Semantic Eye

Semantic Based UAV Images Repository and Searching System

By

NC Kashif Javed

NC Muhammad Husnain Ashfaq

NC Risham Waqar

Capt Adil Hasnain Aslam

Submitted to the Faculty of Computer Science
National University of Sciences and Technology, Rawalpindi in partial fulfillment for the requirements of a B.E Degree in Computer Software Engineering

AUGUST 2009

ABSTRACT

Semantic Eye

Semantic Based UAV Images Repository and Searching System

Manual management of UAV/Satellite images was highly inefficient as the volume of images grows exponentially with time. Searching of images was become impossible in this situation. So to cope this problem we have developed a web based system named “Semantic Eye”. This system is capable of storing a huge amount of images with their metadata (like objects in images, geospatial coordinates, mission date etc.) and search those images semantically and very efficiently.

We have used ASP.Net, Java, C#, Protégé, Jena to develop this system. When the user will upload the image he can tag different regions and objects of image, give its metadata like date, time, coordinates, comments. User can also pre-process this image. This image will be stored in HDF (Database used by NASA to store scientific data in bulk) database and also published semantically for search purpose.

For searching the images user will give the criteria like objects in images, coordinated, location, date, time etc. The image is searched semantically using Jena API. Once the image is searched user can tag, preprocess or comment on image. This project is completed under supervision of Maj Dr Naveed Iqbal Rao [IPC MCS] and Lec. Amna Basharat [FAST-NU].

COPYRIGHT NOTICE

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of any other award or qualification either at National University of Sciences and Technology or any other institution.

DEDICATION

I dedicate this project to my Parents

(Kashif Javed)

I dedicate this project to the four people I love the most, my father, mother, brothers and sister

(Muhammad Husnain Ashfaq)

I dedicate this project to my parents

(Risham Waqar)

Dedicated to my parents

(Capt Adil Hasnain Aslam)

ACKNOWLEDGMENTS

We are very grateful to Allah Almighty for giving us the confidence and strength for carrying out a research based project and for making us able to come up with an application that will not only be useful to Linux users but will also pave path for further research and development in the field of computer memory analysis.

We also thank our instructors and friends for their moral and technical suggestions without which we might not have made it this far.

Above all, we are deeply thankful to our beloved parents for their patience, support and prayers that helped us gain our goal. We simply cannot thank them in words for they sacrificed their own comfort in order to give us a life that we need and by sponsoring us throughout our life.

Table of Contents

List of Tables……

List of figures……..

1.

Introduction…….

1.1 Preface…….

1.2 Project Vision……..

1.3 Proposed Solution……

1.4 Aim of the Project……

1.5 Organization of Project Report……..

2. Literature Review……

3. SYSTEM ANALYSIS…….

3.1

Introduction……..

3.2 Project

Scope……

3.3 Requirement Specification……

3.3.1 External Interface Requirements……..

3.3.1.1 User Interface……

3.3.1.1.1 GUI……

3.3.1.1.2 API……

3.3.1.1.3 Diagnostics……

3.3.1.2 Software Interfaces……

3.3.2 Major Functional Requirements ……

3.3.3 Major Non-Functional Requirements……..

3.4 Use Case Diagram……

3.5 Sequence Diagram……

3.6 Conclusion……

4. SYSTEM DESIGN……

4.1

Introduction……..

4.2 Architectural Diagram……

4.3 High Level Diagram……

4.4 Low Level Diagram***……

4.4.1 Overall Client……..

4.4.2 Overall Server……

4.4.3 Voice Chat Module……

4.4.4 FTP Client……..

4.4.5 FTP Server……

4.5 Class Diagram……

4.6 Data Flow Diagram……

4.7 Interaction Diagram……

4.8 Conclusion……

5. IMPLEMENTATION……

5.1

Introduction……

5.2 Implementation Language……

5.3 Distribution of Classes with respect to Modules***……

5.4 Conclusion……

6. TESTING……

6.1

Introduction……

6.2 Testing Process……

6.2.1 Unit Testing***……

6.2.2 Component Testing***……

6.2.3 Integration Testing***……

6.2.4 White Box Testing……

6.2.5 Black Box Testing***……

6.2.6 Static Analysis of Code……

6.2.6.1 Control Flow Analysis……

6.2.6.2 Data Analysis……..

6.2.6.3 Interface Analysis…….

6.2.7 Conclusion…….

7. FUTURE WORK AND CONCLUSION……

7.1 Future Work…….

7.2 Conclusion…….

APPENDIX A (Research Paper) ……..

APPENDIX B (Hardware & Software Requirements) ……

APPENDIX C (User Manual) ……..

APPENDIX D (Deployment Manual) ……

APPENDIX E (Symbols & Abbreviations) ……

APPENDIX F (Bibliography) ……

Chapter 1

Introduction

1.1 Preface

Semantic Eye is web-based application for storing images with their metadata, annotating and efficient searching UAV/Satellite Images. Manual management of UAV/Satellite images was highly inefficient as the volume of images grows exponentially with time. Searching of images was become impossible in this situation. So to coup this problem we have developed a web based system named “Semantic Eye”. This system is capable of storing a huge amount of images with their metadata (like objects in images, geospatial coordinates, mission date etc.) and searches those images semantically and very efficiently.

We have chosen the option of web application so that no software on client side is needed to use it, other than web browser. The Semantic Web is an evolving development of the World Wide Web in which the semantics of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the web content.

Major drawbacks of the existing system (Manually Managed) are as under:-

- Searching of images was impossible.
- It was hectic job to manage the images.
- It was impossible to search the images which have desired objects in it.

1.2 Project Vision

The present system was manual, managing and searching the images was hectic job. So the current web system will store and search through images very efficiently.

1.3 Proposed Solution

The solution proposed for above mentioned problems is

- To store images through a web application in database.
- To add meta-data like date, time, and geospatial coordinates to images.
- To add tags to different regions and objects in image.
- To pre-process the image for better view of images if desired by user.
- To search through these uploaded images semantically.
- To search objects in images, or to search images with specific objects.
- To add comments by as many as users viewed the image, and desire to add comment.

1.4 Aim of the Project

The aim of the project is store all the UAV/Satellite images in a repository through a web application. To make a system which incorporate the scenario like user want to store image with tags, comments and other data. User will be able to search repository on the base of these tags like tanks, guns, location, terrain, GIS Location, etc. The search result will include the image and all the data associated with it like tags, comments by all users, meta-data. User will be able to comment and tag the searched image again. The aim of the project is to store and search through these images efficiently and using the semantics of images, by semantic of images we mean what objects are tagged in these images and what is the base class of these objects like gun, ships, vehicles, landmarks etc. So the user will be able to search by class as well as the objects itself present in the images. We introduced the semantics in this system by the mean of ontology. Ontology contains all the base classes, inherited classes, interrelated classes, and the individuals of these classes. Now the use of HDF as database of images and Semantics both are the emerging technologies to handle scientific and imagery data efficiently.

This could be achieved by developing a web application with following main modules.

1- **Uploading and tagging**

In this module we will provide the functionality that User will be able to upload the images with their Geographical co-ordinates, date, time, mission number and user who upload them. User can tag the objects in the image. These images will be stored in HDF5 files using high level APIs.

2- Preprocessing images

User will be able to pre-process image to enhance its visibility. Following operations will be provided to enhance the image

a. Binarisation
b. Edge Detection
c. Brightness and contrast correctness
d. Color enhancement
e. Sharpen
f. Reduce Noise

3- Publishing in semantic form

All the tags will be mapped on ontology and saved the tags as the individual of main class of GIS_Images in owl file, using the protégé API.

4- Searching

In this we will provide the functionality to user to search the images using the semantics. User will query the semantic engine via user friendly GUI. The returned image will be displayed the all its metadata and tagged objects.

1.5 Organization of Project Report

The project report has been drafted carefully deciding the sequence to be followed. After the introduction section, the report incorporates the Literature Review chapter summarizing the text studied and research aspects of our project before and during the project's execution. Subsequently, the System Analysis chapter comes which includes the major interface, functional and non-functional requirements of the system. Next is the System Design chapter comprising of the architectural diagram, high and low level design, data flow diagram, class diagram and interaction diagram. Following this the report includes the implementation chapter identifying and elucidating the classes which are implemented. Then is the testing chapter incorporating the testing process employed to test the system and the results that were obtained. The next chapter then discusses the work that can be done in future to further enhance the system and ultimately this chapter wraps the report.

Chapter 2

Literature Review

2.1 Introduction

2.2 Semantic Web

2.2.1 The Semantic Web Impact- Knowledge Management

- Knowledge management concerns itself with acquiring, accessing, and maintaining knowledge within an organization
- Key activity of large businesses: internal knowledge as an intellectual asset
- It is particularly important for international, geographically dispersed organizations
- Most information is currently available in a weakly structured form (e.g. text, audio, video)

2.2.2 Limitations of Current Knowledge Management Technologies

- *Searching information*
  - *Keyword-based search engines*
- *Extracting information*
  - human involvement necessary for browsing, retrieving, interpreting, combining
- *Maintaining information*
  - Inconsistencies in terminology, outdated information.
- *Viewing information*
  - Impossible to define views on Web knowledge

2.2.3 Semantic Web Enabled Knowledge Management

- Knowledge will be organized in conceptual spaces according to its meaning.
- Automated tools for maintenance and knowledge discovery
- Semantic query answering
- Query answering over several documents
- Defining who may view certain parts of information (even parts of documents) will be possible.

2.2.4 Semantic Web Technologies

2.2.4.1 Explicit Metadata

- This representation is far more easily process able by machines
- Metadata: data about data
- Metadata capture part of the *meaning of data*
- Semantic Web does not rely on text-based manipulation, but rather on machine-processable metadata

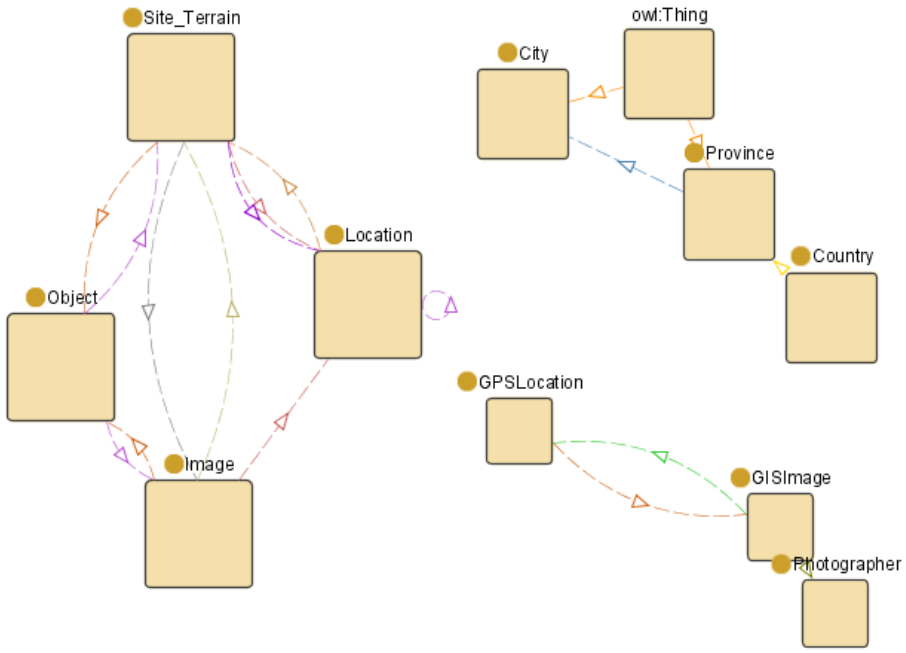
2.2.4.2 Ontologies

- The term ontology originates from philosophy
- The study of the nature of existence
- Different meaning from computer science
- An ontology is an explicit and formal specification of a conceptualization

2.2.4.2.1 Typical Components of Ontologies

- **Terms** denote important concepts (classes of objects) of the domain
- **Relationships** between these terms: typically class hierarchies
- **Properties:**
- **Value restrictions**
- **Disjointness statements**
- **Logical relationships between objects**

2.2.4.2.2 Example of a Class Hierarchy



2.2.4.2.3 The Role of Ontologies on the Web

- Ontologies provide a shared understanding of a domain: semantic interoperability
- overcome differences in terminology
- mappings between ontologies
- Ontologies are useful for the organization and navigation of Web sites

2.2.4.3 Logic and Inference

- Logic is the discipline that studies the principles of reasoning

- Formal languages for expressing knowledge
- Well-understood formal semantics
- Declarative knowledge: we describe what holds without caring about how it can be deduced
- Automated reasoners can deduce (infer) conclusions from the given knowledge

2.2.4.3.1 Logic vs Ontologies

- The previous example involves knowledge typically found in ontologies
  - Logic can be used to uncover ontological knowledge that is implicitly given
  - It can also help uncover unexpected relationships and inconsistencies
- Logic is more general than ontologies
  - It can also be used by intelligent agents for making decisions and selecting courses of action

2.2.5 On HTML

- Web content is currently formatted for human readers rather than programs
- HTML is the predominant language in which Web pages are written (directly or using tools)
- Vocabulary describes presentation

2.2.5.1 Problems with HTML

- Humans have no problem with this
- Machines (software agents) do:
    - How distinguish therapists from the secretary,
    - How determine exact consultation hours
    - They would have to follow the link to the State Of Origin games to find when they take place.

2.3 XML

XML (**Extensible Markup Language**) is a general-purpose *specification* for creating custom markup languages. The term *extensible* is used to indicate that a markup-language designer has significant freedom in the choice of markup elements.

XML's goals emphasize representing documents with simplicity, generality, and usability over the Internet. XML has become widely-used as a general-purpose data-interchange format.

XML's set of tools helps developers in creating web pages but its usefulness goes well beyond that. XML, in combination with other standards, makes it possible to define the content of a document separately from its formatting, making it easy to reuse that content in other applications or for other presentation environments. Most importantly, XML provides a basic syntax that can be used to share information between different kinds of computers, different

applications, and different organizations without needing to pass through many layers of conversion.

2.4 RDF

- RDF is a data model for objects and relations between them
- RDF Schema is a vocabulary description language
- Describes properties and classes of RDF resources
- Provides semantics for generalization hierarchies of properties and classes

2.5 OWL

- A richer ontology language
- relations between classes
    - e.g., disjointness
- cardinality
    - e.g. “exactly one”
- richer typing of properties
- characteristics of properties (e.g., symmetry)

2.6 Reason to use owl

Ontologies are usually expressed in a logic-based language, so that detailed, accurate, consistent, sound, and meaningful distinctions can be made among the classes, properties, and relations. Some ontology tools can perform automated reasoning using the ontologies, and thus provide advanced services to intelligent applications such as conceptual/semantic search and retrieval, software agents, decision support, speech and natural language understanding, knowledge management, intelligent databases, and electronic commerce.

The OWL language provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users:

- OWL Lite supports those users primarily needing a classification hierarchy and simple constraint features.
- OWL DL supports those users who want the maximum expressiveness without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems. OWL DL was designed to support the existing Description Logic business segment and has desirable computational properties for reasoning systems.
- OWL Full is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. OWL Full allowsan ontology to augment the meaning of the predefined RDF or OWL vocabulary.

2.7 Protege

2.7.1 Reasons to use Protégé

2.8 Jena

Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine.

Jena is open source and grown out of work with the HP Labs Semantic Web Program.

The Jena Framework includes:

- A RDF API
- Reading and writing RDF in RDF/XML, N3 and N-Triples
- An OWL API
- In-memory and persistent storage
- SPARQL query engine

2.9 Pellet

For semantically-enabled applications that need to represent and reason about information using OWL, Pellet is the leading choice for systems where sound-and-complete OWL DL reasoning is essential.

Pellet also includes support for OWL 2 profiles including OWL 2 EL. (Very scalable OWL 2 QL reasoning is available in Owlgres.) It also incorporates various optimization techniques, including novel optimizations for nominals, conjunctive query answering, and incremental reasoning. There's more detailed information about the architecture of the system and its features in the Pellet Help.

Pellet fully supports the original OWL DL specification. OWL 2 is a forthcoming revision of the international web standard produced by the W3C for web-friendly ontologies. Pellet currently supports most of the features proposed in OWL 2.

An OWL DL reasoner like Pellet is a core component of ontology-based data management applications.

2.10 Difference between pellet and Jena query engine

The Jena query engine is RDF triple based, so to produce variable bindings it works one triple at a time. In contrast, the Pellet query engine considers the entire conjunctive query. This difference causes the engines to have different performance characteristics and in some cases yields different results.

First, by using a level of representation consistent with the logic, the Pellet query engine can perform optimizations that are inaccessible to the Jena query engine. These optimizations often yield a significant speed-up in query answering.

Second, results may differ based on handling of blank nodes in queries. The semantics of SPARQL are such that blank nodes need not be bound to asserted individuals and can match individuals that are inferred (such as those from existential restrictions and minimum cardinality constraints). The Pellet engine will produce results using these inferred individuals and the Jena engine will not. An example demonstrating how this can change results is included in the examples directory of the Pellet distribution and is accessible in the svn repository.

2.11 Hierarchical Data Format (HDF5)

These are the reasons why we chose HDF5 for data storage.

1. A versatile data model that can represent very complex data objects and a wide variety of metadata.
2. A completely portable file format with no limit on the number or size of data objects in the collection.
3. A software library that runs on a range of computational platforms, from laptops to massively parallel systems, and implements a high-level API with C, C++, Fortran 90, and Java interfaces.
4. A rich set of integrated performance features that allow for access time and storage space optimizations.
5. Tools and applications for managing, manipulating, viewing, and analyzing the data in the collection

Chapter 3

System Analyses

3.1 Introduction

This chapter covers the system analysis phase of the project. In this phase, first of all scope of the project is presented as it's clear definition and understanding is needed for the absolute comprehension of the system's requirements' specification phase, including major functional and non-functional requirements, is described. The requirements' specification phase is then followed by use case diagram and domain model of the system for the better understanding of the system analysis phase of the project.

3.2 Project Scope

Semantic Eye is framework which uses power of Semantic Web in extracting image description from tagged and annotated UAV/Satellite images and retrieval of image intelligently.

Objective of this project is to Maintain Repository of images such that images with similar spatial info are grouped together. These images will be stored with their tags and then semantic search will be used to search through their images and tags.

3.3 Requirements Specification

3.3.1 Introduction

3.3.1.1 Purpose

This Software requirement Document is written to record and analyze the requirements of Semantic Eye framework. Semantic Eye is semantic web based search engine of images and their repository (storage of images with their metadata). This document covers all the features and components of Semantic Eye framework. This

initial SRS document if in coming iterations some changes occur in requirements this document will be revised.

3.3.1.2 Intended Audience

This document is written for following type of audience.

1- Project Team

So that they can see at any stage that which requirements have been met and which are left? And to complete the project within scope.

2- Project Evaluation Team

So that the evaluation team know what is scope of project? What are the requirements of project?

3- User of System

So that they know what this is made to do? What are its functional and non functional requirements? How to use it efficiently?

3.3.1.3 Project Scope

Semantic Eye is framework which uses power of Semantic Web in extracting image description from tagged and annotated images and retrieval of image intelligently.

Objective of this project is to Maintain Repository of images such that images with similar spatial info are grouped together. These images will be stored with their tags and then semantic search will be used to search through their images and tags.

3.3.2 Overall Description

3.3.2.1 Product Perspective

Till now all the search engines are using only tags which are used to search them. But Semantic Eye is framework which will search through those images by the power of semantic, Means that it will classify image based on the relation between the tags of images. While the user will be able to upload, edit and search these images through web interface. While the most importantly we will write .net wrappers for HDF5. HDF5 will be used to store the images and data associated.

3.3.2.2 Product Features

User of this system will be able to perform following functions:

- User will be able to upload the images with their Geographical co-ordinates.
- User can tag the objects in the image.
- These images will be stored in HDF5 files using high level APIs.
- User then can search from the database of images on base of their geospatial coordinates and time.
- Semantic web will be used to search in the images on the basis of semantics of image. First the query given by user will be described in semantics then it will be mapped on the knowledgebase of all the stored images.
- This system will be able to search through the distributed database. That user can access data over the internet regardless of their physical location.
- Proper help and manual will be provided with this product.

3.3.2.3 User Classes and Characteristics

Users will of three classes

1- Administrators

These users are responsible and have privilege to upload images, edit the tags and edit associated information and comments with the images. This type of user should have proper training of system. And they will require proper web interface to upload, edit and search these images.

2- Normal Users

These users will only search through the images. And can see detail attached with those images. These users will not require very strong knowledge of this system.

3.3.2.4 Operating Environment

The server of Semantic Eye system will operate in windows environment but as the users can access it on web so its client side will be platform independent. While in web browser we will require java plug-in. Server of Semantic Eye will be able to run on all windows XP, Windows Server, and Windows Vista. While the web interface will be accessible using any browser but for efficient use Internet Explorer is recommended. Semantic Eye will use Jena APIs for semantic search purpose. At the backend non-semantic data storage it will use HDF version 5.

3.3.2.5 Design and Implementation Constraints

We will only use HDF5 for storage of data in non-semantic form. We shall have to use .net wrappers for its integration to our system. We have to use Jana API'S for query engine to interact with the semantic data (OWL). There may be constraints with regards to defining rules which are developed using SWRL(semantic web rule language).OWL API'S will be used for building ontologies. The system must address many possible data failures including inconsistent data, duplicate data or incorrect data. The semantic layer should provide effective filtering that must sharply reduce the deluge of potential data. This system will use Apache Tomcat servelet / container for handling HTTP requests.

3.3.2.6 User Documentation

Functionality document will be provided to the user along with the software, describing how the system works in detail, while online demo and help will be available for both administrators and normal users.

3.3.2.7 Assumptions and Dependencies

We assume that the tags and geo-spatial coordinates associated with the images are within the domain knowledge means that for example we are talking about the aerial images then tags like lake, crops, river, tanks, cars etc. are within the domain but if the tags like apples, paperclips, computer etc are not within the domain of aerial images so their classification will be erroneous which will lead to wrong decisions.

In the same way if we talk about the time and geo-spatial coordinates these tags should also be realistic. The bandwidth used to connect client with servers will be enough that image searching and retrieving of images. One assumption is that the APIs of HDF5 will work fine as given in its specifications, same with the Jena APIs. We assume the images will not blur and then we assume that we have included every object in our ontology which can be possible to tag in images of our domain. Our system depends upon the web server IIS and Tomcat Apache server; our system will depend on their reliability. The software is dependent upon protégé to build ontologies .It is also dependent on HDF5 for image and metadata storage. While the client side module is dependent upon the client's browser.

3.3.3 External Interface Requirements

3.3.3.1 User Interfaces

We will provide the web based interface for both types of user (administrators and normal users). Administrator will use Google Maps like user interface to drop labels and to tag the images. It will be like pick a marker from the available list , drag it and then drop on the location of object in the image. While there will text fields for the comments, Geo-Spatial coordinates and time stamp. Administrator will also be able the search and then edit it.

For normal user the interface will be of only for searching there will be two options. One for simple searching on simple criteria like time and geo-spatial coordinates. When user will fill the text fields are

filled by user by pressing the search button on new page all related images with their attached metadata and comments will be displayed. Help button and proper hyperlink structure will be followed to prevent cognitive stress. We will use ASP .net forms style interface which is standard and easy to understand. While the options like auto filling, save passwords and login, will be provided.

3.3.3.2 Hardware Interfaces

No hardware will be used explicitly by our system it will only require servers and client machines which are connected to a data network.

3.3.3.3 Software Interfaces

As for as software interfaces are concerned, we will use Jena APIs to provide interface between query engine and semantically classified data. These APIs will be interface between java language and OWL files of ontology, these APIs will be used to creat a OWL file (ontology) of uploaded image at run time and to edit the information stored in OWL file.

As we know the communication between java and .net frame work is not possible without an interface. We have developed a threaded server in java which will run on server and communicate with C# for query processing, publishing etc. We have used socket of protocol TCP/IP and communicate through strings (AASCII Characters).

Another important interface is between the .net framework and HDF5. These wrappers will be used to store images and metadata associated with images from web page to HDF5 and retrieve data from

HDF5 to webpage. These wrappers will act as high level APIs, built on low level APIs. Interface between web browser and our server will be provided using ASP.net and IIS web server.

The interface between several servers will be catered fore by the SPARQL (semantic query language).

3.3.3.4 Communications Interfaces

For communication purposes our system will work by using simplest client server model using HTTP protocol. We will transfer images and data using the same protocol of communication. But for preserving the bandwidth we will AJAX model for communicating like auto filling, suggestions, comment editing, etc.

3.3.4 Nonfunctional Requirements

3.3.4.1 Performance Requirements

Performance of system is dependent on the bandwidth with which clients and servers are connected and the hardware specifications of our server like RAM, CPU clock rate, etc.

For auto filling and search suggestions the AJAX should be used so that in less bandwidth use we get required data from server, while for the efficient image and large data like comments storage system should use normal protocol like HTTP.

Load balancing should be applied on servers so that all clients get good service. Storage of non-semantic data we will use HDF5 so that performance of the system may increase as we can image and its Meta-data in the one file, it improves I/O performance, data can be read and written efficiently on parallel computing systems.

3.3.4.2 Security Requirements

Following are some security requirements of Semantic Eye system

- Password for all users should be changed frequently; this can be achieved by assigning expiry date to each password.
- User authentication can be achieved by digital signature of each user
- To allow only authorized access we should implement matching hashes of password instead of strings.
- There should be a firewall installed on server and client so that hacking attacks can be prevented.
- Anti-Malicious Software should be installed on both client and server side to prevent damage to data.
- As the data is confidential so it should be transmitted over the medium using encryption techniques.

3.3.4.3 Software Quality Attributes

This system will be implemented in currently popular technologies, and its design and development approach will be such that it will be easily maintainable. This system will be operating system

independent on client side. Robustness will provided by using the HDF5 faster and reliable data storage format.

Design and interface of Semantic Eye will be simple and easy to learn, so usability will be high. As in the development, we have used iterative approach so it should be able to testing will be very easy and ripple effect of error will not cause any damage to other components. This server of our system should be available 24/7 so that if any change or uploading is required system is ready to provide services. For good communication between the client and server very good amount of bandwidth should be available.

3.3.5 Functional Requirements

Following are the major functionalities that will be provided by Semantic Eye system

3.3.5.1 Image Annotation

User will be able to annotate images. User can pick a marker form a given list and drop it on the images where the object is present. While the user should also be able to edit and remove the tags associated with an image.

3.3.5.2 Comments and geo-spatial tags

User will be able to store, view and edit the comments and geo spatial coordinates. When the image will be retrieved from the repository these comments will be displayed with the images, then only administrator will be able to change these comments.

The use-case diagram created above shows the possible use-cases for the administrator. The administrator can register a new user if anyone wants to register with this system. The administrator will also be able to update the existing accounts. This use-case diagram shows all the possible use-cases for any user of the system.

3.6 Conclusion

The system analysis of the project has been covered in this chapter. The cope of the project has been revised for the clear understanding of the requirements, key functional and non-functional requirements have been enumerated, the use case diagram showing the major actors and their actions have been included. The sequence diagram identifying the sequence of actions taken has been incorporated as well. This chapter has been written comprehensively so that the fore coming design chapter becomes easy to comprehend.

Chapter 4

System Design

4.1 Introduction

System design is a very important phase in the software development process. The succeeding implementation phase is performed taking into consideration the design constraints. This chapter begins by presenting the high level design of the project showing the main modules of the system without including much detail. Next the low level design is incorporated elucidating the modules identified in the high level design. It is then followed by the data flow diagrams of the project. Class diagram is also included focusing on the implemented classes, their attribute and their relationships with each other.

4.2 ARCHITECTURAL DESIGN

As the requirements specifies we are to develop a web application. As our application will be used concurrently by many user and have complex application logic so we are following the 3-layered architecture. These three layers are:

A. The Data Layer: Providing access to the application data

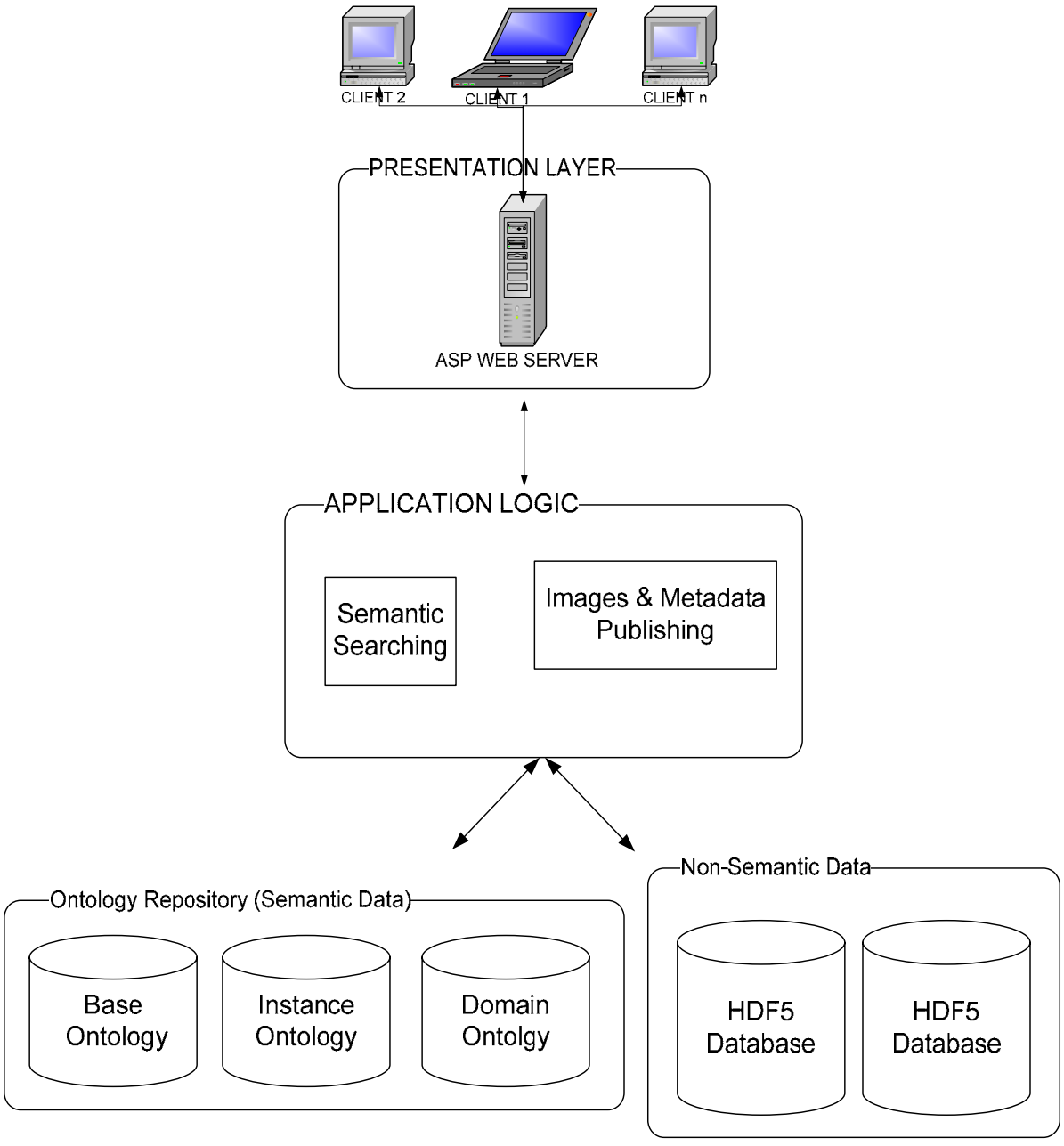
B. The Business Layer: Hosting the business logic of the application in an application server

C. The Presentation Layer: Renders the result of the request in the desired output format.

4.3 HIGH-LEVEL SUBSYSTEMS:

As it is clear from the requirements that our system can be divided into dimensions, so you can see that we have divided the application logic into two major

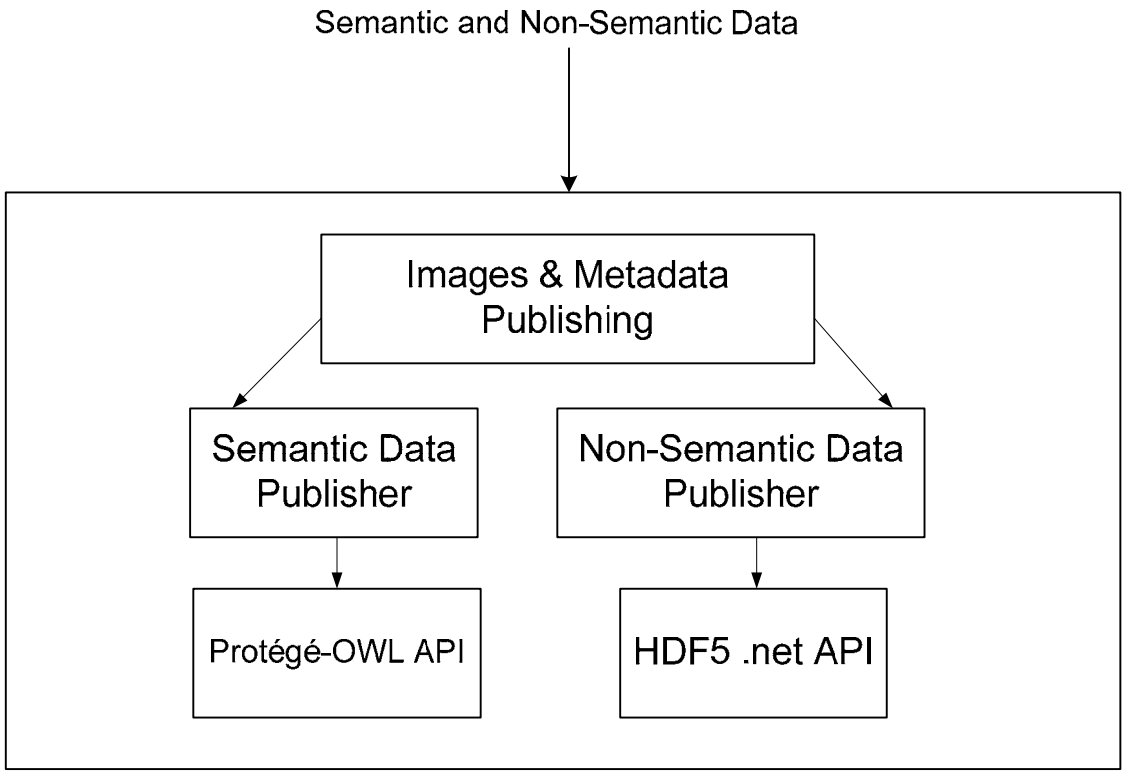
subsystems. First one involves the data storage, semantic and non-semantic data both and the second one involves intelligent semantic-based retrieval of this data.



High-Level Subsystem Decomposition

4.4 RESPONSIBILITIES OF SUBSYSTEM:

4.4.1 DATA PUBLISHING SUBSYSTEM:

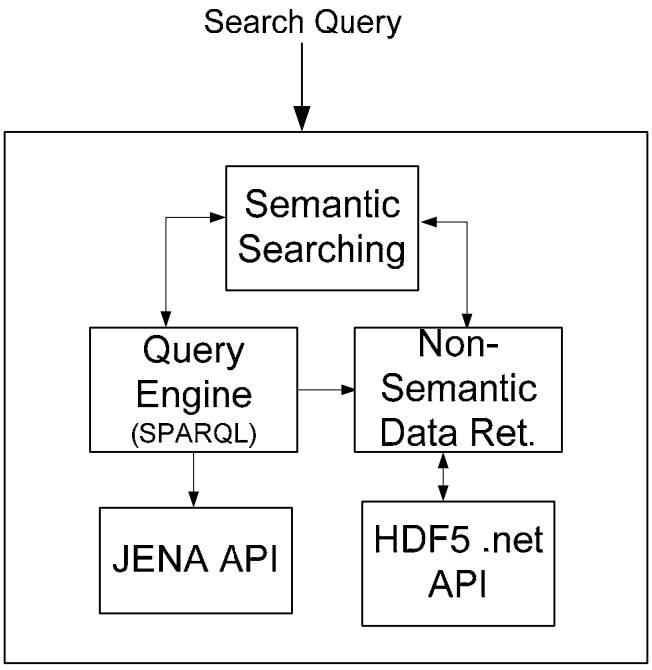


<Detailed Diagram of this subsystem>

1. It receives the non-semantic (images, maps, comments) and semantic (image annotations, markings, time etc) data after being uploaded or entered from the web-browser.
2. After receiving the non-semantic data it forwards the data to its “Non-semantic Data Handler Module”. This module by creating a new group in HDF5 database files stores the data in it as separate datasets. This module uses .net APIs developed for handling HDF5 files.

3. After receiving the semantic data it forwards the data to its “Semantic Data Publisher Module”. This module using the Protégé-OWL API creates individuals of the existing classes (present in domain ontology) in instance ontology.

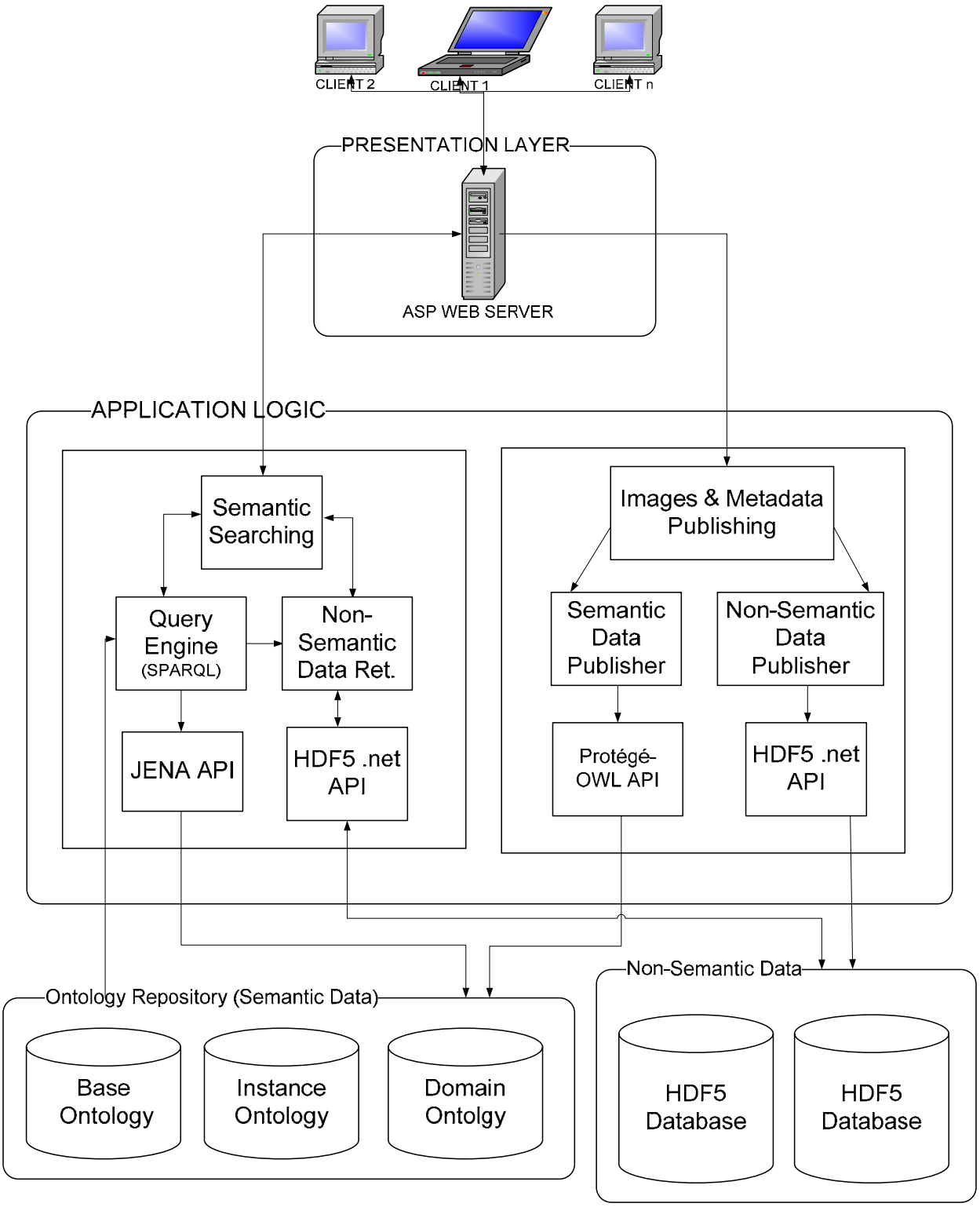
4.4.2 DATA RETRIEVAL SUBSYSTEM:



<Detailed Diagram of this subsystem>

1. It receives the search query entered by the user in the form of different attributes.
2. It concerts the search query into proper SPARQL query.
3. That SPARQL query is passed to Query Engine which executes the query on the instance ontology using the JENA ontology.
4. After getting the instances containing the same concept as specified in query, non semantic data is accessed using the .net APIs developed for handling HDF5 files for retrieving the images.

4.5 **DETAILED MODULAR ARCHITECTURE & COLLABORATION**:



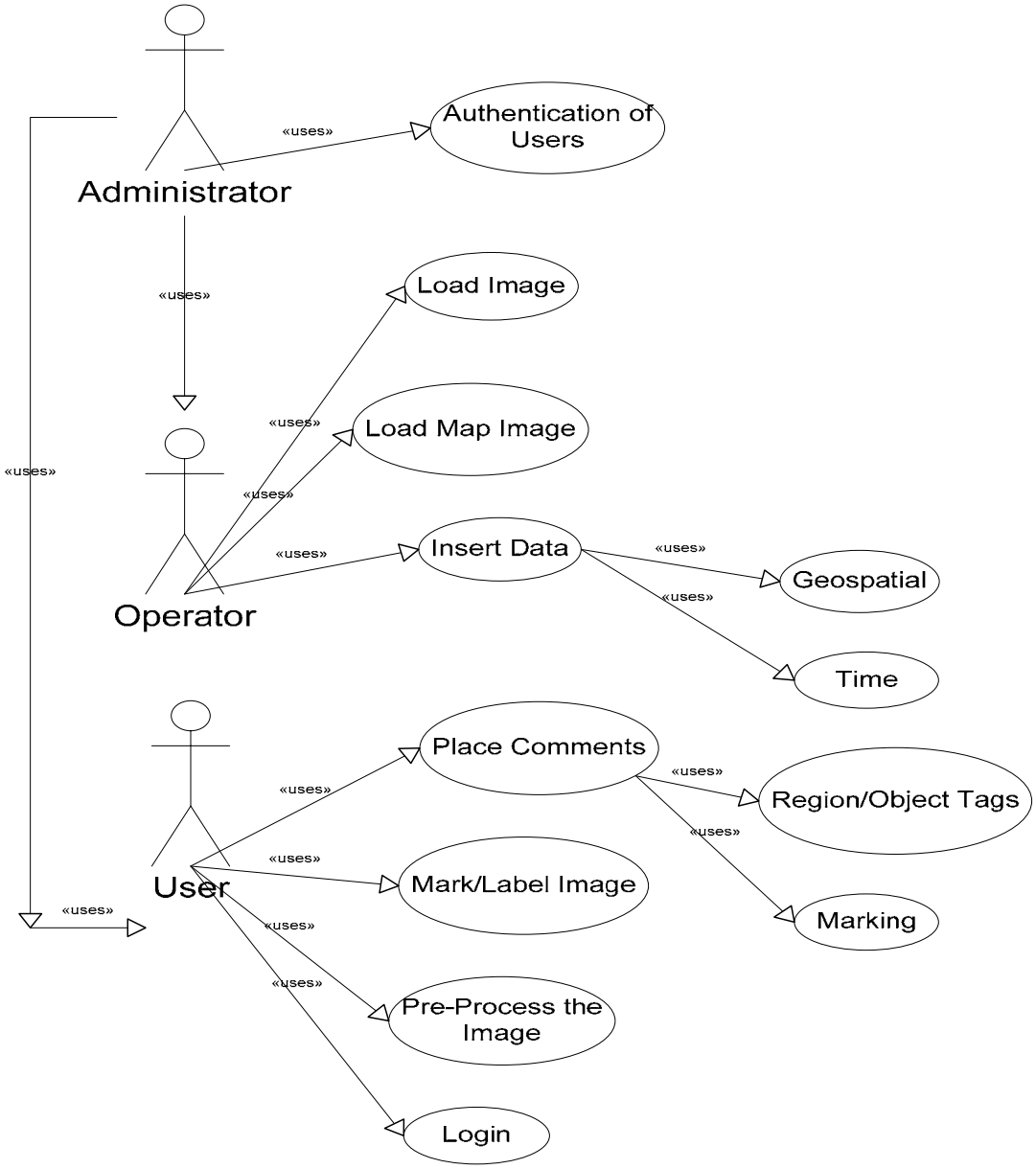
<Whole detailed architecture>

This diagram illustrates all the programmatic modules of our system and their collaboration is shown using the directed arrows. Whole of the process will be described on the basis of this diagram that how these modules collaborate to gain the overall functionality.

When user uploads an image and annotates or marks the image it comes to “Image Handler and Annotation Handler Module”. This module separates the semantic and non-semantic and sends them to “Semantic Data Publisher Module” and “Non-semantic Data Handler Module”. “Semantic Data Publisher Module” classifies the concept using the “Reasoner” which at the back uses the Rule-Base defined using SWRL. This classified concept is stored in the instance ontology using the JENA-Ontology. “Non-semantic Data Handler Module” after receiving the non-semantic creates a separate Data-Group in the HDF-5 Database files and writes each non-semantic data as separate data-set. This whole task of handling and storing in HDF5 storage is carried out by using HDF5 .net APIs.

Whenever user goes to search page, he can specify different possible attributes/criteria for his search. On pressing the search button these filtering criteria is forwarded to the “Search Module” which after receiving converts those attributes to proper SPARQL query and forwards the query to the Query Engine which executes the query on the instance ontology using the JENA ontology. After getting the instances containing the same concept as specified in query, non semantic data is accessed using the .net APIs developed for handling HDF5 files for retrieving the images.

4.6 Use Case Diagram



4.7 Sequence Diagram

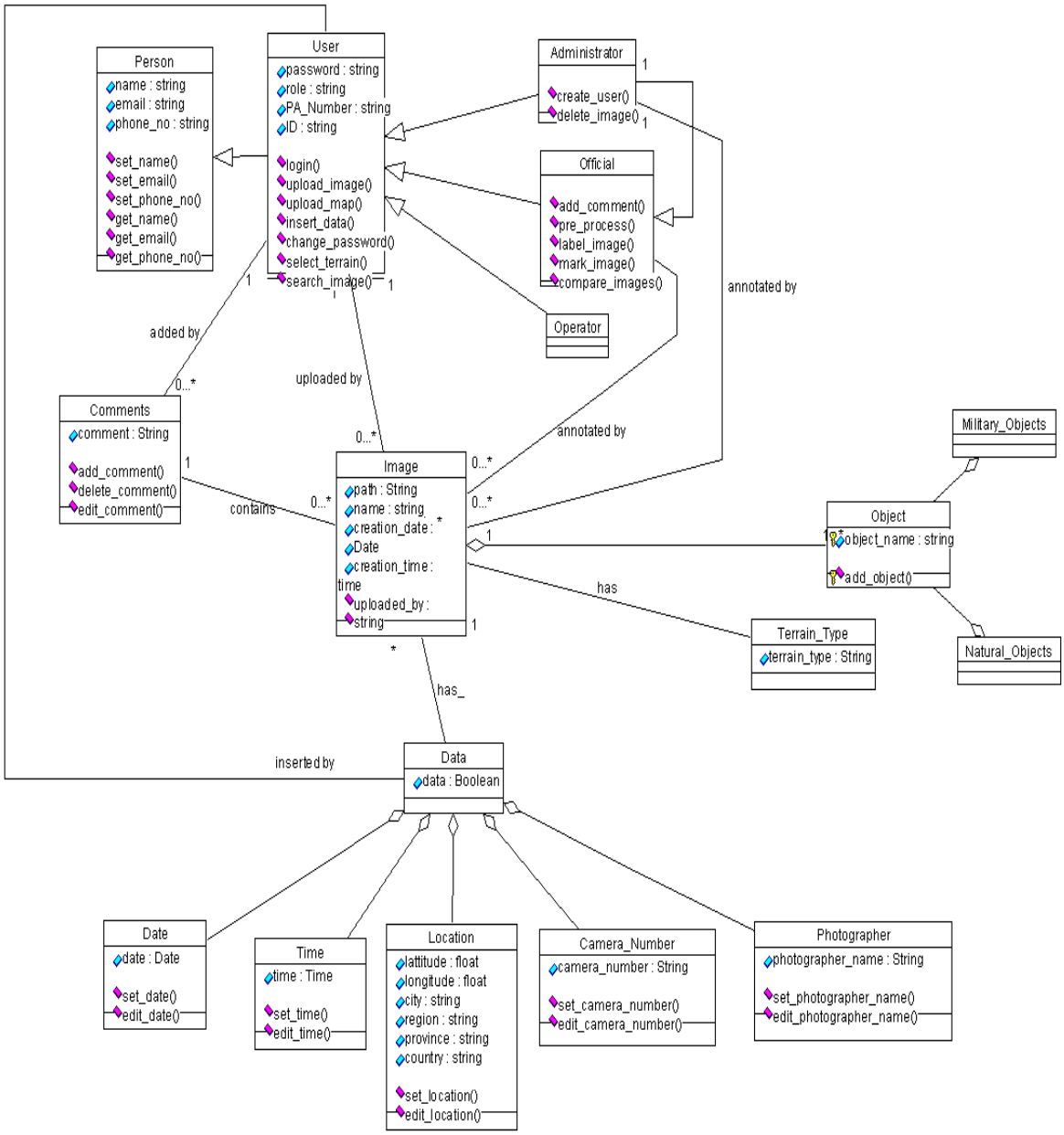
4.7.1 LOGIN

4.7.2 SEARCHING

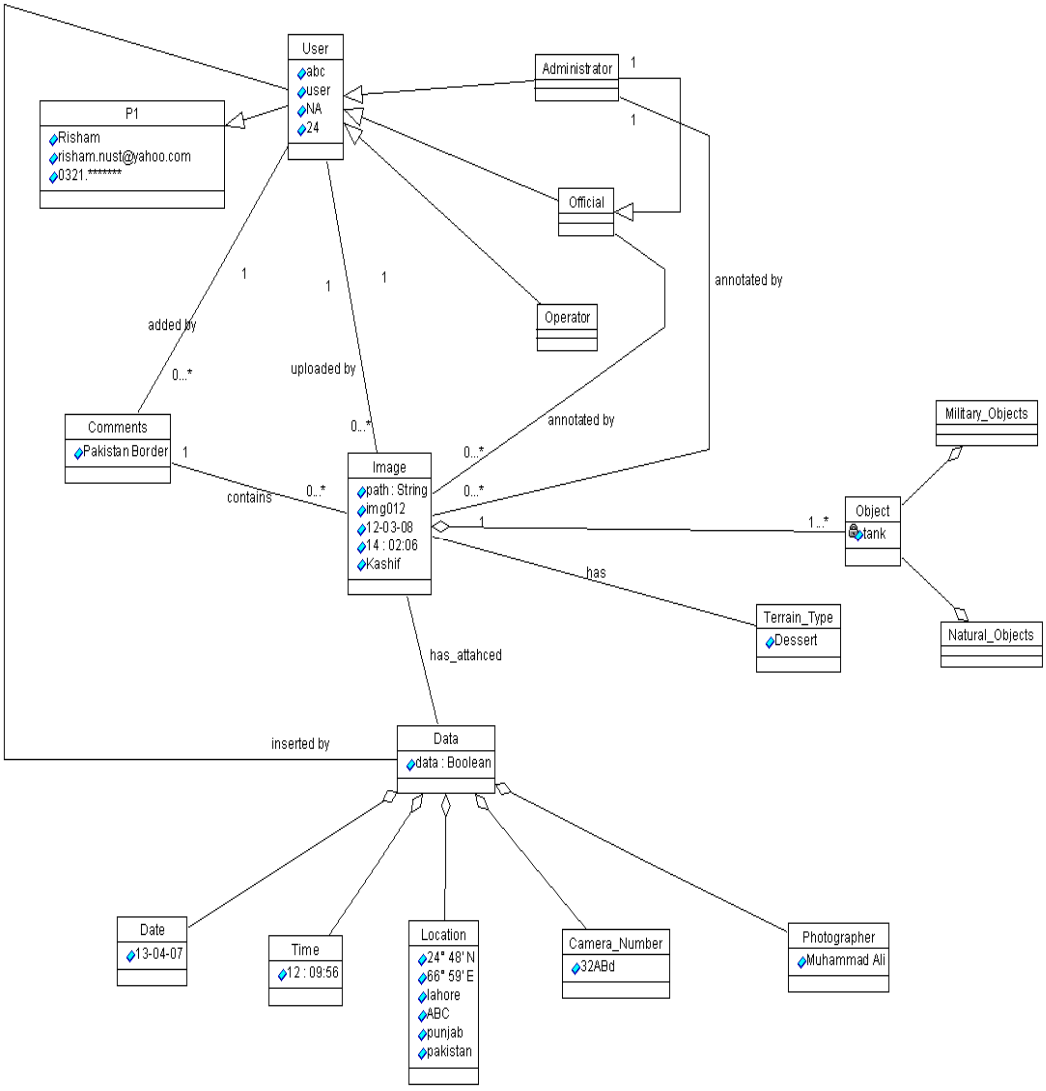
4.7.3 UPLOAD AND DATA INSERTION

4.7.3 UP**LOA**D AND DATA INSERTION

3.2.3 Class Diagram



4.8 Object Model



4.9. Data Design

We will be using HDF files for storing our non-semantic data. HDF (Hierarchical Data Format) technologies are relevant when the data challenges being faced push the limits of what can be addressed by traditional database systems, XML documents, or in-house data formats. Leveraging the powerful HDF products and the expertise of The HDF

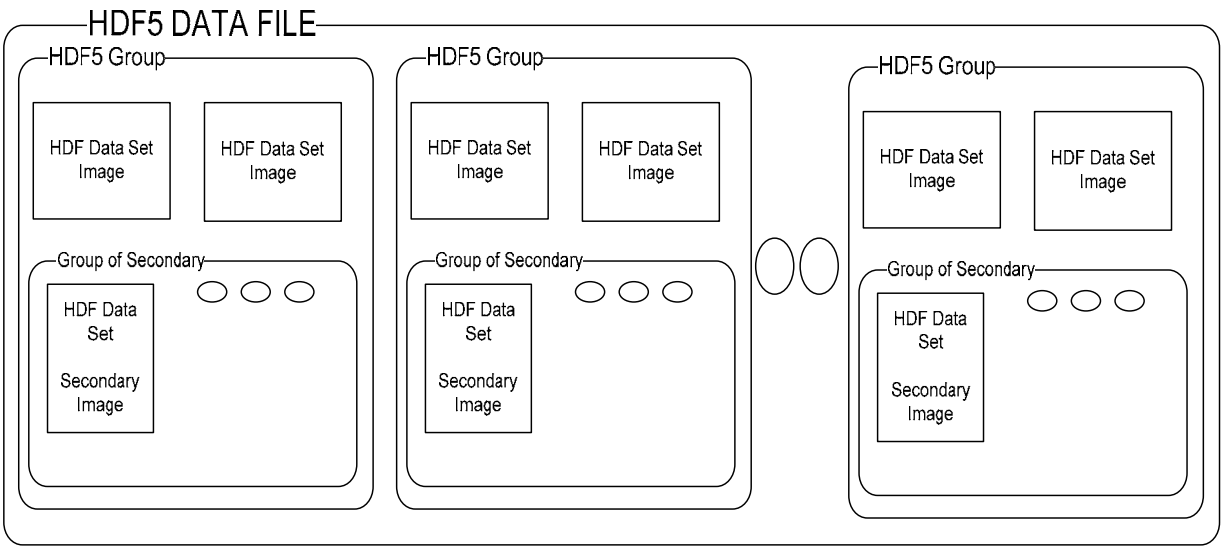
Group, organizations realize substantial cost savings while solving challenges that seemed intractable using other data management technologies.

Many HDF adopters have very large datasets, very fast access requirements, or very complex datasets. Others turn to HDF because it allows them to easily share data across a wide variety of computational platforms using applications written in different programming languages. Some use HDF to take advantage of the many open-source and commercial tools that understand HDF.

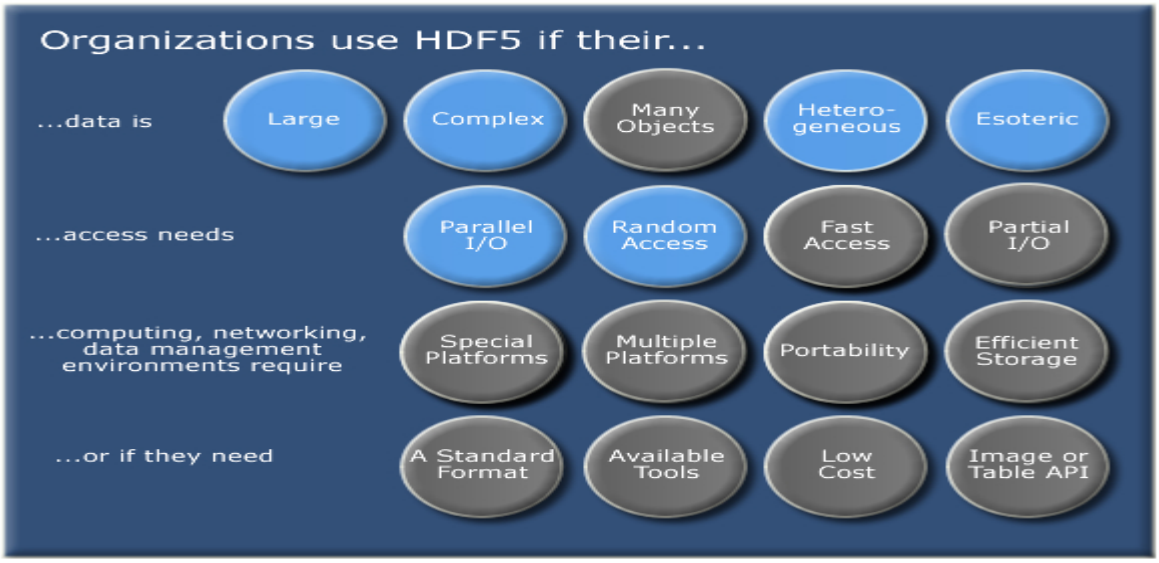
Similar to XML documents, HDF files are self-describing and allow users to specify complex data relationships and dependencies. In contrast to XML documents, HDF files can contain binary data (in many representations) and allow direct access to parts of the file without first parsing the entire contents.

HDF, not surprisingly, allows hierarchical data objects to be expressed in a very natural manner, in contrast to the tables of relational database. Whereas relational databases support tables, HDF supports n-dimensional datasets and each element in the dataset may itself be a complex object. Relational databases offer excellent support for queries based on field matching, but are not well-suited for sequentially processing all records in the database or for sub setting the data based on coordinate-style lookup.

The following will be our HDF5 file organization:



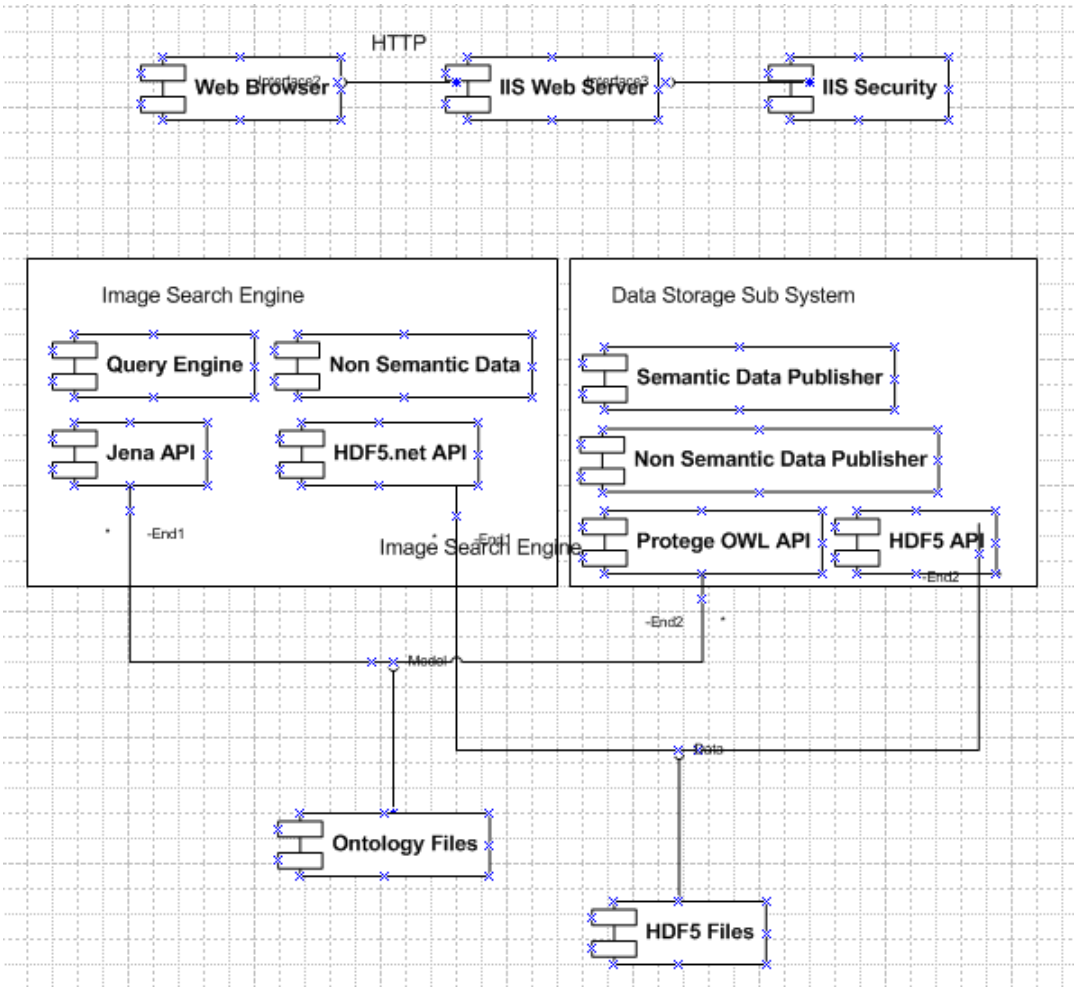
Whole HDF5 file will be divided into HDF5 groups where each group belongs to a single image. This group will contain many datasets for storing images, comments, marks, labels and further different groups for storing different secondary images obtained after preprocessing the original image.



We are using HDF5 for data-storage because:

1- It allows handling of huge data volume. A single UAV flight takes thousands of images.
2- It allows coordinate style data access efficiently, which is the prime necessity when storing layers of tags and marks.
3- It permits one to different types of data.
4- There is virtually no limit on the file size.
5- It allows runtime compression/decompression in I/O operations highly reducing the storage space required.
6- It also supports almost all of the data types especially those specified by IEEE.

4.10 Component Design



4.8 Conclusion

This chapter presented the architecture of Instant Messenger for Integrated Messaging System (IMIMS). It has incorporated the high level design, low level design, data flow diagram and class diagram of the system. Four main modules have been identified which are instant messaging, offline messaging, voice chat and file transfer.

Chapter 5

Implementation

5.1 Introduction

This chapter presents the implementation details of the project. The coding is done in C#. As illustrated in the prior chapter, there are four modules of the system, Instant messaging, Offline messaging, Voice chat and File transfer. As the system is distributed so the classes which are implemented are disseminated among these modules. The implementation chapter is hence structured in the same way i.e. mentioning each module and then elucidating the classes which are needed on that module.

5.2 Implementation Language***

The implementation language which has been chosen for the project is C#. It has been preferred over other languages because of several reasons e.g.

Chapter 6

Testing

6.1 Introduction

Testing is a very important phase in the software development process. Once the coding process is completed, then the software goes under the testing process which involves checking the codes for errors and bugs. It involves any activity aimed at evaluating n attribute or capability of a program or a system and determining that it meets it required results*. This chapter involves all the testing techniques which have been employed in the project and conclusions which have been deduced on the basis of the results of the testing procedures. Test cases for different units and components have been drafted illustrating their expected behaviors on the success and failure of each test. The output of each test is then compared with the one documented in the test case to make sure that the system behaves in the same way in which it is meant to behave.

6.2 Testing Process

The testing process has been carried out throughout the development process as an iterative approach has been used in the project for development. Each phase of development was visited several times making sure that the testing process goes in parallel with the development process. The testing was basically done at three levels, Unit testing, Integration and System testing.

6.2.1 Unit Testing

Unit testing has been done to determine that whether the individual units of the source program work in the same way in which they are expected to work. The units in the project include those methods which cannot be tested by simple inspection and those classes which cannot be broken down into smaller units for testing. The identified

units of the project along with the corresponding test cases are illustrated under the following headings.

6.2.2 Component Testing

Different units together form a component. After unit testing, the components have been tested to make sure that they behave in the expected way. The test cases for different components of the system are elucidated and shown under the following headings.

6.2.3 Integration Testing

Integration testing means testing the functionality of the system stepwise while integration the components or modules. While amalgamating the components, tests are carried out each time the components are integrated. If the tests are successful, then further integration of the system takes place. Otherwise the components are debugged and integrated again and again until the tests are successful.

6.2.4 White Box Testing

White box testing or structural testing uses an internal perspective of the system to design test cases based on internal structure. It requires programming skills to identify all paths through the software**. The white box testing of the system has been done at both unit testing and component testing stages.

6.2.5 Black Box Testing

Black Box Testing is testing without knowledge of the internal workings of the item being tested**. It attempts to derive sets of inputs that will fully exercise all the functional requirements of a system. For each set of inputs, outputs are known and in black box testing, the inputs are fed in and if the output matches the predicted output it means that the system delivers the expected functionality.

6.2.6 Static Analysis of Code

Besides testing the code dynamically, static analysis of the code has been done as well to find defects, if any, in the blocks of code due to which it does not implement the exact requirement or to determine the ways by which the code can be optimized to make it full proof

The code has been statically analyzed in many ways which are briefly illustrated under following headings.

6.2.6.1 Control Flow Analysis

Control flow analysis has been carried out for the verification and validation of control blocks in the source code, for instance, the 'for', 'while' loops and the 'if' condition blocks. It has been observed that no unnecessary code has been included and all these blocks are optimized.

6.2.6.2 Data Analysis

Data analysis has been done to find and remove improper initializations, unnecessary assignments and those variables that are declared but never used. All such unnecessary lines have been eliminated thus giving a refine code.

6.2.6.3 Interface Analysis

Interface analysis has also been done to insure consistency of interface, class, procedure declaration, definition and their use. It has also been observed through test that all the method declared in the interface is correctly implemented in the classes and that there are no redundant methods.

6.2.7 Conclusion

This chapter illustrated the testing process of the system that has been carried out and the corresponding results obtained. The testing of a system has been done in complete detail. The test cases have been returned for the three main phases of the testing, unit testing, and component testing integration testing. Using these test cases, results of the test cases have been authenticated. Both white box and black box testing have been carried out to determine that whether the system delivers all the functional requirements that it should be delivering. Even static inspection of the code has been carried out as well so that it becomes optimized and does not become redundant. All the test results were very successful proving that system delivers all its functionalities in an efficient way.

APPENDIX

User Manual

User Manual

This application software has three categories of users.

1. Administrators.
2. Officials.
3. Staff

We have discussed how to use the system easily and efficiently. Manual includes the detailed guideline for each page of website.

1. Login Page

Once the program is opened, the user will be ready to begin using the messenger. The initial interface of the application will look like as follows: