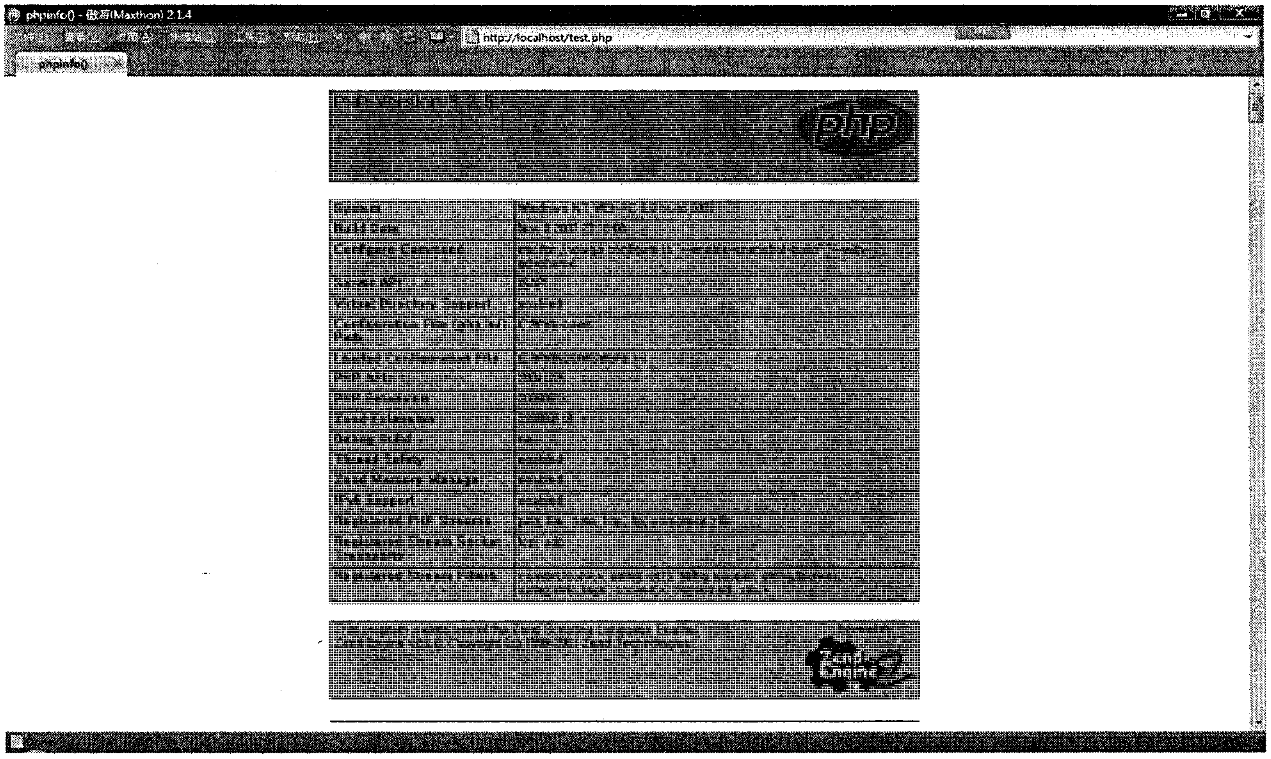
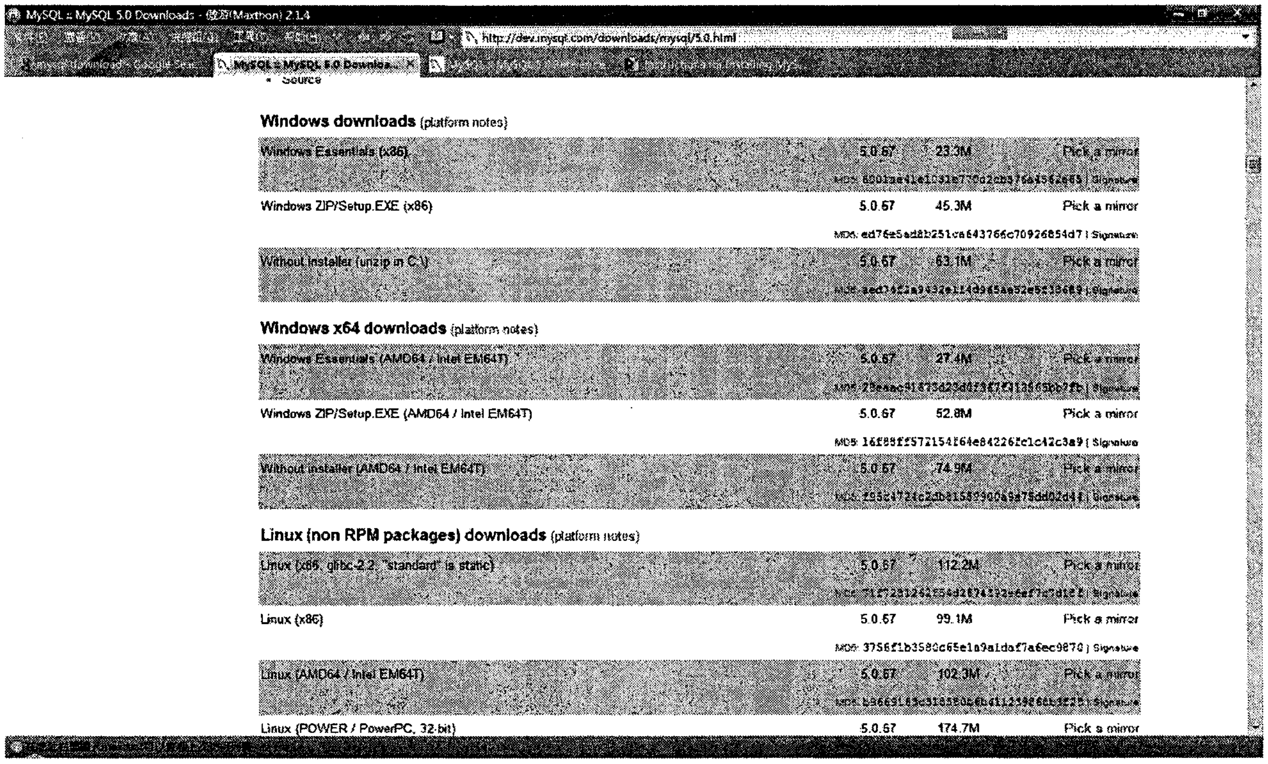


Figure A.13 phpinfo Page

A.2.5. Installation of MySQL

Download MySQL from http://dev.mysql.com/downloads/mysql/5.0.html . For Windows operation system, choose "Windows Essentials" under "Windows downloads" on the web page as Figure A.14.


MySQL :: MySQL 5.0 Downloads - (Maxthon) 2.1.4
http://dev.mysql.com/downloads/mysql/5.0.html
MySQL :: MySQL 5.0 Downloa...
Source

Windows downloads (platform notes)

Windows Essentials (x86)	5.0.67	23.3M	Pick a mirror
			MD5: [illegible] | Signature
Windows ZIP/Setup.EXE (x86)	5.0.67	45.3M	Pick a mirror
			MD5: ed76e5ad8b251ca643766c70926854d7 | Signature
Without installer (unzip in C:\)	5.0.67	63.1M	Pick a mirror
			MD5: [illegible] | Signature

Windows x64 downloads (platform notes)

Windows Essentials (AMD64 / Intel EM64T)	5.0.67	27.4M	Pick a mirror
			MD5: [illegible] | Signature
Windows ZIP/Setup.EXE (AMD64 / Intel EM64T)	5.0.67	52.8M	Pick a mirror
			MD5: 16f88ff572154f64e84226fc1c42c3a9 | Signature
Without installer (AMD64 / Intel EM64T)	5.0.67	74.9M	Pick a mirror
			MD5: [illegible] | Signature

Linux (non RPM packages) downloads (platform notes)

Linux (x86, glibc-2.2, "standard" is static)	5.0.67	112.2M	Pick a mirror
			MD5: [illegible] | Signature
Linux (x86)	5.0.67	99.1M	Pick a mirror
			MD5: 3756f1b3580c65e1a9a1daf7a6ec9870 | Signature
Linux (AMD64 / Intel EM64T)	5.0.67	102.3M	Pick a mirror
			MD5: [illegible] | Signature
Linux (POWER / PowerPC, 32-bit)	5.0.67	174.7M	Pick a mirror

Figure A.14 MySQL Download Web Page

Double click the msi installation file download from the web page as Figure A.14 to start the installation. Use the following setting during the installation.

- Typical Setup
- Skip Sign-Up
- make sure "Configure the MySQL Server now" is checked
- "Detailed Configuration"
- "Developer Machine"
- "Multifunctional Database"
- "InnoDB Tablespace Settings" - leave everything default
- "Decision Support (DSS)/OLAP"

- make sure "Enable TCP/IP Networking" is checked and leave the port number at 3306 (at this point, if you have a firewall, it will usually try to access itself on the localhost)
- "Standard Character Set"
- check "Install As Windows Service"
- enter your root password
- press "execute" and it'll install and set it up.

By default, MySQL will run automatically when Windows operation system starts.

A.2.6. Importing GAS Configuration Data to MySQL

Unzip wordnet30.zip and wordpress.zip on CD to <MySQL Installation Directory>/data.

A.2.7. Installation of GAS

Unzip GAS.zip to web root directory specified in "Edit Site" dialog as Figure A.8.

A.2.8. Run GAS

Open a web browser (Internet Explorer, Firefox, etc) and type http://localhost/GAS in address bar as Figure A.15.

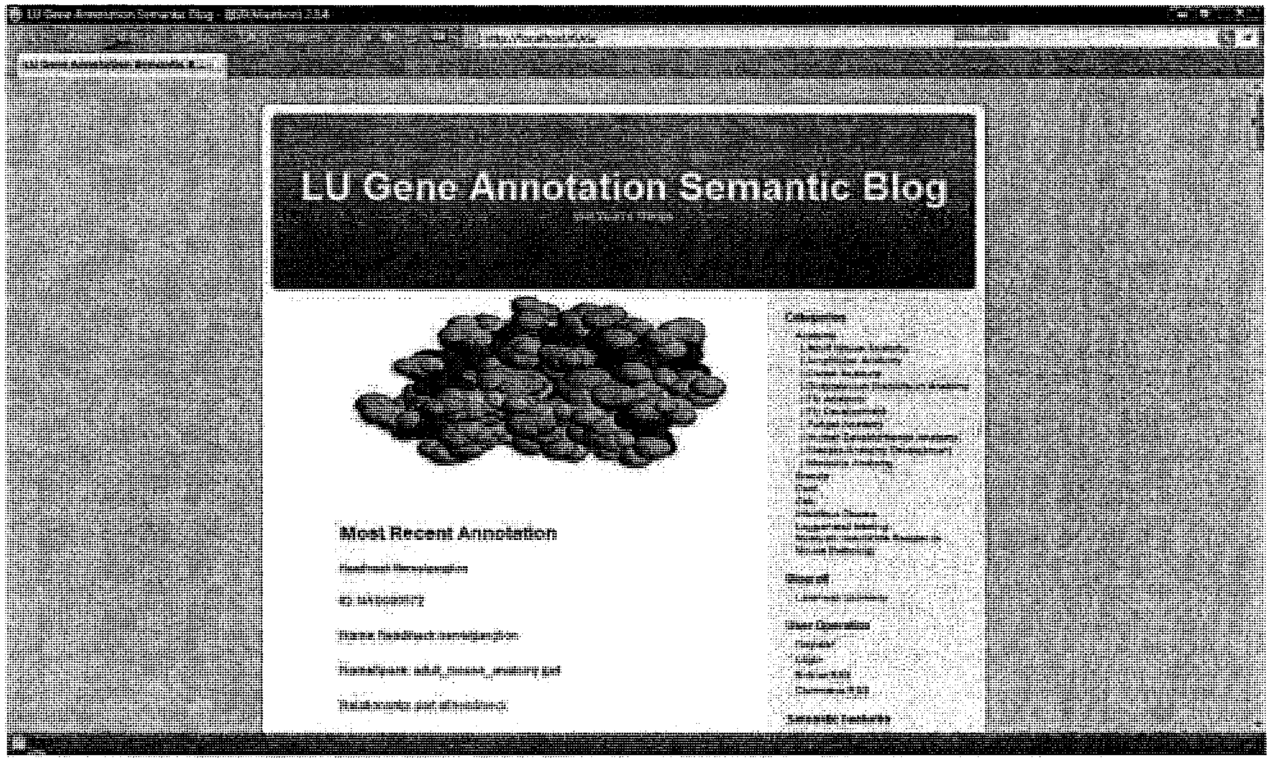
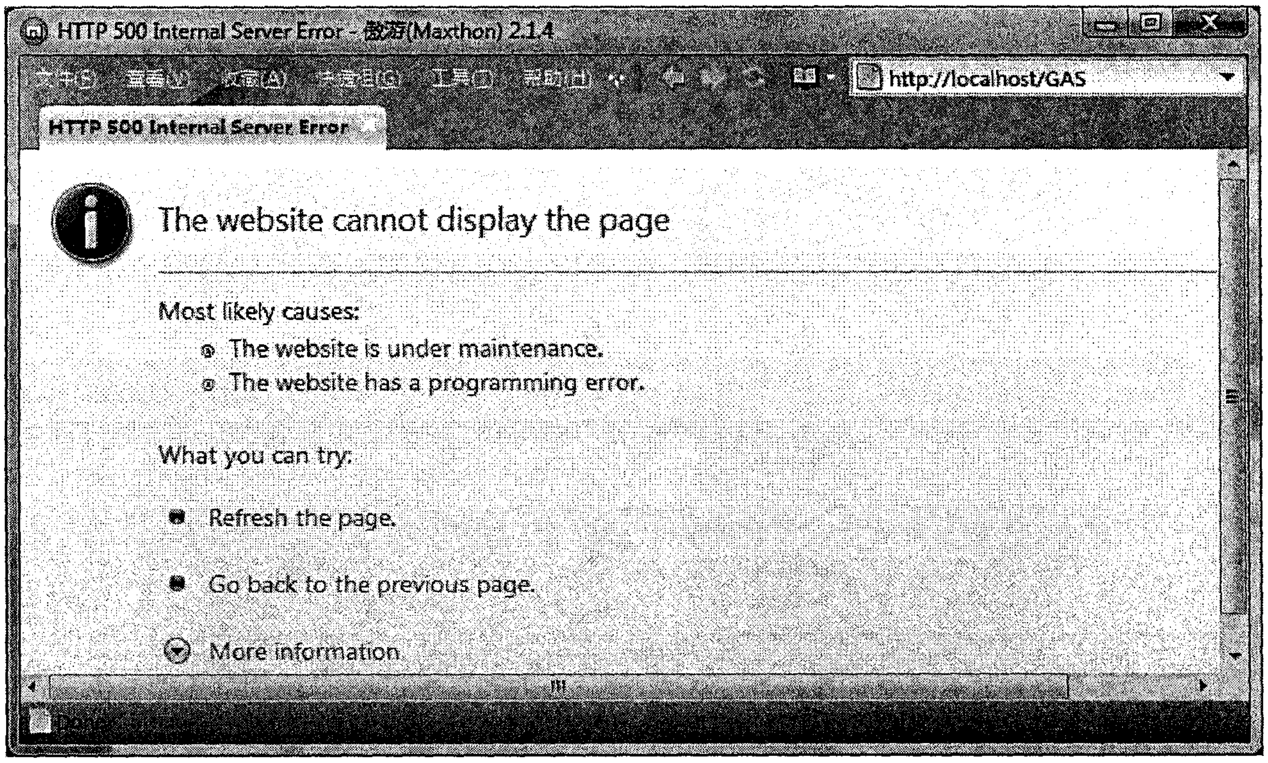


Figure A.15 GAS Interface

A.2.9. Troubleshooting

There are two frequent problems if URL http://localhost/GAS leads to an error page as Figure A.16 instead of GAS interface as Figure A.15.


HTTP 500 Internal Server Error - 傲游(Maxthon) 2.1.4
http://localhost/GAS
HTTP 500 Internal Server Error

The website cannot display the page

Most likely causes:

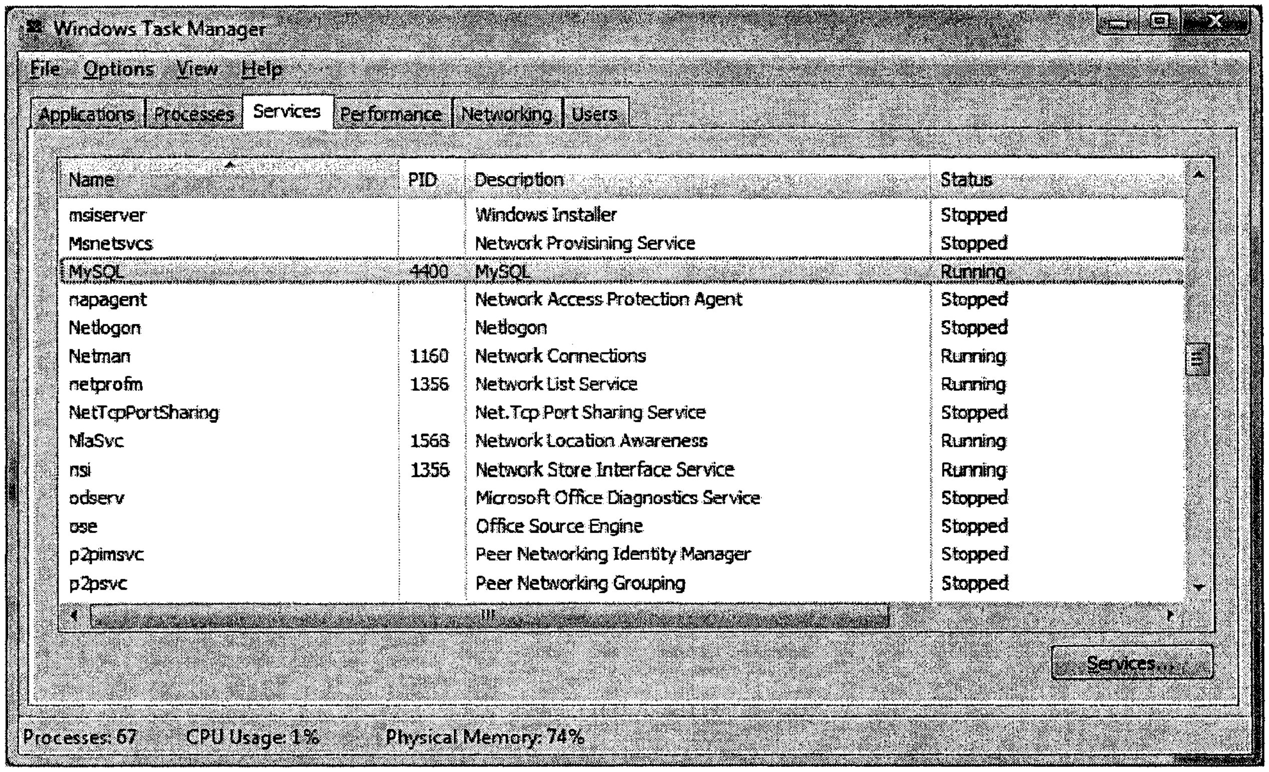
- The website is under maintenance.
- The website has a programming error.

What you can try:

- Refresh the page.
- Go back to the previous page.
- More information

Figure A.16 Error Page

First thing to do is to go to Windows Task Manager as Figure A.17, and make sure that status of service MySQL is "Running".


Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
Services...

Name	PID	Description	Status
msiserver		Windows Installer	Stopped
Msnetsvcs		Network Provisining Service	Stopped
MySQL	4400	MySQL	Running
napagent		Network Access Protection Agent	Stopped
Netlogon		Netlogon	Stopped
Netman	1160	Network Connections	Running
netprofm	1356	Network List Service	Running
NetTcpPortSharing		Net.Tcp Port Sharing Service	Stopped
NlaSvc	1568	Network Location Awareness	Running
nsi	1356	Network Store Interface Service	Running
odserv		Microsoft Office Diagnostics Service	Stopped
ose		Office Source Engine	Stopped
p2pimsvc		Peer Networking Identity Manager	Stopped
p2psvc		Peer Networking Grouping	Stopped

Processes: 67 CPU Usage: 1% Physical Memory: 74%

Figure A.17 Windows Task Manager

Then, go to Internet Information Services (IIS) Manager as Figure A.7 and make sure that "Start" under "Manage Web Site" section on right side bar of Internet Information Services (IIS) Manager Window as Figure A.7 is grey, which means it is running.

APPENDIX B
MAIN PROGRAMMING MODULES OF GAS

The following is a list of tools and technologies used in GAS system.

AJAX: AJAX is a technique that allows web browsers exchange data with server without reloading the web page. It was used in GAS system to prompt users with possible evidence code when they annotating.

Flock: Flock is a web browser based on Firefox. However, it integrates the feed aggregator with it. Flock was used as a feed aggregator (Figure 3.10) to syndicate Gene Ontology and annotation feeds in this thesis.

Internet Information Services (IIS): Internet Information Services (IIS) is web server made by Microsoft. It is the second most popular web server in the world. In this thesis, Internet Information Services (IIS) version 7 was used as a web server on Windows Vista platform to develop GAS system.

MySQL: MySQL is a rational database management system. MySQL community server version 6.02 was used in GAS system to store system configuring information.

PHP: PHP is a server-side HTML embedded scripting language. PHP version 5.2.5 is the programming language for developing GAS system.

PSPad: PSPad is a programming editor on Windows platform. It was used as an editor to code PHP files.

RSS: RSS is a web feed format and it was used in GAS system as syndication format for Gene Ontology and annotations.

SCORM: SCORM manifest standard was used to wrap Gene Ontology Navigation results into Learning Objects.

WordNet: WordNet® is a large lexical database of English. It was used to look up synonyms of metadata in Gene Ontology Navigation.

WordPress: WordPress was the open source used as system framework on which GAS system was built.

XML-based Languages: In GAS system, Gene Ontology, annotation, and Gene Ontology Navigation results are stored in XML-based languages.