Semantic desktop

focusing on harvesting domain specific information in planning aid documents

Final thesis in Computer and Information Science at Linköping Institute of Technology

By

Ali Reza Etezadi

LIU-IDA/LITH-EX-A--08/007--SE

Linköping 2008
Semantic desktop

focusing on harvesting domain specific information in planning aid documents

Final thesis in Computer and Information Science at Linköping Institute of Technology

By

Ali Reza Etezadi

LIU-IDA/LITH-EX-A--08/007--SE

Linköping 2008

Examiner: Henrik Eriksson
IDA, Linköping University

Supervisor: Ola Leifler
IDA, Linköping University
Semantic desktop focusing on harvesting domain specific information in planning aid documents

Ali Reza Etezadi

Planning is indeed a highly regulated procedure at the operational level such as military related activities where the staff may benefit from documents such as guidelines that regulate the work process, responsibilities and results of such planning activities. This thesis proposes a method for analyzing office documents that make up an operational order according to document ontology. With the semantic desktops aiming at combining semantic annotations and intelligent reasoning in desktop computers, the product of this project intends to add a plug-in to such environments such as IRIS semantic desktop, which accordingly enables such application to interpret documents whether the they or change within the application.

The result of our work helps the end user to extract data using his/her favorite patterns such as goals, targets or even milestones that make up decisive points. This information eventually form semantic objects, which ultimately reside in the knowledgebase of the semantic desktop for further reasoning in the future referring of the application, whether automatically or upon the user's request.

Semantic Desktop, IRIS, Planning Aid, ODF4J, Xpath, Ontology, DOM, Knowledge Base
Acknowledgements

To my academic staff:

I like to start by thanking Prof. Henrik Eriksson for trusting me and providing me with the opportunity of putting my hands on such genuine thesis project.

I also would like to extend my appreciation to Ola Leifler who supervised me patiently through all the long period that the project was going on.

Jalal Maleki, you are unforgettable. Without your attention and kind advices, time would have stretched for me forever.

Special thanks to Siv Soderlund for her non-stopping support and great advices, which kept our hearts warm during the long period of studying in Sweden.

In addition, I would like to appreciate prof. Kahani for the motivation and the endless support that he always offered me.

To those who inspired me:

Studying in Sweden provided me with a rare opportunity of knowing many people who each played their role in my life and their memory will always stay with me forever. I would like to enlist them as a way of appreciation in here: Behzad, David, Amir Farhadi, Amir Eghbali, Amin Ojani, and Azadeh. Thank you all for the great time you provided me. I also love to remember all my friends at the corridor Bjornkarrsgatan 4B in Ryd Linkoping for the friendship they offered me.
To my family:

Words are not capable of bearing the true burden and meaning of the gratitude I feel deep within my heart and soul for my beloved ones.

I would like to express my appreciation to my wife for her support during the long period of my studying in Sweden.

How can I possibly thank my parents enough for what they have done? All this time I always felt my wife and kid were fully safe with your non-stopping presence beside them.

In addition, I would like to express my gratitude to my brothers and sister for being such wonderful friends for me. My friend Mohammad Forouzangohar also is apprciated in here as a brother for the type of love and support that he dedicated to me.

The finally yet importantly, my son Arya, I love you! You were two years old when I left you and now you are more than three years of age! However, you were the big reason and remained the only courage for all of the family to take the challenge. Thank you for the positive energy you injected into us. I will never leave you again.

To Sweden:

With great power comes greater responsibility. Even a short step from a big nation will lead to greater steps by other nations. I must appreciate to Sweden for the opportunity they awarded me for which I will always feel devoted. In addition, my warmest wishes stay for Linkoping University for the quality education that they provided me. I will never forget the experience in my lifetime.
“I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.”

Alan Turing, Computing Machinery and Intelligence 1950
# Table of Contents

Acknowledgements ............................................................................................................................... vi

Abstract ................................................................................................................................................ 1

Introduction ........................................................................................................................................ 3

1.1 Motivation ................................................................................................................................... 3

1.2 Problem definition ....................................................................................................................... 4

1.3 Outline of the report .................................................................................................................... 5

Background ........................................................................................................................................ 7

2.1 Semantic Web: ............................................................................................................................. 7

2.2 RDF: ............................................................................................................................................ 12

2.3 Ontology: .................................................................................................................................... 13

2.4 Levels of representation ............................................................................................................. 15

Semantic desktops: IRIS ....................................................................................................................... 16

3.1 Semantic Desktops ...................................................................................................................... 16

3.2 IRIS Developing Environment .................................................................................................. 27

Previous works, current attempt ........................................................................................................ 30

Methodology ....................................................................................................................................... 32

4.1 Foreword ..................................................................................................................................... 32

4.2 Planning for an Office system: .................................................................................................. 33


**Table of Figures**

Figure 1 the smart data continuum [5] ............................................................................................................. 8

Figure 2 Ontology Representation Levels [5]........................................................................................................ 15

Figure 3 IRIS integration framework of three layers [9].................................................................................... 22

Figure 4 IRIS User Interface [9]....................................................................................................................... 22

Figure 5 CALO suggest pane on the right side [9]............................................................................................ 26

Figure 6 the schema of the bigger project [7].................................................................................................... 30

Figure 7: A view of the simple intended GUI..................................................................................................... 38

Figure 8: The output for the results ................................................................................................................... 53
Abstract

Planning is indeed a highly regulated procedure at the operational level such as military related activities where the staff may benefit from documents such as guidelines that regulate the work process, responsibilities and results of such planning activities.

This thesis proposes a method for analyzing office documents that make up an operational order according to document ontology. With the semantic desktops aiming at combining semantic annotations and intelligent reasoning in desktop computers, the product of this project intends to add a plug-in to such environments such as IRIS semantic desktop, which accordingly enables such application to interpret documents whether the user or the system add them or they change within the application.

The result of our work helps the end user to extract data using his/her favorite patterns such as goals, targets or even milestones that make up decisive points. This information eventually form semantic objects, which ultimately reside in the knowledgebase of the semantic desktop for further reasoning in the future referring of the application, whether automatically or upon the user's request.

Accordingly, it turns out to be very important that there exists a method of parsing, processing and storing data in such domain-specific environments. Regarding the time and the
resources, the need for finding the precise information and eventually avoiding dealing with unnecessary data is very critical.

The original plan was to design a plug-in that would integrate with the services of IRIS semantic desktop and be capable of presenting a service that would respond to the events of the system intelligently. However, we designed a stand-alone application, which still integrates with IRIS semantic desktop, extracts the information and eventually stores them in the knowledgebase. However, because of such procedure, the semantic desktop becomes capable of using that information in its current events such as analyzing emails or calendar events that semantically refer to this information.
Chapter 1

Introduction

1.1 Motivation
This thesis suggests a method for harvesting information in domain-specific plan documents that exist in planning areas such as military planning. Harvesting is the act of importing external data into ontology\(^1\)-based (semantic\(^2\)) structures. A set of plan documents relate to a desired goal state or desired plan actions in order to reach that goal. This information plays an important role in automated planning procedures.

Such solution helps the intended user to extract his/her desired data out of documents and store them in a special data storage for further analysis and decision making in the future through semantic procedures. It is very

\(^1\) Ontology is a hierarchical structure of knowledge about things by subcategorizing them according to their essentials. It is a collection of facts and information. Ontology is an explicit specification of a conceptualization.

\(^2\) According to Cambridge Advanced Learner’s Dictionary, the word semantic means “Connected with the meanings of words”.
important that there exists a method of parsing, processing and storing data in domain-specific documents because at the age of information and the crisis of information explosion, the need for finding the precise information and eventually avoiding dealing with unnecessary information is very critical.

The use of IRIS Semantic Desktop as the running environment of our application plug-in integrates our intended capabilities with those of its other independent plug-ins and other various semantic services that work tightly together in this application as a desktop solution.

1.2 Problem definition
Users store and retrieve data from various data formats. In formal documents, there are rules and guidelines to follow. Such documents travel among different environments such as emails or word-processing applications. A semantic desktop ties and integrates the outside world of the Web to the one of the desktop computers where users have the most interaction with their own data and applications. Semantic mechanisms provide more facilities for the machines and the humans to interact more efficiently and equally over the data either on the Web or the desktop
computers. In such way, if a user tries to locate a piece of information in a planning aid document according to a certain pattern in a specific domain of information, he/she must be able to do so by entering the intended keyword and selecting the desired structure in the document to look in. Behind the scene, there should be means of obtaining the inputs from the user, generating the appropriate pattern for locating and extracting the information and storing the data for further analysis in the future. Because this information is stored in a semantic data storage, more reasoning will happen on the data upon the users’ request at any time. Plain data harvesting methods have already existed but none has been available for the specific needs of data harvesting from popular formats of documents with a specified domain. In order to do so there shall be means of analyzing certain file formats such as Microsoft Word or Open Office documents at the structural level. In fact, the thesis aims at answering these questions.

1.3 Outline of the report
The first part of this thesis briefly explains the concepts that exist in semantic area of information technology. More will come on the idea of semantic desktops: what they are and why we need them. Moreover, you will read about the types of semantic desktops that currently exist on the
research table and the reasons that we go for IRIS semantic desktop in this thesis. Planning aid documents and their characteristics come next and the way this thesis aims at connecting them to semantic desktop.

Next, we discuss the area of working and you will read about any previous work related to such field of research. Then, you learn about the importance of carrying out this project.

Methodology of the current thesis appears in detail in the upcoming chapter and subsequently the result of such research comes for you to see. You also learn more about the kinds of documents that we tested and the reasons why we selected a specific kind of Office document. Java was the developing platform for the thesis. Accordingly, we had to apply different libraries and packages to reach our goal. Those libraries and techniques come into view.
Chapter 2

Background

2.1 Semantic Web:
According to “Cambridge Advanced Learner’s Dictionary©”, the word semantic means “Connected with the meanings of words”. Today’s web information entities are using advanced methods of indexing and case-sensitive searching and relating of data such as state-of-the-art Google search engine which uses complicated searching methodologies mainly benefiting the indexed data or most referenced data over the web in order to provide the user with the most precise and highly referenced pages available on the world wide web. Although such efforts are highly pushing the humans’ lives forward in virtual fields of information by being highly precise and reliable, still there is no element of meaning involved in them. They can not exactly relate discretely located data, which relate to a specific meaning in order to appear a whole. There has been already two generations of Web in our hands, which we refer to as "Web 1" and "Web
2". With the start of Web 1, people learned to share and access data in a wide range while by using Web 2, they found means of interacting even more efficiently with the websites. Therefore, the Semantic Web is not a separate web but an extension of the current one, in which information relates to well-defined meaning, better enabling computers and people to work in cooperation. In order to help the semantic web to function, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning[1].

![Figure 1: The Smart Data Continuum](image)
In order for the Semantic Web to operate, the machines must be able to access the collection of information and sets of deduction rules that eventually result them in being capable of conducting an automated reasoning. In fact, the semantic web enables machines to comprehend semantic documents and data [1].

Semantic Web intends to expand the meaning of data that appears on existing and future documents on the web. Such effort provides the web with even more potential to share the data between human and machine resources with equal ability of processing the semantics of the data among broader ranges of communities.

Such environment has two major benefits:

1. The distinction between real useful data on a document and other unrelated data such as ads becomes clearer.

2. Different sets of data appear in the form of describing elements that machines can also interpret as well as humans in a standard form. In such way, even if data are from different sources, still they can integrate with each other so that if one encounters an instance of
such information and it is semantically capable of relating to other sets of data, the machines can decide on the relating procedures.

Many areas apply this concept such as *Resource discovery and classification* in which more effective domain specific search engines emerge to revolutionize searching over the Internet. It also causes the integration of data resided in different locations presented in different formats and treats them as a whole. In online shopping and electronic catalogues and brochures, semantic web may play a major role in facilitating the activities with content description and any possible relationships among them. Also in digital libraries, by means of applying intelligent software agents, knowledge sharing and exchanging gets even more convenient. In addition, another important field may be for describing intellectual property rights of existing web pages or documents[10].

Tim Berners-Lee presented a vision for the future of the web that consists of two categories. In the first place, Web intends to be a more concerted media. Secondly, machines must be able to process the web, as well as humans. In the original vision, the Web appears more than just HTML

---

3 Hyper Text Markup Language: The publishing language used by the World Wide Web
pages retrieved from Web servers. According to that vision, relations between web entities define terms like “including”, describing” or “writing”. Yet, today’s existing Web does not currently cover such concepts. Therefore, a useful technology named RDF\(^4\) captures such shortage. The conclusion is that Lee’s futuristic vision added another level of Meta data onto the current layers of the working Web. The machines use such Meta data to process the target information on the Web.

The Question that rises would be “how shall we produce a web of information which the machines can process?” One first step might be that there should be a change in the way we think of data. Traditionally we think of data on the web as information chunks owned by specific applications. Processing the data has been prior to the data itself and it emphasized the importance of data. Accordingly, it turned out that data should be both verified and protected. This fact resulted in features as in object-oriented programming where data turns out to be of much importance. However, this trend led to the data kept proprietary to the vendors’ applications for

---

\(^4\) Resource Description Framework: A language that represents information about resources in World Wide Web
competitive reasons. With the developing of XML\textsuperscript{5} and recently Semantic Web, the shift in the importance of data over applications in a moderate manner is happening. In order to make the data more comprehensible for the machines, the data itself should look smarter in nature (Figure 1)\cite{5}.

2.2 RDF:
RDF or \textit{Resource Description Framework} is a metadata model. In order for the machines to understand web documents, they need certain tools. RDF deals with meaning that is in form of triples that benefit from the XML tags form. RDF declaration of the triples is in form of (\textit{particular things, their properties, certain values}). An example of such kind may be ("Lena", "Student of", "Linköping University"). To identify the subjects and objects we can benefit from URI (Universal Resource Identifier). URI actually enables the encoding of the information in documents and therefore it is correct to say that RDF triples develop future webs of information about related things. An example of an xml presentation of an RDF comes as follows\cite{2}.

Assume that the sentence is:

\textit{“Martin is a student of Linköping University”}

\textsuperscript{5} Extensible Markup Language: Simple and flexible text format originally meant for large-scale publishing over the Web and in the exchange of wide variety of data on the web among its numerous different resources
This sentence may seem in RDF/XML as:

```xml
<rdf:RDF>
  <rdf:Description about="Linkoping University">
    <s:Student>Martin</s:student>
  </rdf:Description>
</rdf:RDF>
```

2.3 Ontology:

Ontology is a hierarchical structure of knowledge about things by subcategorizing them according to their essentials. It is a collection of facts and information. Ontology is an explicit specification of a conceptualization. The zip code of Linköping University is 58183. However, many other residents or organizations may share this code at the same time. In this case, different databases may apply different indicators for the very same concept. Now suppose that a program intends to work with such information that at least two different database entities hold and present. This program must be equipped with a mechanism to learn that the two different data actually indicate one single concept. This is where another component of the semantic web seems to be of much help and it is Ontology. The term is coming from philosophy. However, in Knowledge-based systems, any existence is presentable. When the facts within a domain illustrate a declarative form, a set of objects reveals namely the universe of discourse. Once the set of objects and the describable
relationships among them is in hand, they bear the form of a representational vocabulary. The knowledge-based programs denote knowledge by relying on such vocabulary. Using such ontology, the names of entities in the universe of discourse through the definitions can be associated with human-readable texts that explain what the names stand for. Besides, there is formal maxims which restrain the explanation of these terms and their well-formed way of usage[3].

In order to define and illustrate web ontology, some kind of tool is required. That tool is *Web Ontology Language (OWL)*. The abbreviation stems from Owl in Winnie the Pooh (Milne, 1996) who spells his name as “W O L”! [5] Any ontology founded in such way contains descriptions of classes and their properties and instances. Applications that require more than just displaying the information to the user, but also to process the content of that information benefit from such structure. There are two major advantages for OWL over technologies such as XML or RDF: It provides additional vocabulary along with formal semantics. OWL tries to provide a common way of processing the semantic side of the web [4].
2.4 Levels of representation

Ontologies are like Languages or syntactic vocabularies with conveysing semantics. Furthermore, Ontologies are content. Contents can only be expressed when a content language is ready which is also addressed as "Knowledge Representation Language". The term Meta level usually refers to language level of knowledge representation. Another level at which the Ontologies are stated exists which is called the ontology concept level. A third level is also required in order to show instances of the existing Ontologies. You may view the instance level as object level while the upper level or the Meta level is holding class or the universal knowledge. Below, you may study the levels of ontology representation.

![Ontology Representation Levels](image-url)
Chapter 3

Semantic desktops: IRIS

3.1 Semantic Desktops

The semantic web maintains promises for organizing the information and selective access to them, which eventually result in offer standard means for formulating and distributing metadata\(^6\) and ontology. However, personal computers do not benefit from such concepts yet. Using Ontology, metadata and semantic web protocols result in even more meaningful integration of desktop environments and the web. This results in even more concentrated management of personal information as well as more information distribution and collaboration on the web, further than email [17]. This idea has been around for quite some time. Futurists such as Englebart and Vannevar Bush devised and realized this idea to some extent. Englebart first introduced and demonstrated such a system called

---

\(^6\) Metadata: generally referred to data about data in computer science
NLS\textsuperscript{7} in 1968 at SRI\textsuperscript{8}. This idea dates back to the WW2 when Dr. Englebart was attracted to the original idea by Vannevar Bush when he was in Philippines \cite{12}. Yet, nobody fully implemented those ideas in the real world because they had to wait for the technology to develop in the way that the necessary infrastructure emerges in order to let those ideas to exist. Only recently, the computer science community has managed to develop initial means and infrastructures that turn those ideas into reality.

Today, there are a number of projects aiming at providing us with the capability of semantic interpretation of desktop environments for the machines and those are:

- **Haystack System at MIT**: Haystack is a project at the Massachusetts Institute of Technology to research and develop several applications around personal information management and the Semantic Web

\footnotesize{\textsuperscript{7} oNLine System (NLS) is an instrument for helping humans operate within the domain of complex information structures, Doug Englebart, Stanford University. Available at http://sloan.stanford.edu/mousesite/1968Demo.html (9/1/2008)}

\footnotesize{\textsuperscript{8} Stanford research Institute (SRI)}
• **GnowSYS**: Gnowledge Networking and Organizing SYStem is an application for developing and maintaining semantic web content and is originally designed using Python

• **Chandler**: a free Personal Information Manager written in Python developed by Open Source Application Foundation (OSAF)

• **Open IRIS**: Ontology based semantic desktop application developed by Stanford Research Institute (SRI). It is consisted of many independent ontology-driven plug-in that provides services through Ontological descriptions. Some of these plug-in provide GUI services and some others gather semantic information from the IRIS working environment. When used as a desktop environment for a user, it offers both embedded applications and stand-alone. The embedded applications within IRIS closely join with the other applications and are capable of responding to application events instantly such as opening an email conversation with or without an attachment. IRIS benefits from SEMEX\(^9\) in order to gather information.

---

\(^9\) SEMantic EXplorer is an open architecture for information extraction from different document types, to extract the contents of various text documents in a workspace\([7]\).
Before focusing more on IRIS project, we take a quick look at CALO project [11]. This is a project led by SRI International funded by DARPA\textsuperscript{10} under its program called PAL\textsuperscript{11}. The goal of CALO project is to develop cognitive software systems that are capable of

- Reasoning
- Learning from Experiences
- Performing what they are told to
- Explaining what they are doing
- Consider their experiences
- React toward surprises

To achieve that, CALO should be equipped with a system that provides a semantically consistent view of the user's life and the ability to interact with the user in a natural way. IRIS turns out to be the solution to those areas.

\textsuperscript{10} The Defense Advanced Research Projects Agency (DARPA) is the central research and development organization for the Department of Defense (DoD). It manages and directs selected basic and applied research and development projects for DoD, and pursues research and technology where risk and payoff are both very high and where success may provide dramatic advances for traditional military roles and missions. Available at http://www.darpa.mil/ (2008/01/17)

\textsuperscript{11} Personalized Assistant that Learns
within CALO project and enables them to sketch out the key elements. CALO acts as cooperative partner in this project by learning how to inhibit much of the semantic content and the relationships.

There are number of characteristics that make IRIS a unique platform as a semantic desktop environment [8]:

- **Real enough** to convince people rely on it and let it do their daily work
- The ability to *integrate* and/or *being implemented in Java* while CALO, as a core of knowledge manipulation in IRIS is already developed in Java
- Ontology-based knowledge store, which has the ability and flexibility of capturing every aspect of the user’s work environment
- Capable of supporting organization of personal knowledge assets for the users in the way that suits them most while preserving semantic interpretability with other CALO installations.
- Cross-platform in the sense that it is usable on many operating systems such as MS Windows, Mac OS or Linux

This is the point where IRIS is born. This application framework enables users to provide a “personal map” across their office-related information objects. IRIS stands for “Integrate, Relate, Infer, and Share”.

20
• **Integrate** - IRIS integrates data from different sources by using reified semantic classes and typed relations[8]. As an example, the following sentence should be possible to express even if different third party vendors develop each discrete responsible application: “Thesis X is presented at meeting M by Tom, who is the thesis student of thesis X.”

The integration services within IRIS framework consist of three layers (Figure 3).

1. IRIS should be able to access information resources (e.g. email message, calendar appointment) and the applications that generate or operate on them. Despite a very light-weighted kernel, all the functionalities within IRIS is presented in form of plug-in framework consisting of:

1.1. User Interface (UI)

1.2. Applications

1.3. Back-end persistence store

1.4. Learning modules

1.5. Harvesters, and so on
2. Knowledge Base offers unification of data models, a persistent store and mechanisms for querying across the information assets and their semantic relations.

3. The IRIS user interface plug-in enables applications to embed their own user interface into IRIS and universal UI services.

Figure 3 IRIS integration framework of three layers [8]

Figure 4 IRIS User Interface [8]
• **Relate** - IRIS integrates the tools of knowledge work semantically. This means providing a knowledge representation by which relics of user's experience (e.g. email conversations, calendar events or even the files stored on the local machine’s hard disk) can be stored and related to each other across other users’ or applications’ entities.

To achieve that, IRIS has benefited from OWL, recommended by W3C. It offers flexibility in designing the schemas.

There are services designed within the framework to ease access, improve maintenance, and interoperate with data structures from Java, which are OWL-based. For each class in the ontology, IRIS SDK creates a POJO in the compile time. POJO is abbreviation for either “Plain Old Java Object” or “Programmable Ontology Java Object”. POJOs represent references to other POJOs through property relations in order to provide with a complete API for accessing Ontology. Think of IEmailMessagePojo, for example. For accessing header, it contains a method (IEmailMessageHeaderPojo). The same is true for accessing body (IMessageBodyPojo), addresses (IMessageAddressPojo) and so on. Methods generated on the POJO communicate directly with the properties on the ontology class and
include “binding paths”. Binding paths are yet another way of reducing the overhead of accessing a complex ontology. They also appear as methods on the POJOs in order to make the programming API easy.

In addition, IRIS provides a framework for harvesting application data and incrementing user actions in IRIS applications. Harvesting is the act of importing external data into ontology-based (semantic) structures.

- **Infer** - IRIS insists on machine learning and the implementation of a plug-and-play learning framework [8]. IRIS understands the problem of limiting the semantic web’s growth and mass adoption by benefiting from machine learning. In such way, there exists an automated way of entering all of the required links and knowledge. For example, in harvesting data from an email activity, the following steps are involved in order to construct a semantic representation of the user’s work:

  - **Step 1: Email Harvesting**: As the user receives an email, IRIS automatically harvests messages, gathering and adding them to the knowledge base (KB). First, normalization of the names in
the address fields occurs. Then there are links to the current existing records in the KB and new records enter the knowledge base for those people who do not exist in the storage. The system then advertises the events that indicate the new email messages and the new contacts and people added to the knowledge base in the Instrumentation BUS for the other applications to learn about them.

- **Step 2: Contact Discovery:** When records about a contact or person is added to the KB, a service called DEX\(^{12}\) triggers and tries to find yet more information about that person from the domain that is reflecting his email address.

- **Step 3: Learn from File:** Just like an email, the service harvests the files existing on a user’s desktop, as well. SEMEX is the technology responsible for this, which currently opens LaTeX, MS Office files to extract facts about people and add or update to the KB.

\(^{12}\) We use DEX to perform analysis on data. In IRIS, it finds contact data for a person. The property that corresponds to the found data will be updated only if the current property has not been set [18]
• **Step 4: Project Creation:** For each project, the application applies a label, using the most significant words. It also creates links for the participants involved in the project.

• **Step 5: Classification according to project:** The application also applies a classifier to the contents and relations extracted before in order to put forward the relations between the projects and other objects such as emails, web pages and so on. IRIS displays the suggestions to the user and he/she is able to interact with the algorithm actively by indicating correct values.

![Figure 5 CALO suggest pane on the right side [8]](image-url)
• **Step 6: Higher-level Reasoning:** plenty of specialized reasoning entities within CALO examine the user’s activity stream without any delay. For example, if the user clicks on an email message, the system tries to predict whether the user replies to the message or forwards it. Another plug-in may summarize the text of the email for faster future reading of the user to be faster.

• **Share** - Shared structures are vital to both end-user applications and infrastructural components such as machine learning algorithms.

### 3.2 IRIS Developing Environment

Although we talked about IRIS before, here comes a developer’s perspective. Iris is a desktop application with a database and a number of built-in applications and services. Generally, IRIS main components are:

- User Interface
- Applications
- Database

The central part of IRIS is its *kernel*, which binds these components to each other and allows other applications called *plug-ins* to be developed
later by developers and eventually added to the overall functionality of the system. The architecture of IRIS services consists of three major parts:

- IRIS kernel
- IRIS GUI
- IRIS Backside

In order to build and install a plug-in within IRIS, there are key services, which are of much importance to the developers:

- IObjectStore: Using this service the developer is able to create and query semantic objects
- IUser: This service makes the information about the logged in user available.
- IViewerHost: This service provides the developer with the access to the user interface implementation if the plug-in needs to have any. One must register any UI\textsuperscript{13} element before providing the capability.

\textsuperscript{13} User Interface: Means that enable people to interact with a system or some particular machine and hides that complexity of the object and provides input and output paths for the machine and the users
Main viewers, widgets and toolbars are examples of those elements that one must register in this way.

- IDispatcher: This service tells us about IRIS system level events
- INotificationService: In order to show notification text messages in the notification window of IRIS.
Chapter 3

Previous works, current attempt

This thesis\(^{14}\) is as sub-part of a bigger project consisting of four sections (Figure 6):

---

14 Thesis description for Open IRIS semantic desktop, Ola Leifler, Linköping University (May 15, 2007)
As shown in the picture, thesis 2 is the subject of this article. According to the thesis description pamphlet, the goal is to analyze office documents that make up an operational order according to document ontology whenever a user or the system services and applications open or modify these documents within IRIS. The system processes the documents against extensible document ontology where various patterns describe documents such as their location in a file structure, their name or any other information. In addition, the document harvester should be extensible enough to accommodate changes to whatever it harvested from documents. Investigating and improving the annotation mechanisms for MS Word or Open Office in this domain are part of the project.

In other words, this thesis focuses on harvesting domain-specific data from office documents that experts use for decisive purposes as in military decision-makings. The goal is to harvest data semantically with domain-specific criteria from plan aid documents such as NATO Guidelines for Operational Planning. By the time of decision making about the this subject, no other projects had existed in the same area; therefore, it is likely to be one of the first in its kind and unique.
Chapter 4

Methodology

4.1 Foreword

Early studying suggested the following stages to cover the earlier stages of the problem definition:

- Using a Java API\textsuperscript{15} to import an Open Office document in the application
- Methods of extracting the DOM\textsuperscript{16} structure of the original content of the document (In this case content.xml)
- Certain Java libraries to perform an XPath\textsuperscript{17} filtering onto the DOM structure

\textsuperscript{15} A set of declarations of functions or procedures, which a library, service or operating system provides to handle requests made by computer programs

\textsuperscript{16} An object model for representing HTML or XML related formats without regarding to their platform or the language they use

\textsuperscript{17} A language for extracting nodes, calculating values from the content of XML documents
• Using display features of IRIS framework to draw a GUI on the screen of the IRIS application

• Deciding on a GUI that provides the user with the effective data harvesting experience

• Benefiting from POJOs\(^{18}\) as an abstraction layer for manipulating the data inside the IRIS ontology which conceal the complexity of working directly with Semantic Object Models or the ontology level data structures

4.2 Planning for an Office system:
We studied both Microsoft Office (2003) and Open Office documents for the structure of their product files.

Open Office documents, saved in “.odt” file extension, benefit from the archiving Jar structure in which different parts of document may appear as commonly known xml files in most of their structures:

• Content.xml

• Meta.xml

\(^{18}\) POJO (Programming Ontology Java Objects): The programmatic abstract layer for working with the ontology and manipulate the necessary semantic objects which hides the complexity of working with semantic objects
Therefore, we decided to use Open Office format. Moreover, there were already libraries available to work with in Java. The main contents of an Open Office document reside in the “content.xml” portion of the original Jar file and they were the main target of our attention. Whenever you type and save contents using Open Office, those contents reside in content.xml file while the styles and layout or other Meta information produce other basic files that together with the content.xml form the final jar file called an odt file. On the next page, you see part of the XML structure of a sample content.xml file. If you rename an odt file to the extension (jar) and then unzip it into a folder, you will discover that the file explained earlier exist in the folder. Microsoft Corporation is also inclined toward using such file structure for storing document files in its MS Office suite.

```xml
<?xml version="1.0" encoding="UTF-8"?>
xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0"
xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0"
```
4.3 Java API for importing an Open Office document

ODF4J\textsuperscript{19} is an open source Java library that contains classes for working with ODF files and it maintains the ability of validating of Open Document XML files. The first step in using ODF4J is to load an Open Office document into the code level. For doing so, an instance of OdfPackage helps us in loading the document\textsuperscript{20}.

OdfPackage class has a static final property, which let us exactly choose which portion of the odt file we are interested to work with. In our case, STREAMNAME_CONTENT stands for “content.xml” portion of the file. Doing so lets us move to the next step of accessing and manipulating the DOM structure of this portion of odf file.

\textsuperscript{19} Open Document Format for Java (ODF4J)

\textsuperscript{20} Since we are receiving the document path in the standard Windows path addressing convention, it is necessary to take care of the backslashes and change them to double backslashes so that the string works with OpenDocumentFactory.load() method otherwise it does not. This means that within the file path string of the selected document, “\\” should be changed to “\\\\\\\\”. 
4.4 Manipulating the DOM structure

Once we are able to extract the content portion of the document, it is time to work directly with the DOM structure of the xml file. As you noticed earlier in the simple example of the structure of “content.xml” file, it is a standard XML file and it has its own tags for defining certain sections and attributes of the current document, which work together with other XML or non-XML files that exist inside an ODT file. They altogether form a document that has styles, tables, paragraphs and other standard document elements. These tags are very specific to Open Office application. They usually consist of two-word tag convention like <text: p> or <table: cell>. Such XML document can provide us with a DOM structure.

According to W3C organization, “The Document Object Model (DOM) is an application programming interface (API) for valid HTML and well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. In the DOM specification, experts use the term "document" in the broad sense - increasingly, XML appears as a way of representing many different kinds of information that may be stored in diverse systems, and many see much of this as data rather than as documents. Nevertheless, XML presents this data as
documents, and the DOM comes to manage this data. With the Document Object Model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content.” [15]

The package “org.w3c.dom.document” provides us with an instance of Document class, which in return lets us use the DOM tree interpretation of the document content. The document object acts as a container of the DOM tree for later manipulations of XPath query and filtering.

![Figure 7: A view of the simple intended GUI](image)

**4.5 Applying the XPath query filtering in the DOM tree:**

When the content of the xml file was loaded to the memory in the form of a DOM tree, it is time to apply the XPath method for filtering the nodes for the specific structure that the user originally desired to work with.

W3C describes XPath like this:
“The primary purpose of XPath is to address parts of an XML document. In support of this primary purpose, it also provides basic facilities for manipulation of strings, numbers and Booleans. XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values. XPath operates on the abstract, logical structure of an XML document, rather than its surface syntax. XPath gets its name from its use of a path notation as in URLs for navigating through the hierarchical structure of an XML document.

In addition to its use for addressing, XPath acts so that it has a natural subset it can use for matching (testing whether or not a node matches a pattern). XPath models an XML document as a tree of nodes. There are different types of nodes, including element nodes, attribute nodes and text nodes. XPath defines a way to compute a string-value for each type of node. Some types of nodes also have names. XPath fully supports XML Namespaces. Thus, the name of a node works as a pair consisting of a local part and a possibly null namespace URI; this is an expanded-name. [16]”
The package “org.w3c.dom.xpath.*” provides us with an instance of XPath evaluator by which it becomes possible to apply a string that contains the pattern which should be applied to the target XML file. The user defines this XPath string according to the inputs he delivers to the application through the user interface. In the primary section of the GUI, there is one field that the user uses in combination with or without a “browse” button that provides the document path. In the next step, there is another text field, which lets the user define the keyword he/she likes the application to look for. Next to this section, there exists one group of radio buttons, by which we are able to define the structure in the document to look for. The combination of the value of the radio buttons together with the query expression creates the XPath string.

An instance of the “XPathResult” holds the value of the applied XPath query for later processes. Suppose that the user intends to look for the keyword “DP” as in “Decisive Points” in the “Table” elements of the document. In such way, the application indeed checks the radio groups for the selection and the textbox for the keyword. Eventually the XPath string generates as you see below:
Once this XPath string evaluates and filters the DOM structure of the content.xml portion of the odt file, the application obtains the table element(s) of the document. Note that Open Office uses a specific XML notation, so before going any further, it is necessary to have a look at the structure and syntax of the XML structure of such documents. For example, <table:table> denotes a table and <table:table-row> stands for the row element of a table as <text:p> refers to paragraph. So it will be error-prone if there is no prior knowledge of the syntax of the tags of content.xml file and instead attempt to construct XPath strings according to the user input.

A view of the product of this stage looks like this:

```xml
  table:name="Table1" table:style-name="Table1">
  <table:table-column table:style-name="Table1.A"/>
  <table:table-column table:style-name="Table1.B"/>
  <table:table-column table:style-name="Table1.C"/>
  <table:table-column table:style-name="Table1.D"/>
  <table:table-column table:style-name="Table1.E"/>
  <table:table-column table:style-name="Table1.F"/>
  <table:table-column table:style-name="Table1.C"/>
  <table:table-column table:style-name="Table1.H"/>
  <table:table-column table:style-name="Table1.I"/>
</table:table>
```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DP</strong></td>
<td><strong>objectives</strong></td>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td></td>
<td>SVFOR force build up in staging areas</td>
<td>Preparations for operations in SVEALANDIA</td>
</tr>
</tbody>
</table>

...
As you have noticed, the generated XPath string has not evaluated the document effectively. It is actually at the row level that we would like to start traversing the product tree nodes. The less redundant nodes we have the better we can carry on with the nodes. Therefore, it will be a nice idea if we append another line to our XPath string as:

```
+"\n" + "//table:table-row","n
```

Upon completion of the evaluation of XPath in the original content, Only the <table:row> exists on the output, meaning that another round of evaluation happens to the later product.
The idea of a planning aid application is to help users with harvesting and managing data in any kind of planning areas. In our case, we specifically focused on military plan documents.

As mentioned before, Open Office implements its own syntax of XML tags. Military operational plan documents, as a form of specialized documents, use their own conventions of displaying information. That is, they (re)use tags to emphasize a special term or concept. For example, you see a row of data from a table containing specific data, which the user may be interested to harvest:

<table>
<thead>
<tr>
<th>DP</th>
<th>DP/objective</th>
<th>Purpose</th>
<th>Criteria for success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SVFOR force build up in staging areas</td>
<td>Preparations for operations in SVEALANDIA</td>
<td>Main body of the Forces are trained and prepared to deploy to SVEALANDIA. SVFOR can logistically support the operation.</td>
</tr>
</tbody>
</table>

There are four columns in this table and each indicates a role factor in the deciding procedure. This row contains fields related to *Decisive points* and their attributes in a *Conduct of Operations* document. In *DP/Objective* column, you see the value “*SVFOR force build up in staging areas*”. The xml form of the current cell and its value shows that it is not a single value, but two *span* tags:
Once the XPath has filtered the DOM structure according to the pattern generated from user’s input, there are ways of continuing the harvesting of the values. One way is to apply regular expressions, delete the unwanted portions of the tags, and finally take out the pure data. As a result of such method, we will no longer hold the DOM structure of the primary result and there shall be certain ways of storing the data for future processing. However, a better way is to traverse through the tree nodes and meet the nodes that XPath produced.

4.6 Traversing the nodes

Now that we have the desired portion of our XML document, it is time for us to extract the data by traversing through the nodes. Our application uses “org.w3c.dom.Node” package in order to have an instance of Node class. As long as the tree has children for visiting, a loop adds the relevant values to an array of strings so that it is capable of using them later for populating the knowledge base, which takes us to the next level: The IRIS level.
4.7 Transferring data into the knowledge base

So far, the application has performed the following tasks:

1. With the help of the user interface, the user browses for the file.
2. The application receives an input pattern of harvesting from the user on both the structural and keyword levels through the User Interface.
3. Using ODF4J APIs, the plug-in is capable of opening the odt zipped archive file and extract the content section.
5. It then generates an “XPath” string based on the inputs received from the previous levels.
6. After applying the XPath pattern onto the DOM tree, traverses among the child nodes and retrieves any relevant intended value.

At this level, in order to have a better understanding of the rest of the procedure, we first describe a brief explanation of Email harvesting task in IRIS according to the IRIS documentation[9].

IRIS converts emails into ontology objects. For representing an email, it uses the ontology class “clib:EmailMessage” while the POJO
IEMailAddressPojo represents it programmatically. It also involves other classes to store folders, attachments, headers and the email body [14].

The process of Email harvesting actually reifies the data:

It creates an instance of EmailMessage that represents the email in the ontology.

The procedure reifies any people appearing in the headers of the email into the “Person” entity within the ontology.

It also takes care of any email attachment by relating them to FileAttachmentRecord instances. The attachments reside on the hard disk and therefore an instance of ComputerFile reifies any FileAttachmentRecord, as well.

Accordingly, we are able to store our harvested data in the storage. In order to do so, one way would be to indicate various types of information that we are going to store. There are all sorts of information available on a harvesting process, which can be the metadata that the word processors embed in documents such as the author, the data of modifying or other
sorts of data. In such way, even the name of the document and the path where the document exists will be of importance to us.

It is preferred that just like the Email, there exists the same ontology class dedicated to decisive points in military plan aid documents. However, we can still rely on the more general existing ontology classes and map the most suitable one to the kind of data that the plug-in extracts from open Office documents. Once you create the ontology class, you need to implement the appropriate POJO to work with the class, as well.

“File name” and the “Path” where the file exists are important data to store. Referring to Protégé$^{21}$ and observing the IRIS ontology you learn that it shows a class named “clib:ComputerFile” that holds the properties we need. Those selected properties are “clib:directoryPathIs” and “clib:filenameIs”. You already noticed that within the IRIS framework, you use POJOs as the programmatic abstract layer to work with the ontology and manipulate the necessary semantic objects. Therefore, in order to access the class and point to its properties, IRIS has provided us with

---

$^{21}$ A free open-source Java tool providing an extensible architecture for the creation of customized knowledge-based applications [19]
“IComputerFilePojo” which in return represents the following methods for using [13]:

- “setDirectoryPathIs”
- setFileNameIs

The harvested data generated by the user’s input may appear in different structures and layouts of the original document. They can be in Lists, Tables or even paragraphs. In addition, their types may differ. Some may be images while others may be texts. However, they have one thing in common: They are all decisive points. Suppose that we have the table row in page 43 of this thesis as the product of our harvesting procedure. Whatever the types of each cell might be, the whole row stands for an instance of a semantic object. As a reminder, in Email application, the ontology class “clib:EmailMessage” represented an email. The same rule is true about the decisive points.

When you review the different general classes that exist in the ontology via Protégé, you may find different candidates for such purpose. By observing the class “Specified thing” this idea comes to mind that in the absence of dedicated ontology classes and relevant POJOs for military decisive points,
it is still possible to use a number of the methods in the relevant POJOs to populate the knowledge base with the products that the plug-in had already harvested in the odt document. Some candidate methods are:

- `setPriority`
- `setAnnotations`
- `setDescription`
- `setInformationContentOf`
- `setNoteTextIs`

It is possible to obtain an instance of `ISpecifiedThingPojo` and deal with our different kinds of data through its available interfaces.
Chapter 5

Results and Future Work

5.1 Results
The main goal of this thesis is harvesting information that exists specifically in office-related word processing documents such as Microsoft Word or Open Office Writer. It suggests a method of harvesting semantic entities that appear throughout the documents according to the user’s own pattern regarding the document structure and the keyword.

A semantic entity can exist in various locations and structures of a document. Either the semantic object is a table row or a paragraph; the application will filter the document according to the structure or the specified keyword. As an example, one will notice that the whole row of the table in page 44 actually indicates a semantic object. While this table is only a small portion of the bigger table, each row in the main table resembles a semantic “object” that, the application will later process for inserting into the knowledge base.
As mentioned earlier in page 33, our planning for choosing an office environment led to deciding on Open Office Writer application. You may read more in there. In the first place, our solution for helping the users to extract semantic entities from Open Office Writer documents provides the users with an interface in order to let them define the query structures. Moreover, the application provides an xml portion of the original document where the user’s keyword and preferred searching structure affect the original document as shown in page 41.

The next result of such harvesting is the objects (strings …) extracted by traversing the nodes of the DOM structure of the earlier product. Finally, the plug-in attempts to store the facts in the knowledge base and subsequently those data are available to other services of the IRIS semantic desktop and applications that analyze and refer to those facts as suggestions which, appear later in the suggestion pane, e.g.

This result plays an important role in equipping the IRIS environment with more data to make future reasoning and suggestions even smarter on such planning aid documents. In such way, in our estimation, the user interface is not of much priority because the true capability of the application lies behind the simple GUI.
We tested the application against specialized unclassified NATO documents that contained a series of decisive points and specific notations and punctuations whether in table cells or in other objects of the document.

![Figure 8: The output for the results](image)

5.2 Future Work
The current thesis aimed at bringing the issue of harvesting information within domain specific documents to the front. However, there is more room for expanding the idea and improving the efficiency of the project:

- designing and adding dedicated classes for decisive points or more general criteria for planning aid documents
- A more flexible and versatile method for harvesting information from within any part or structure of the document
- Studying Microsoft Word 2007 docx format since it represents the same ideas as in Open Office odt extension. In such way, the developer should redefine and study the XML tags

53
- A GUI that is capable of transferring the user’s preferences for a much more versatile harvesting of information not limited to certain structures within the document
- Implementing the application in the form of an IRIS service instead of a stand-alone plug-in, in the way that the plug-in triggers its service each time any harvesting task happens in the system, whether it is reifying an email attachment or opening a local file on the system
- The capability of dealing with any MIME type as a semantic object
Bibliography


[4]. W3C, OWL Web Ontology Language Guide, (W3C Recommendation 10 February 2004), Available at http://www.w3.org/TR/owl-guide/ (17/1/2008 15:00)


[15]. W3C organization, "What is Document Object Model?", Available at http://www.w3.org/TR/DOM-Level-2-Core/introduction.html (23/02/2008)

[16]. W3C organization, XML Path language, Available at http://www.w3.org/TR/xpath (23/02/2008)
[17]. Semantic desktop, Available at
https://www.semanticdesktop.org/xwiki/bin/view/Main/WebHome
(23/02/2008)

[18]. OpenIRIS module documentation Available at
http://www.openiris.org/downloads/IRIS-nightly/doc-
current/data/docs/DEX.owl (23/02/2008)

[19]. Protégé at Stanford University website Available at
http://protege.stanford.edu (23/02/2008)