Analysis and Design Job Information Exchange using XML based Technology

By

Utomo Budiyanto

A Master's Thesis
Submitted to the Faculty of

INFORMATION TECHNOLOGY

in partial fulfillment of the
requirements for the Degree of

MASTER OF SCIENCES
WITH A MAJOR IN INFORMATION TECHNOLOGY

SWISS GERMAN UNIVERSITY
Campus German Centre
Bumi Serpong Damai – 15321
Island of Java, Indonesia
www.sgu.ac.id

January 2005

Analysis and Design Job Information Exchange using XML based Technology

By

Utomo Budiyanto

A Master's Thesis
Submitted to the Faculty of

INFORMATION TECHNOLOGY

in partial fulfillment of the
requirements for the Degree of

MASTER OF SCIENCES
WITH A MAJOR IN INFORMATION TECHNOLOGY

SWISS GERMAN UNIVERSITY
Campus German Centre
Bumi Serpong Damai – 15321
Island of Java, Indonesia
www.sgu.ac.id

January 2005

STATEMENT BY THE AUTHOR

I hereby declare that this submission is my own work and to the best of my knowledge, it contains no material previously published or written by another person, not material which to a substantial extent has been accepted for the award of may other degree or diploma at any educational institution, except where due acknowledgement is made in the thesis.

Utomo Budiyanto Date

Approved by:

Thesis Advisor Date

Chairman of the Examination Steering Committee Date

ABSTRACT

Analysis and Design Job Information Exchange using XML based Technology

By

Utomo Budiyanto

SWISS GERMAN UNIVERSITY
Bumi Serpong Damai

Dwi Anoraganingrum, Advisor

Internet has become a big library which provide a lot of information about everything, as the growth of Internet sometimes information gathered doesn't fit with our needs, for example information about a job, there are many website offer the information about job but the information comes vary this is because there is no formal standard to displaying job information.

This thesis will analyze information from several website to get a pattern and design a pattern which can be used to display a job information using XML based technology so the information can be exchange between different types of computers using different types of operating system and application languages

This thesis develope prototype for Job Information Pattern Application and Job Information Grabber Application

DEDICATION

I dedicate this thesis to my parents and my two special agents my wife Agustina Hingyuningsih and my son Bijan Austin Pratama

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 – INTRODUCTION

1.1 Background

Internet has become a big library which provide a lot of information about everything, as the growth of Internet sometimes information gathered doesn't fit with our needs. for example information about a job, there are many website offer the information about job but the information comes vary from the complete one to minimum one from one site to another.

This fact will make a little difficulty for the visitor, the site owner or even the search engine, the visitor want to get a complete job information, search engine want to grab job information easily with pattern matching or parsing and the site owner want to exchange information easily between different types of computers using different types of operating system and application languages. Since people are free to use technology to display information, search engine must find a pattern to get a better result and this is quite complex, site owner can't exchange their information with any site because information that provide probably different with other site.

1.2 Problem Statement

There are several problems that apply in these services:

1. There is no formal standard for displaying job information in Internet, every site that provide job information have their own style.
2. It's difficult to exchange job information between one to another

1.3 Objective

The main purpose of this thesis is to analyze job information which provided by several website and get the pattern to compose and design what things needed for displaying job information using XML based technology so it can exchange effectively, easily between different types of computers using different types of operating system and application languages

1.4 Methodology

1.4.1 Analyze Phase

In analyze phase, all information gathered from several website in Internet to get pattern so it can be used for composing pattern that will be used.

1.4.2 Design Phase

In design phase, result from analyze phase will be used to design patern for displaying job information system that can exchange between different types of computers using different types of operating system and application languages

1.5 Scope of the Thesis

Scope of this thesis is to analyze and design a pattern that need to provide job information using XML based technology, so information will exchange between different types of computers using different types of operating system and application languages, tesis will not implement it become an application.

1.6 Structure of the Thesis

Chapter 1 explains the problem, objective that want to be achieve, methodology to solve the problem and scope that limit the thesis.

Chapter 2 contents all the theoretical that used in solving the problem.

Chapter 3 covers all the analysis phase

Chapter 4 covers the result of analysis phase

Chapter 5 serves as a conclusion, as well as considers any scope for further work.

CHAPTER 2 – LITERATURE REVIEW

2.1 Extensible Markup Language (XML)

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.[1]

Extensible Markup Language, abbreviated XML, describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language [ISO 8879]. By construction, XML documents are conforming SGML documents. [1]

XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure. [1]

A software module called an **XML processor** is used to read XML documents and provide access to their content and structure. It is assumed that an XML processor is doing its work on behalf of another module, called the **application**. This specification describes the required behavior of an XML processor in terms of how it must read XML data and the information it must provide to the application. [1]

2.1.1 Origin and Goals

XML was developed by an XML Working Group (originally known as the SGML Editorial Review Board) formed under the auspices of the World Wide Web Consortium (W3C) in 1996. It was chaired by Jon Bosak of Sun Microsystems with the active participation of an XML Special Interest Group (previously known as the SGML Working Group) also organized by the W3C. The membership of the XML

Working Group is given in an appendix. Dan Connolly served as the WG's contact with the W3C.

The design goals for XML are: [1]

1. XML shall be straightforwardly usable over the Internet.

   Users must be able to view XML documents as quickly and easily as HTML documents. In practice, this will only be possible when XML browsers are as robust and widely available as HTML browsers, but the principle remains. [2]

2. XML shall support a wide variety of applications.

   XML should be beneficial to a wide variety of diverse applications: authoring, browsing, content analysis, etc. Although the initial focus is on serving structured documents over the web, it is not meant to narrowly define XML. [2]

3. XML shall be compatible with SGML.

   Most of the people involved in the XML effort come from organizations that have a large, in some cases staggering, amount of material in SGML. XML was designed pragmatically, to be compatible with existing standards while solving the relatively new problem of sending richly structured documents over the web. [2]

4. It shall be easy to write programs which process XML documents.

   The colloquial way of expressing this goal while the spec was being developed was that it ought to take about two weeks for a competent computer science graduate student to build a program that can process XML documents. [2]

5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.

Optional features inevitably raise compatibility problems when users want to share documents and sometimes lead to confusion and frustration. [2]

6. XML documents should be human-legible and reasonably clear.

   If you don't have an XML browser and you've received a hunk of XML from somewhere, you ought to be able to look at it in your favorite text editor and actually figure out what the content means. [2]

7. The XML design should be prepared quickly.

   XML was needed immediately and was developed as quickly as possible. [2]

8. The design of XML shall be formal and concise.

   In many ways a corollary to rule 4, it essentially means that XML must be expressed in EBNF and must be amenable to modern compiler tools and techniques. There are a number of technical reasons why the SGML grammar *cannot* be expressed in EBNF. Writing a proper SGML parser requires handling a variety of rarely used and difficult to parse language features. XML does not. [2]

9. XML documents shall be easy to create.

   Although there will eventually be sophisticated editors to create and edit XML content, they won't appear immediately. In the interim, it must be possible to create XML documents in other ways: directly in a text editor, with simple shell and Perl scripts, etc. [2]

10. Terseness in XML markup is of minimal importance.

    Several SGML language features were designed to minimize the amount of typing required to manually key in SGML documents.

> These features are not supported in XML. From an abstract point of view, these documents are indistinguishable from their more fully specified forms, but supporting these features adds a considerable burden to the SGML parser (or the person writing it, anyway). In addition, most modern editors offer better facilities to define shortcuts when entering text. [2]

2.1.2 Documents

A data object is an **XML document** if it is well-formed, as defined in this specification. A well-formed XML document may in addition be valid if it meets certain further constraints. [1]

Each XML document has both a logical and a physical structure. Physically, the document is composed of units called entities. An entity may refer to other entities to cause their inclusion in the document. A document begins in a "root" or document entity. Logically, the document is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit markup. The logical and physical structures must nest properly. [1]

2.1.3 Well-Formed XML Documents

A textual object is a well-formed XML document if:

1. Taken as a whole, it matches the production labeled document.
2. It meets all the well-formedness constraints given in this specification.
3. Each of the parsed entities which are referenced directly or indirectly within the document is well-formed.

Document

```
[1] document::= prolog element Misc*
```

Matching the document production implies that:

1. It contains one or more elements.
2. There is exactly one element, called the **root**, or document element, no part of which appears in the content of any other element. For all other elements, if the start-tag is in the content of another element, the end-tag is in the content of the same element. More simply stated, the elements, delimited by start- and end-tags, nest properly within each other.

As a consequence of this, for each non-root element C in the document, there is one other element P in the document such that C is in the content of P, but is not in the content of any other element that is in the content of P. P is referred to as the **parent** of C, and C as a **child** of P. [1]

2.1.4 Extended Backus-Naur Form (EBNF)

One of the most significant design improvements in XML is to make it easy to use with modern compiler tools. Part of this improvement involves making it possible to express the syntax of XML in Extended Backus-Naur Form (EBNF). If you've never seen EBNF before, think of it this way: [2]

- EBNF is a set of rules, called productions
- Every rule describes a specific fragment of syntax
- A document is valid if it can be reduced to a single, specific rule, with no input left, by repeated application of the rules.

Let's take a simple example that has nothing to do with XML (or the real rules of language):

```
[1] Word       ::= Consonant Vowel+ Consonant
[2] Consonant  ::= [^aeiou]
[3] Vowel      ::= [aeiou]
```

Rule 1 states that a word is a consonant followed by one or more vowels followed by another consonant. Rule 2 states that a consonant is any letter other than a, e, i, o, or u.

Rule 3 states that a vowel is any of the letters a, e, i, o, or u. (The exact syntax of the rules, the meaning of square brackets and other special symbols, is laid out in the XML specification.) [2]

Using the above example, is this red a `Word`? Yes.

1. red is the letter r followed by the letter e followed by the letter d: `'r' 'e' 'd'`.
2. `r` is a `Consonant` by rule 2, so red is: `Consonant 'e' 'd'`
3. `e` is a vowel by rule 3, so red is: `Consonant Vowel 'd'`.
4. By rule 2 again, red is: `Consonant Vowel Consonant` which, by rule 1, is a `Word`.

By the same analysis, reed , road , and xeaiioug are also words, but rate is not. There is no way to match `Consonant Vowel Consonant Vowel` using the EBNF above. XML is defined by an EBNF grammar of about 80 rules. Although the rules are more complex, the same sort of analysis allows an XML parser to determine that `<greeting>Hello World</greeting>` is a syntactically correct XML document while `<greeting]Wrong Bracket!</greeting>` is not. [2]

While EBNF isn't an efficient way to represent syntax for human consumption, there are programs that can automatically turn EBNF into a parser. This makes it a particularly efficient way to represent the syntax for a language that will be parsed by a computer. [2]

2.1.5 What Do XML Documents Look Like?

If you are conversant with HTML or SGML, XML documents will look familiar. A simple XML document is presented in Example 1.

Example 1. A Simple XML Document

```
<?xml version="1.0"?>

<oldjoke>

<burns>Say <quote>goodnight</quote>,
Gracie.</burns>

<allen><quote>Goodnight,
```

```
Gracie.</quote></allen>
<applause/>
</oldjoke>
```

A few things may stand out to you:

- The document begins with a processing instruction: `<?xml ...?>`. This is the *XML declaration*. While it is not required, its presence explicitly identifies the document as an XML document and indicates the version of XML to which it was authored.
- There's no document type declaration. Unlike SGML, XML does not require a document type declaration. However, a document type declaration can be supplied, and some documents will require one in order to be understood unambiguously.
- Empty elements (`<applause/>` in this example) have a modified syntax. While most elements in a document are wrappers around some content, empty elements are simply markers where something occurs (a horizontal rule for HTML's `<hr>` tag). The trailing /> in the modified syntax indicates to a program processing the XML document that the element is empty and no matching end-tag should be sought. Since XML documents do not require a document type declaration, without this clue it could be impossible for an XML parser to determine which tags were intentionally empty and which had been left empty by mistake.
  XML has softened the distinction between elements which are declared as `EMPTY` and elements which merely have no content. In XML, it is legal to use the empty-element tag syntax in either case. It's also legal to use a start-tag/end-tag pair for empty elements: `<applause></applause>`. If interoperability is of any concern, it's best to reserve empty-element tag syntax for elements which are declared as `EMPTY` and to only use the empty-element tag form for those elements.

XML documents are composed of markup and content. There are six kinds of markup that can occur in an XML document: elements, entity references, comments, processing instructions, marked sections, and document type declarations. The following sections introduce each of these markup concepts. [2]

2.1.6 Elements

Elements are the most common form of markup. Delimited by angle brackets, most elements identify the nature of the content they surround. Some elements may be empty, as seen above, in which case they have no content. If an element is not empty, it begins with a start-tag, `<element>`, and ends with an end-tag, `</element>`. [2]

2.1.7 Attributes

Attributes are name-value pairs that occur inside start-tags after the element name. For example,

```
<div class="preface">
```

is a `div` element with the attribute `class` having the value `preface`. In XML, all attribute values must be quoted.[2]

2.1.8 Entity References

In order to introduce markup into a document, some characters have been reserved to identify the start of markup. The left angle bracket, < , for instance, identifies the beginning of an element start- or end-tag. In order to insert these characters into your document as content, there must be an alternative way to represent them. In XML, entities are used to represent these special characters. Entities are also used to refer to often repeated or varying text and to include the content of external files.[2]

Every entity must have a unique name. Defining your own entity names is discussed in the section on entity declarations. In order to use an entity, you simply reference it by name. Entity references begin with the ampersand and end with a semicolon.[2]

For example, the `lt` entity inserts a literal < into a document. So the string `<element>` can be represented in an XML document as `&lt;element>`.

A special form of entity reference, called a character reference, can be used to insert arbitrary Unicode characters into your document. This is a mechanism for inserting characters that cannot be typed directly on your keyboard.[2]

Character references take one of two forms: decimal references, `&#8478;`, and hexadecimal references, `&#x211E;`. Both of these refer to character number U+211E from Unicode (which is the standard Rx prescription symbol, in case you were wondering).[2]

2.1.9 Comments

Comments begin with `<!--` and end with `-->`. Comments can contain any data except the literal string `--`. You can place comments between markup anywhere in your document.

Comments are not part of the textual content of an XML document. An XML processor is not required to pass them along to an application.[2]

2.1.10 Processing Instructions

Processing instructions (PIs) are an escape hatch to provide information to an application. Like comments, they are not textually part of the XML document, but the XML processor is required to pass them to an application.[2]

Processing instructions have the form: `<?name pidata?>`. The name, called the PI target, identifies the PI to the application. Applications should process only the targets they recognize and ignore all other PIs. Any data that follows the PI target is optional, it is for the application that recognizes the target. The names used in PIs may be declared as notations in order to formally identify them.[2]

PI names beginning with `xml` are reserved for XML standardization.

2.1.11 CDATA Sections

In a document, a `CDATA` section instructs the parser to ignore most markup characters.

Consider a source code listing in an XML document. It might contain characters that the XML parser would ordinarily recognize as markup (< and &, for example). In order to prevent this, a CDATA section can be used. [2]

```
<![CDATA[
*p = &q;
b = (i <= 3);
]]>
```

Between the start of the section, `<![CDATA[` and the end of the section, `]]>`, all character data is passed directly to the application, without interpretation. Elements, entity references, comments, and processing instructions are all unrecognized and the characters that comprise them are passed literally to the application. [2]

The only string that cannot occur in a CDATA section is `]]>`. [2]

2.1.12 Document Type Declarations

A large percentage of the XML specification deals with various sorts of declarations that are allowed in XML. If you have experience with SGML, you will recognize these declarations from SGML DTDs (Document Type Definitions). If you have never seen them before, their significance may not be immediately obvious. [2]

One of the greatest strengths of XML is that it allows you to create your own tag names. But for any given application, it is probably not meaningful for tags to occur in a completely arbitrary order. Consider the old joke example introduced earlier. Would this be meaningful?

```
<gracie><quote><oldjoke>Goodnight,
<applause/>Gracie</oldjoke></quote>

<burns><gracie>Say <quote>goodnight</quote>,
</gracie>Gracie.</burns></gracie>
```

It's so far outside the bounds of what we normally expect that it's nonsensical. It just doesn't *mean* anything. [2]

However, from a strictly syntactic point of view, there's nothing wrong with that XML document. So, if the document is to have meaning, and certainly if you're writing a stylesheet or application to process it, there must be some constraint on the

sequence and nesting of tags. Declarations are where these constraints can be expressed. [2]

More generally, declarations allow a document to communicate meta-information to the parser about its content. Meta-information includes the allowed sequence and nesting of tags, attribute values and their types and defaults, the names of external files that may be referenced and whether or not they contain XML, the formats of some external (non-XML) data that may be referenced, and the entities that may be encountered. [2]

There are four kinds of declarations in XML: element type declarations, attribute list declarations, entity declarations, and notation declarations. [2]

2.1.13 Element Type Declarations

Element type declarations identify the names of elements and the nature of their content. A typical element type declaration looks like this:

```
<!ELEMENT oldjoke (burns+, allen, applause?)>
```

This declaration identifies the element named `oldjoke`. Its *content model* follows the element name. The content model defines what an element may contain. In this case, an `oldjoke` must contain `burns` and `allen` and may contain `applause`. The commas between element names indicate that they must occur in succession. The plus after `burns` indicates that it may be repeated more than once but must occur at least once. The question mark after `applause` indicates that it is optional (it may be absent, or it may occur exactly once). A name with no punctuation, such as `allen`, must occur exactly once.[2]

Declarations for `burns`, `allen`, `applause` and all other elements used in any content model must also be present for an XML processor to check the validity of a document.

In addition to element names, the special symbol `#PCDATA` is reserved to indicate character data. The moniker PCDATA stands for parseable character data .

Elements that contain only other elements are said to have *element content*. Elements that contain both other elements and `#PCDATA` are said to have *mixed content*.

For example, the definition for `burns` might be

```
<!ELEMENT burns (#PCDATA | quote)*>
```

The vertical bar indicates an or relationship, the asterisk indicates that the content is optional (may occur zero or more times); therefore, by this definition, `burns` may contain zero or more characters and `quote` tags, mixed in any order. All mixed content models must have this form: `#PCDATA` must come first, all of the elements must be separated by vertical bars, and the entire group must be optional.[2]

Two other content models are possible: `EMPTY` indicates that the element has no content (and consequently no end-tag), and `ANY` indicates that *any* content is allowed. The `ANY` content model is sometimes useful during document conversion, but should be avoided at almost any cost in a production environment because it disables all content checking in that element.[2]

Here is a complete set of element declarations for Example 1:

Example 2. Element Declarations for Old Jokes

```
<!ELEMENT oldjoke  (burns+, allen, applause?)>
<!ELEMENT burns    (#PCDATA | quote)*>
<!ELEMENT allen    (#PCDATA | quote)*>
<!ELEMENT quote    (#PCDATA)*>
<!ELEMENT applause EMPTY>
```

2.2 Resource Description Framework (RDF)

The Web is a lot like a really *really* big library. There are millions of things out there, and if you know the URL (in effect a kind of call number) you can get them. Since the Web has books, movies, and pizza joints, the number of ways you might want to look things up includes all the things a library uses, plus all the things the video store uses, plus all the things the Yellow Pages use, and lots more.[3]

The problem at the moment is that there is hardly any metadata on the Web. So how do we find things? Mostly by using dumb, brute force techniques. The dumb, brute force is supplied by the wandering web robots of search engine sites like Altavista, Infoseek, and Excite. These sites do the equivalent of going through the library, reading every book, and allowing us to look things up based on the words in the text. It's not surprising that people complain about search results, or that the robots are always way behind the growth and change of the Web.[3]

In fact there is one metadata-based general purpose lookup facility: Yahoo! Yahoo doesn't use a robot. When you search through Yahoo, you're searching through human-generated subject categories and site labels. Compared to the amount of metadata that a library maintains for its books, Yahoo! is pitiful, but its popularity is clear evidence of the power of (even limited) metadata.[3]

The simplest definition of metadata is " structured data about data." [4]

Metadata is descriptive information about an object or resource whether it be physical or electronic. While metadata itself is relatively new, the underlying concepts behind metadata have been in use for as long as collections of information have been organized. Library card catalogs represent a well-established type of metadata that has served as collection management and resource discovery tools for decades.[4]

Metadata can be generated either "by hand" or derived automatically using software [4]

RDF (Resource Description Framework), as its name implies, is a framework for describing and interchanging metadata. It is built on the following rules.[3]

1. A **Resource** is anything that can have a URI; this includes all the Web's pages, as well as individual elements of an XML document. An example of a resource is a draft of the document you are now reading and its URL is `http://www.textuality.com/RDF/Why.html`
2. A **Property** is a Resource that has a name and can be used as a property, for example `Author` or `Title`. In many cases, all we really care about is the name; but a Property needs to be a resource so that it can have its own properties.

3. A **Statement** consists of the combination of a Resource, a Property, and a value. These parts are known as the 'subject', 'predicate' and 'object' of a Statement. An example Statement is "The Author of `http://www.textuality.com/RDF/Why.html` is Tim Bray." The value can just be a string, for example "Tim Bray" in the previous example, or it can be another resource, for example "The Home-Page of `http://www.textuality.com/RDF/Why.html` is `http://www.textuality.com`."
4. There is a straightforward method for expressing these abstract Properties in XML, for example:

```
<rdf:Description about='http://www.textuality.com/RDF/Why-RDF.html'>
<Author>Tim Bray</Author>
<Home-Page rdf:resource='http://www.textuality.com' />
</rdf:Description>
```

RDF is carefully designed to have the following characteristics.

Independence

Since a Property is a resource, any independent organization (or even person) can invent them. I can invent one called Author, and you can invent one called Director (which would only apply to resources that are associated with movies), and someone else can invent one called Restaurant-Category. This is necessary since we don't have a GOD to take care of it for us. [3]

Interchange

Since RDF Statements can be converted into XML, they are easy for us to interchange. This would probably be necessary even if we *did* have a GOD. [3]

Scalability

RDF statements are simple, three-part records (Resource, Property, value), so they are easy to handle and look things up by, even in large numbers. The Web is already big and getting bigger, and we are probably going to have (literally) billions of these floating around (millions even for a big Intranet). Scalability is important. [3]

Properties are Resources

Properties can have their own properties and can be found and manipulated like any other Resource. This is important because there are going to be lots of them; too many to look at one by one. For example, I might want to know if anyone out there has defined a Property that describes the genre of a movie, with values like Comedy, Horror, Romance, and Thriller. I'll need metadata to help with that. [3]

Values Can Be Resources

For example, most web pages will have a property named Home-Page which points at the home page of their site. So the values of properties, which obviously have to include things like title and author's name, also have to include Resources. [3]

Statements Can Be Resources

Statements can also have properties. Since there's no GOD to provide useful assertions for all the resources, and since the Web is way too big for us to provide our own, we're going to need to do lookups based on other people's metadata (as we do today with Yahoo!). This means that we'll want, given any Statement such as "The Subject of this Page is Donkeys", to be able to ask "Who said so? And When?" One useful way to do this would be with metadata; so Statements will need to have Properties. [3]

2.2.1 Why Not Just Use XML?

XML allows you to invent tags, which may contain both text data and other tags. XML has a built-in distinction between *element types*, for example the `IMG` element type in HTML, and *elements*, for example an individual `<img src='Madonna.jpg'>`; this corresponds naturally to the distinction between Properties and Statements. So it seems as though XML documents should be a natural vehicle for exchanging general purpose metadata. [3]

XML, however, falls apart on the **Scalability** design goal. There are two problems:

1. The order in which elements appear in an XML document is significant and often very meaningful. This seems highly unnatural in the metadata world. Who cares whether a movie's Director or Title is listed first, as long as both

are available for lookups? Furthermore, maintaining the correct order of millions of data items is expensive and difficult, in practice.

2. XML allows constructions like

```
<Description>The value of this property contains some
text, mixed up with child properties such as its temperature
(<Temp>48</Temp>) and longitude
(<Longt>101</Longt>). [&Disclaimer;]</Description>
```

When you represent general XML documents in computer memory, you get weird data structures that mix trees, graphs, and character strings. In general, these are hard to handle in even moderate amounts, let alone by the billion. [3]

On the other hand, something like XML is an absolutely necessary part of the solution to RDF's **Interchange** design goal. XML is unequalled as an exchange format on the Web. But by itself, it doesn't provide what you need in a metadata framework. [3]

above define the central ideas of RDF. It turns out that it takes quite a lot of abstract terminology and XML syntax to define them precisely enough that people can write computer programs to process them. In particular, turning Statements into Resources is quite tricky. It also turns out that in a (very) few cases, you do need to order your properties, and this requires quite a bit of syntax. [3]

2.2.2 Vocabularies

RDF, as we've seen, provides a model for metadata, and a syntax so that independent parties can exchange it and use it. What it *doesn't* provide though is any Properties of its own. RDF doesn't define Author or Title or Director or Business-Category. That would be a job for GOD, if there were one. Since there isn't, it's a job for everyone.[3]

It seems unlikely that one Property standing by itself is apt to be very useful. It is expected that these will come in packages; for example, a set of basic bibliographic Properties like Author, Title, Date, and so on. Then a more elaborate set from OCLC and a competing one from the Library of Congress. These packages are called **Vocabularies**; it's easy to imagine Property vocabularies describing books, videos, pizza joints, fine wines, mutual funds, and many other species of Web wildlife.[3]

2.2.3 What RDF Might Mean

The Web is too big for anyone person to stay on top of. In fact, it contains information about a huge number of subjects, and for most of those subjects (such as fine wines, home improvement, and cancer therapy), the Web has too much information for any one person to stay on top of and much of anything else .[3]

This means that opinions, pointers, indexes, and anything that helps people discover things are going to be commodities of very high value. Nobody thinks that everyone will use the same vocabulary (nor should they), but with RDF we can have a marketplace in vocabularies. Anyone can invent them, advertise them, and sell them. The good (or best-marketed) ones will survive and prosper. Probably most niches of information will come to be dominated by a small number of vocabularies, the way that library catalogs are today.[3]

And even among people who are sharing the use of metadata vocabularies, there's no need to share the same software. RDF makes it possible to use multiple pieces of software to process the same metadata, and to use a single piece of software to process (at least in part) many different metadata vocabularies.[3]

With any luck, this should make the Web more like a library, or a video store, or a phone book, than it is today.

2.2.4 RDF Example

This is a few lines from a list of CD records:

Title	Artist	Country	Company	Price	Year
Empire Burlesque	Bob Dylan	USA	Columbia	10.90	1985
Hide your heart	Bonnie Tyler	UK	CBS Records	9.90	1988
...					

Table 2.1 List of CD Records

This is a few lines of an XML file with a reference to an RDF description:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd">
<rdf:Description
 rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
```

```
  <cd:artist>Bob Dylan</cd:artist>
  <cd:country>USA</cd:country>
  <cd:company>Columbia</cd:company>
  <cd:price>10.90</cd:price>
  <cd:year>1985</cd:year>
</rdf:Description>
<rdf:Description
 rdf:about="http://www.recshop.fake/cd/Hide your heart">
  <cd:artist>Bonnie Tyler</cd:artist>
  <cd:country>UK</cd:country>
  <cd:company>CBS Records</cd:company>
  <cd:price>9.90</cd:price>
  <cd:year>1988</cd:year>
</rdf:Description>
.
.
.

</rdf:RDF>
```

The first line in the XML file is the XML declaration, telling the version of XML.

The **rdf:RDF** element (starting with rfd:RDF and ending with /rdf:RDF) indicates that the content is RDF.

The **xmlns:rdf** namespace, specifies that tags with the rdf: prefix are from the namespace defined by "http://www.w3.org/1999/02/22-rdf-syntax-ns#".

The **xmlns:cd** namespace, specifies that tags with the cd: prefix are from the namespace defined by "http://www.recshop.fake/cd".

The **rdf:Description** element (starting with rdf:Description and ending with /rdf:Description) contains the description of a resource identified by the **rdf:about** attribute.

The **cd:artist** element describes a property of the resource, and so does cd:country, etc.

2.2.5 Resources, Properties, Values

When you think of RDF, think the following sentence: "Resources have Properties with Values".

From the example above you can read:

The Resource: http://www.recshop.fake/cd/Hide your heart" has a Property called "artist" with the value "Bonnie Tyler"

The Resource: http://www.recshop.fake/cd/Hide your heart" has a Property called "price" with the value "9.90"

Or like humans would normally say: "The CD Hide your heart costs $9.90" [5]

2.3 Really Simple Syndication (RSS)

RSS stands for "Really Simple Syndication" and is a dialect of XML created to allow lists of information, known as "feeds", to be published by content producers and subscribed to by readers. Originated by Netscape in the late 1990s, RSS works by taking key bits of websites, such as headlines, and distributing that information out a bare form, stripped of all fancy graphics and layouts. [6]

RSS is a dialect of XML. All RSS files must conform to the XML 1.0 specification, as published on the World Wide Web Consortium (W3C) website. [7]

At the top level, a RSS document is a <rss> element, with a mandatory attribute called version, that specifies the version of RSS that the document conforms to. If it conforms to this specification, the version attribute must be 2.0. Subordinate to the <rss> element is a single <channel> element, which contains information about the channel (metadata) and its contents.[7]

First we document the required and optional sub-elements of <channel>, and then document the sub-elements of <item>. The final sections answer frequently asked questions, and provide a roadmap for future evolution, and guidelines for extending RSS.[7]

2.3.1 Required channel elements

Here's a list of the required channel elements, each with a brief description, an example, and where available, a pointer to a more complete description.[7]

Element	Description	Example
Title	The name of the channel. It's how people refer to your service. If you have an HTML website that contains the same information as your RSS file, the title of your channel should be the same as the title of your website.	GoUpstate.com News Headlines
Link	The URL to the HTML website corresponding to the channel.	http://www.goupstate.com/
description	Phrase or sentence describing the channel.	The latest news from GoUpstate.com, a Spartanburg Herald-Journal Web site.

2.3.2 Optional channel elements

Here's a list of optional channel elements.[7]

Element	Description	Example
language	The language the channel is written in. This allows aggregators to group all Italian language sites, for example, on a single page. A list of allowable values for this element, as provided by Netscape, is here. You may also use values defined by the W3C.	en-us
copyright	Copyright notice for content in the channel.	Copyright 2002, Spartanburg Herald-Journal
managingEditor	Email address for person responsible for editorial content.	geo@herald.com (George Matesky)
webMaster	Email address for person responsible for technical	betty@herald.com (Betty Guernsey)

	issues relating to channel.	
pubDate	The publication date for the content in the channel. For example, the New York Times publishes on a daily basis, the publication date flips once every 24 hours. That's when the pubDate of the channel changes. All date-times in RSS conform to the Date and Time Specification of RFC 822, with the exception that the year may be expressed with two characters or four characters (four preferred).	Sat, 07 Sep 2002 00:00:01 GMT
lastBuildDate	The last time the content of the channel changed.	Sat, 07 Sep 2002 09:42:31 GMT
category	Specify one or more categories that the channel belongs to. Follows the same rules as the <item>-level category element. More info.	<category>Newspapers</category>
generator	A string indicating the program used to generate the channel.	MightyInHouse Content System v2.3
docs	A URL that points to the documentation for the format used in the RSS file. It's probably a pointer to this page. It's for people who might stumble across an RSS file on a Web server 25 years from now and wonder what it is.	http://blogs.law.harvard.edu/tech/rss
cloud	Allows processes to register with a cloud to be notified of updates to the channel, implementing a lightweight publish-subscribe protocol for RSS feeds. More info here.	<cloud domain="rpc.sys.com" port="80" path="/RPC2" registerProcedure="pingMe" protocol="soap"/>
ttl	ttl stands for time to live. It's a number of minutes that indicates how long a channel can be cached before	<ttl>60</ttl>

	refreshing from the source. More info here.
image	Specifies a GIF, JPEG or PNG image that can be displayed with the channel. More info here.
rating	The PICS rating for the channel.
textInput	Specifies a text input box that can be displayed with the channel. More info here.
skipHours	A hint for aggregators telling them which hours they can skip. More info here.
skipDays	A hint for aggregators telling them which days they can skip. More info here.

2.3.3 <image> sub-element of <channel>

<image> is an optional sub-element of <channel>, which contains three required and three optional sub-elements.

<url> is the URL of a GIF, JPEG or PNG image that represents the channel.

<title> describes the image, it's used in the ALT attribute of the HTML <img> tag when the channel is rendered in HTML.

<link> is the URL of the site, when the channel is rendered, the image is a link to the site. (Note, in practice the image <title> and <link> should have the same value as the channel's <title> and <link>.

Optional elements include <width> and <height>, numbers, indicating the width and height of the image in pixels. <description> contains text that is included in the TITLE attribute of the link formed around the image in the HTML rendering.

Maximum value for width is 144, default value is 88.

Maximum value for height is 400, default value is 31.[7]

2.3.4 <cloud> sub-element of <channel>

<cloud> is an optional sub-element of <channel>.

It specifies a web service that supports the rssCloud interface which can be

implemented in HTTP-POST, XML-RPC or SOAP 1.1.
Its purpose is to allow processes to register with a cloud to be notified of updates to the channel, implementing a lightweight publish-subscribe protocol for RSS feeds.
<cloud domain="rpc.sys.com" port="80" path="/RPC2" registerProcedure="myCloud.rssPleaseNotify" protocol="xml-rpc" />

In this example, to request notification on the channel it appears in, you would send an XML-RPC message to rpc.sys.com on port 80, with a path of /RPC2. The procedure to call is myCloud.rssPleaseNotify. [7]

2.3.5 <ttl> sub-element of <channel>

<ttl> is an optional sub-element of <channel>.
ttl stands for time to live. It's a number of minutes that indicates how long a channel can be cached before refreshing from the source. This makes it possible for RSS sources to be managed by a file-sharing network such as Gnutella. [7]
Example: <ttl>60</ttl>

2.3.6 <textInput> sub-element of <channel>

A channel may optionally contain a <textInput> sub-element, which contains four required sub-elements.
<title> -- The label of the Submit button in the text input area.
<description> -- Explains the text input area.
<name> -- The name of the text object in the text input area.
<link> -- The URL of the CGI script that processes text input requests.

The purpose of the <textInput> element is something of a mystery. You can use it to specify a search engine box. Or to allow a reader to provide feedback. Most aggregators ignore it.[7]

2.3.7 Elements of <item>

A channel may contain any number of <item>s. An item may represent a "story" -- much like a story in a newspaper or magazine; if so its description is a synopsis of the story, and the link points to the full story. An item may also be complete in itself, if

so, the description contains the text (entity-encoded HTML is allowed), and the link and title may be omitted. All elements of an item are optional, however at least one of title or description must be present.[7]

Element	Description	Example
Title	The title of the item.	Venice Film Festival Tries to Quit Sinking
Link	The URL of the item.	http://www.nytimes.com/2002/09/07/movies/07FE
description	The item synopsis.	Some of the most heated chatter at the Venice Film Festival this week was about the way that the arriv stars at the Palazzo del Cinema was being staged.
author	Email address of the author of the item.	oprah@oxygen.net
category	Includes the item in one or more categories.	Simpsons Characters
comments	URL of a page for comments relating to the item.	http://www.myblog.org/cgi-local/mt/mt-comments.cgi?entry_id=290
enclosure	Describes a media object that is attached to the item.	
guid	A string that uniquely identifies the item.	http://inessential.com/2002/09/01.php#a2
pubDate	Indicates when the item was published.	Sun, 19 May 2002 15:21:36 GMT
source	The RSS channel that the item came from.	Quotes of the Day

2.3.8 <source> sub-element of <item>

<source> is an optional sub-element of <item>.
Its value is the name of the RSS channel that the item came from, derived from its <title>. It has one required attribute, url, which links to the XMLization of the source.
<source url="http://static.userland.com/tomalak/links2.xml">Tomalak's Realm</source>

The purpose of this element is to propagate credit for links, to publicize the sources of news items. It can be used in the Post command of an aggregator. It should be

generated automatically when forwarding an item from an aggregator to a weblog authoring tool.[7]

2.3.9 <enclosure> sub-element of <item>

<enclosure> is an optional sub-element of <item>.
It has three required attributes. url says where the enclosure is located, length says how big it is in bytes, and type says what its type is, a standard MIME type.
The url must be an http url.

<enclosure url="http://www.scripting.com/mp3s/weatherReportSuite.mp3" length="12216320" type="audio/mpeg" />[7]

2.3.10 <category> sub-element of <item>

<category> is an optional sub-element of <item>.
It has one optional attribute, domain, a string that identifies a categorization taxonomy.

The value of the element is a forward-slash-separated string that identifies a hierarchic location in the indicated taxonomy. Processors may establish conventions for the interpretation of categories. Two examples are provided below:

<category>Grateful Dead</category>
<category domain="http://www.fool.com/cusips">MSFT</category>

You may include as many category elements as you need to, for different domains, and to have an item cross-referenced in different parts of the same domain.[7]

2.3.11 <pubDate> sub-element of <item>

<pubDate> is an optional sub-element of <item>.
Its value is a date, indicating when the item was published. If it's a date in the future, aggregators may choose to not display the item until that date. [7]

<pubDate>Sun, 19 May 2002 15:21:36 GMT</pubDate>

2.3.12 <guid> sub-element of <item>

<guid> is an optional sub-element of <item>.
guid stands for globally unique identifier. It's a string that uniquely identifies the item. When present, an aggregator may choose to use this string to determine if an item is new.

<guid>http://some.server.com/weblogItem3207</guid>

There are no rules for the syntax of a guid. Aggregators must view them as a string. It's up to the source of the feed to establish the uniqueness of the string.

If the guid element has an attribute named "isPermaLink" with a value of true, the reader may assume that it is a permalink to the item, that is, a url that can be opened in a Web browser, that points to the full item described by the <item> element. An example:

<guid isPermaLink="true">http://inessential.com/2002/09/01.php#a2</guid>

isPermaLink is optional, its default value is true. If its value is false, the guid may not be assumed to be a url, or a url to anything in particular.[7]

2.3.13 <comments> sub-element of <item>

<comments> is an optional sub-element of <item>.
If present, it is the url of the comments page for the item[7]

<comments>http://rateyourmusic.com/yaccs/commentsn/blogId=705245&itemId=271</comments>

2.3.14 <author> sub-element of <item>

<author> is an optional sub-element of <item>.
It's the email address of the author of the item. For newspapers and magazines syndicating via RSS, the author is the person who wrote the article that the <item> describes. For collaborative weblogs, the author of the item might be different from the managing editor or webmaster. For a weblog authored by a single individual it would make sense to omit the <author> element.[7]

<author>lawyer@boyer.net (Lawyer Boyer)</author>

2.3.15 Comments

RSS places restrictions on the first non-whitespace characters of the data in <link> and <url> elements. The data in these elements must begin with an IANA-registered URI scheme, such as http://, https://, news://, mailto: and ftp://. Prior to RSS 2.0, the specification only allowed http:// and ftp://, however, in practice other URI schemes were in use by content developers and supported by aggregators. Aggregators may have limits on the URI schemes they support. Content developers should not assume that all aggregators support all schemes.[7]

In RSS 0.91, various elements are restricted to 500 or 100 characters. There can be no more than 15 <items> in a 0.91 <channel>. There are no string-length or XML-level limits in RSS 0.92 and greater. Processors may impose their own limits, and generators may have preferences that say no more than a certain number of <item>s can appear in a channel, or that strings are limited in length.[7]

In RSS 2.0, a provision is made for linking a channel to its identifier in a cataloging system, using the channel-level category feature, described above. For example, to link a channel to its Syndic8 identifier, include a category element as a sub-element of <channel>, with domain "Syndic8", and value the identifier for your channel in the Syndic8 database. The appropriate category element for Scripting News would be <category domain="Syndic8">1765</category>.

A frequently asked question about <guid>s is how do they compare to <link>s. Aren't they the same thing? Yes, in some content systems, and no in others. In some systems, <link> is a permalink to a weblog item. However, in other systems, each <item> is a synopsis of a longer article, <link> points to the article, and <guid> is the permalink to the weblog entry. In all cases, it's recommended that you provide the guid, and if possible make it a permalink. This enables aggregators to not repeat items, even if there have been editing changes.[7]

CHAPTER 3 – ANALYSIS

This chapter describe about analysis phase. Analysis phase will gather information from several sites that provide job and career information, all information gathered will be analyzed to get pattern that will be used in compose new RDF specification.

Every site has their own style to display the information e.g. there are sites provide information with HTML (Hyper Text Markup Language) or ASP (Active Server Pages) or PHP (PHP:Hypertext Preprocessor) or JSP (Java Server Pages) or even use RSS (Really Simple Syndication).

3.1 Gathering Information

This phase will gather information from several websites that provide job and career information, which are:

- http://www.unesco.org
- http://www.siemens.com
- http://www.sap.com
- http://www.jobsdb.com
- http://www.pickajob.com

3.1.1 Gathering Information from http://www.unesco.org

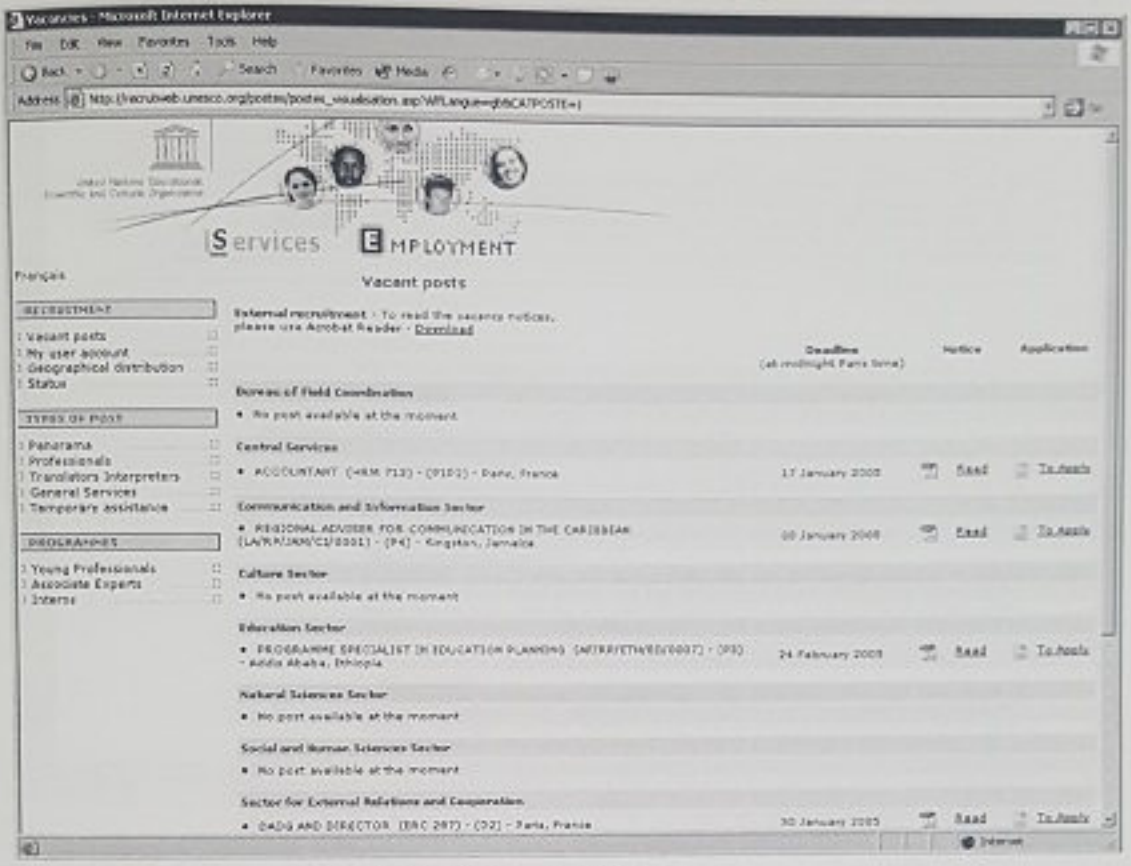

Figure 3.1 Screenshot Official Website of UNESCO

United Nations Educational, Scientific and Cultural Organization (UNESCO)[14], provide employment service to recruit and publish job vacancy from around the world.

Everyone can see and use this service, but people must register to this site if they want to apply for vacancies.

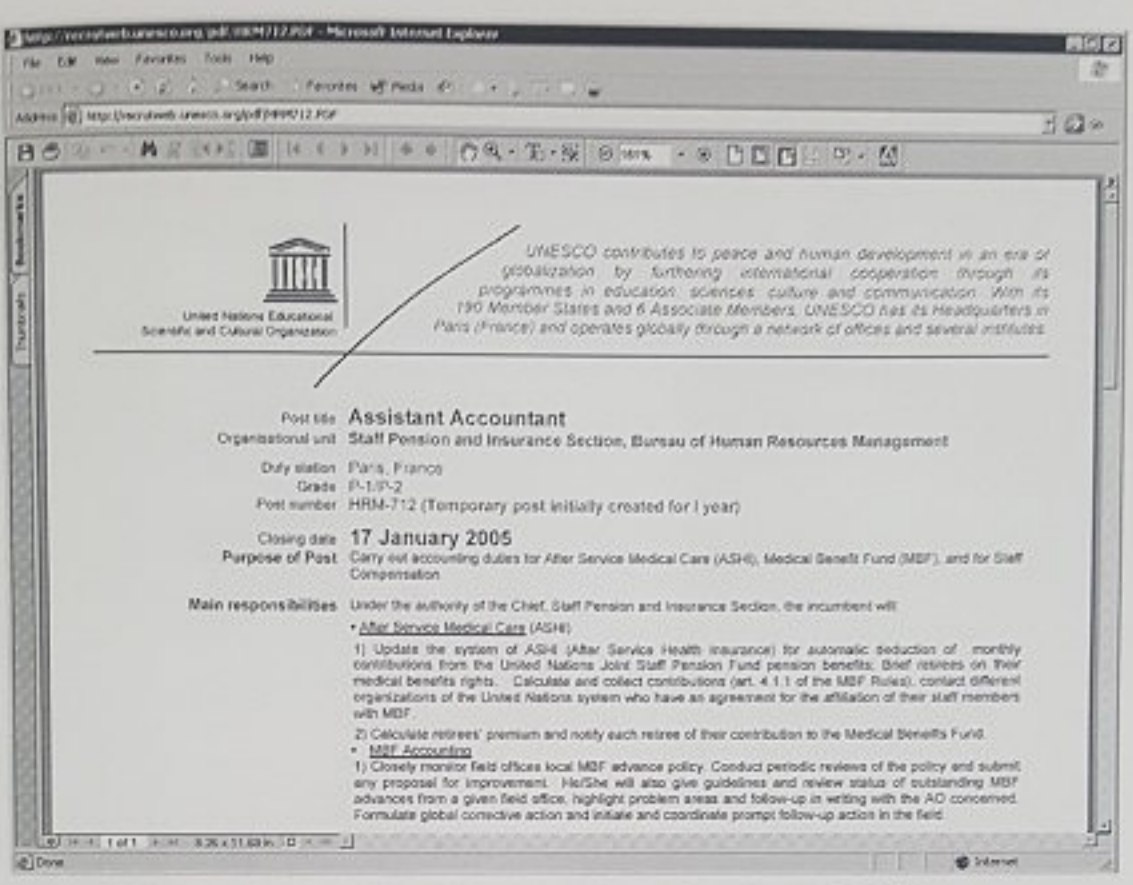

Figure 3.2 Screenshot one of Job Vacancy Information from UNESCO

Attributes from Job Vacancy at UNESCO

- Post Title
- Organisational unit
- Duty station
- Grade
- Post number
- Closing date
- Purpose of Post
- Main responsibility
- Profile
- Conditions of employment
- How to apply
- Notes

3.1.2 Gathering Information from http://www.siemens.com

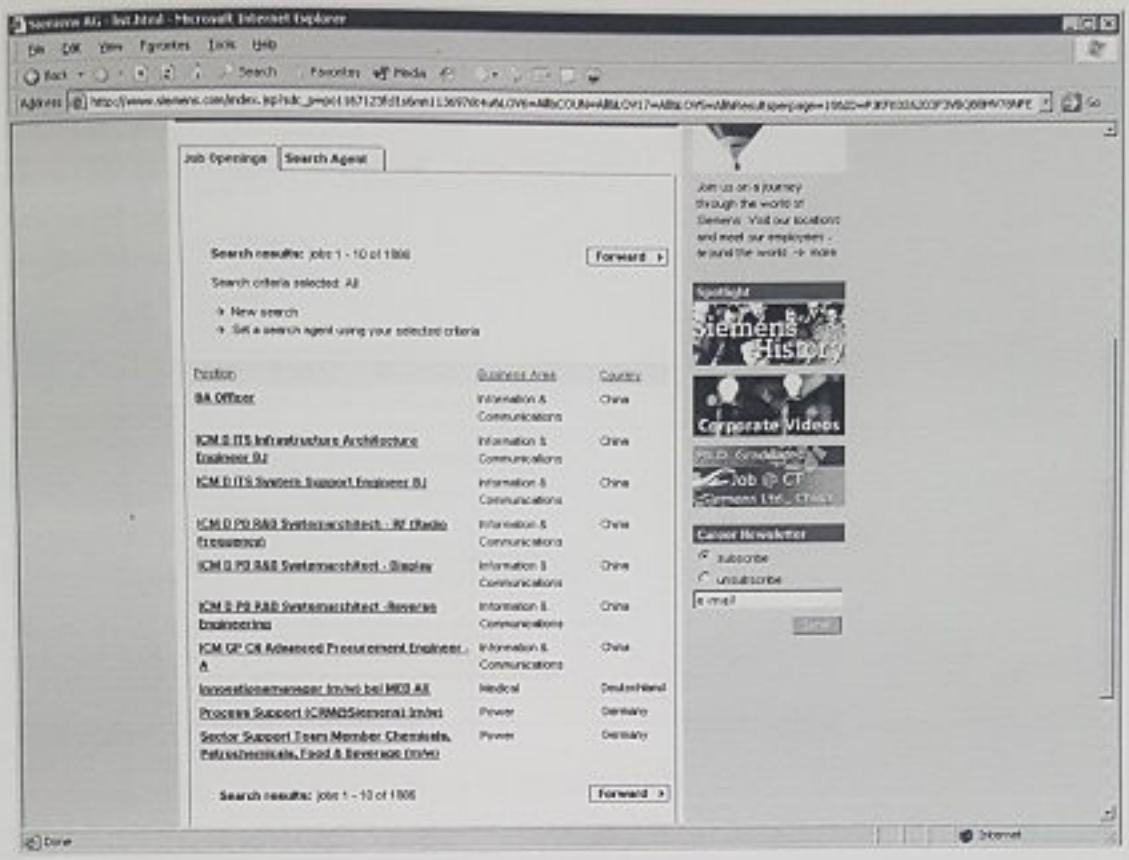

Figure 3.3 Screenshot Official Website of SIEMENS

SIEMENS AG[15] one of leading company in the world provide job information in their site, this is public service, people can access this site and can apply job vacancy directly without registration first.

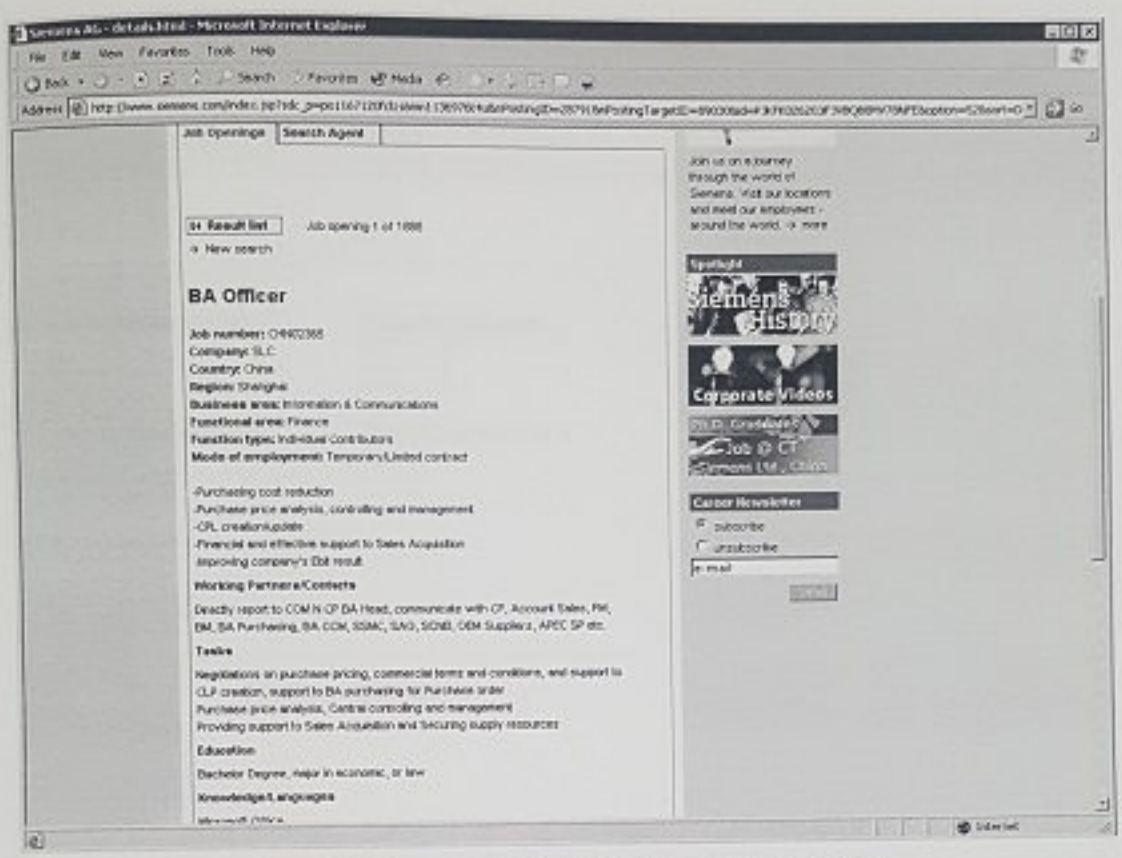

Figure 3.4 Screenshot Job Vacancy Information from siemens.com

Attributes from Job Vacancy at SIEMENS

- Job Title
- Job Number
- Country
- Region
- Business area
- Functional area
- Function type
- Mode of employment
- Working Partners/Contacts
- Tasks
- Education
- Knowledge/Language
- Experience
- Capabilities
- Additional Information

3.1.3 Gathering Information from http://www.sap.com

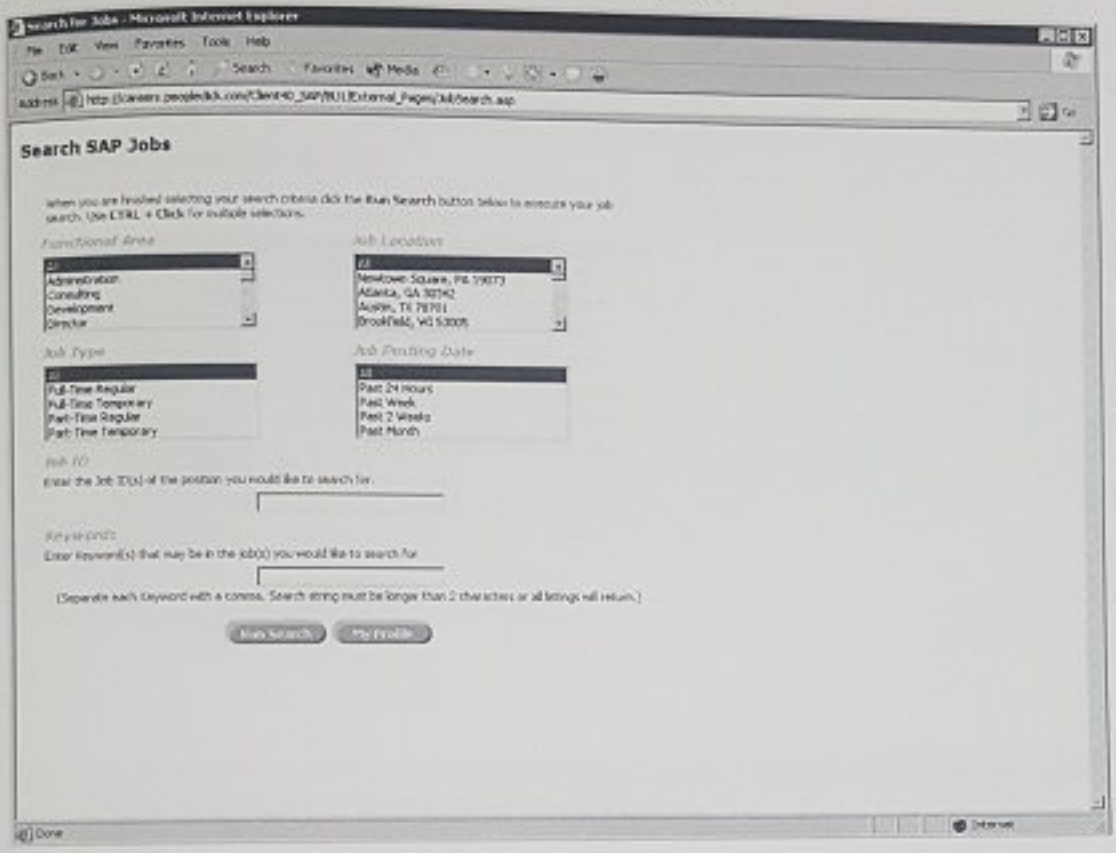

Figure 3.5 Screenshot job search from sap.com

Founded in 1972, SAP is the recognized leader in providing collaborative business solutions for all types of industries and for every major market.

With 12 million users, 84,000 installations, and 1,500 partners, SAP is the world's largest inter-enterprise software company and the world's third-largest independent software supplier overall. [8]

SAP provide job information in their website and public service, people can access information and apply online directly without registration.

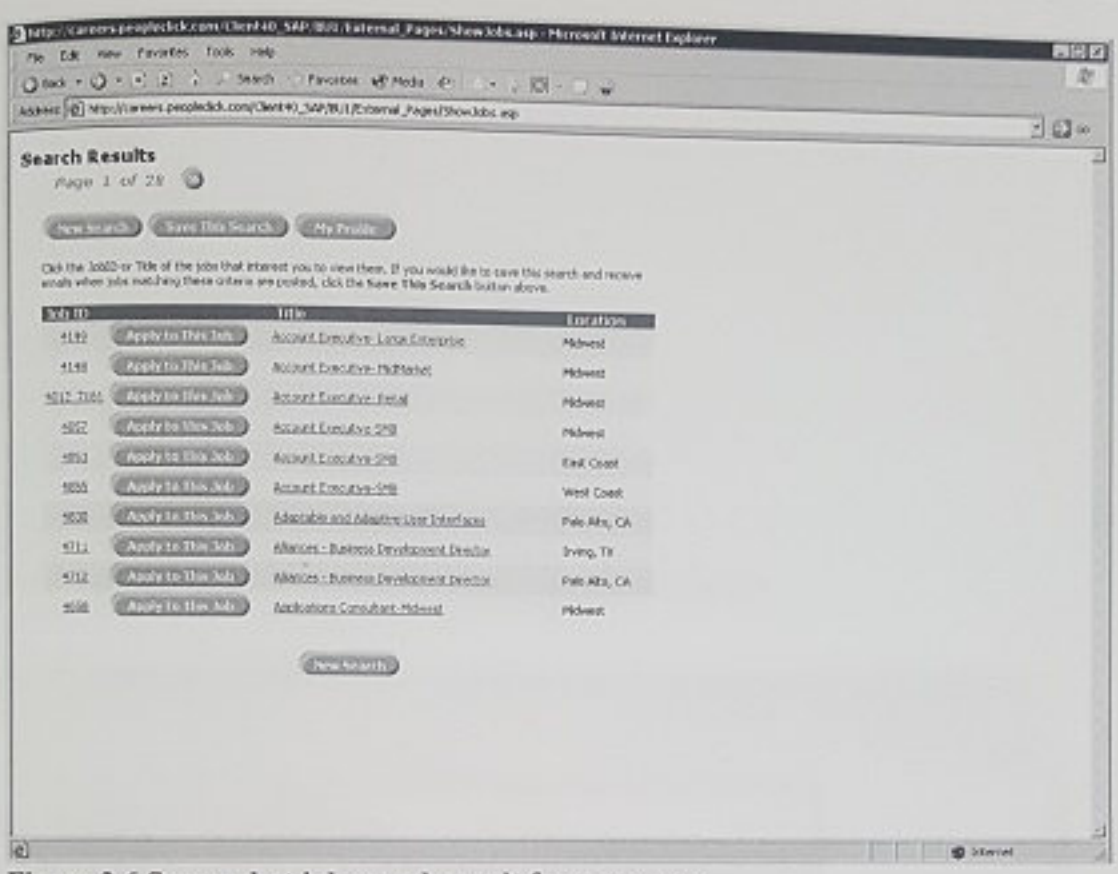

Figure 3.6 Screenshot job search result from sap.com

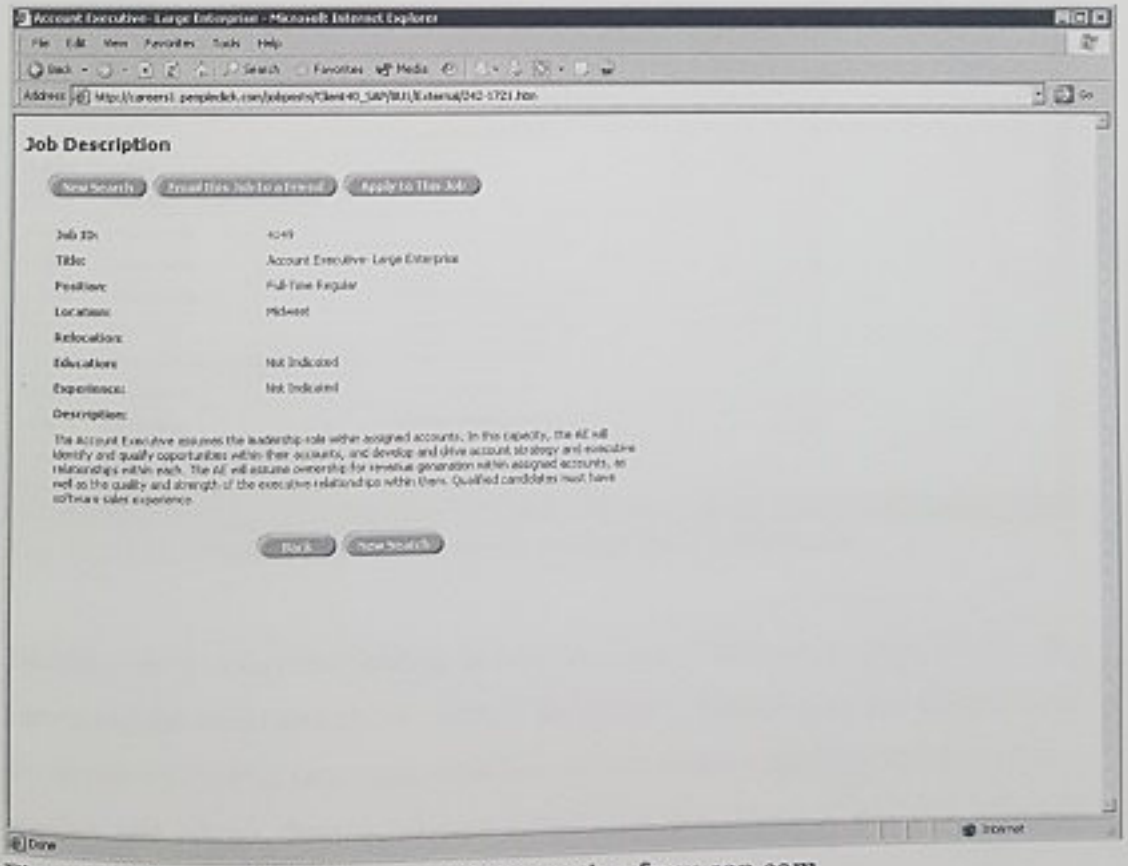

Figure 3.7 Screenshot job vacancy information from sap.com

Attributes from Job Vacancy at SAP

- Job ID
- Title
- Position
- Location
- Relocation
- Education
- Experience
- Description

3.1.4 Gathering Information from http://www.jobsdb.com

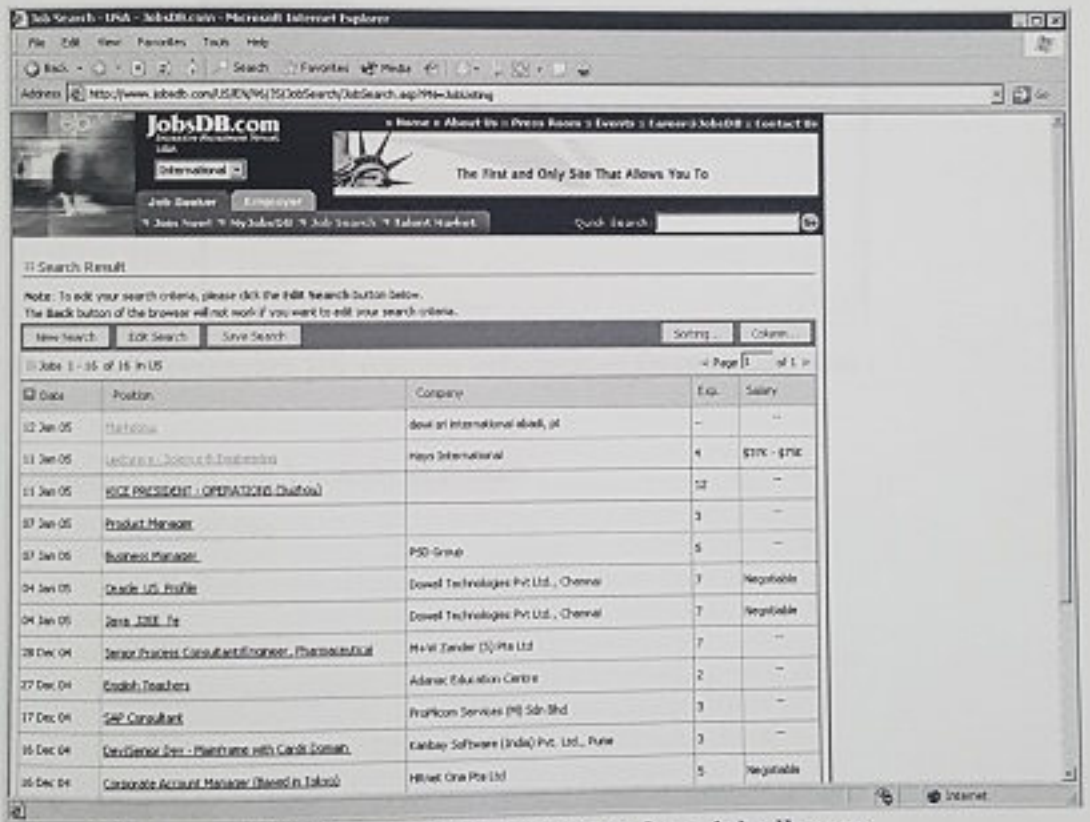

Figure 3.8 Screenshot job vacancy information from jobsdb.com

JobsDB.com is today the leading online recruitment network in Asia Pacific. By harnessing the speed and global reach of the Internet, JobsDB.com has designed and developed a powerful recruitment medium, which brings employers and job seekers together and allows them to interact directly for fast, efficient and cost effective recruitment. [9]

Jobsdb provide job information in their website and public service, if visitor want to apply job, they have to register first

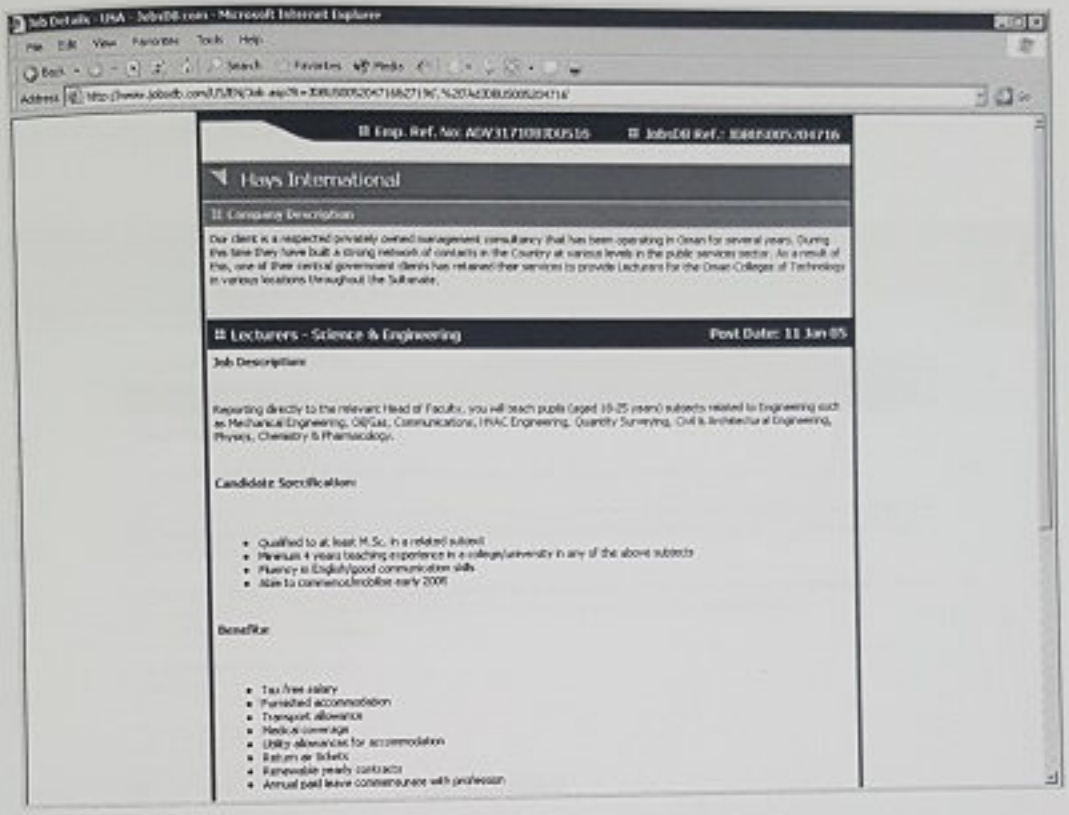

Figure 3.9 Screenshot job vacancy information from jobsdb.com

Attributes from Job Vacancy at jobsdb

- Emp. Ref. No
- Jobsdb Ref
- Company Name
- Company Description
- Job Title
- Post Date
- Job Description
- Candidate Specification
- Benefits
- Closing Date
- How to apply
- Qualifications

- Years Experience
- State
- Salary
- Job Type

3.1.5 Gathering Information from http://www.pickajob.com

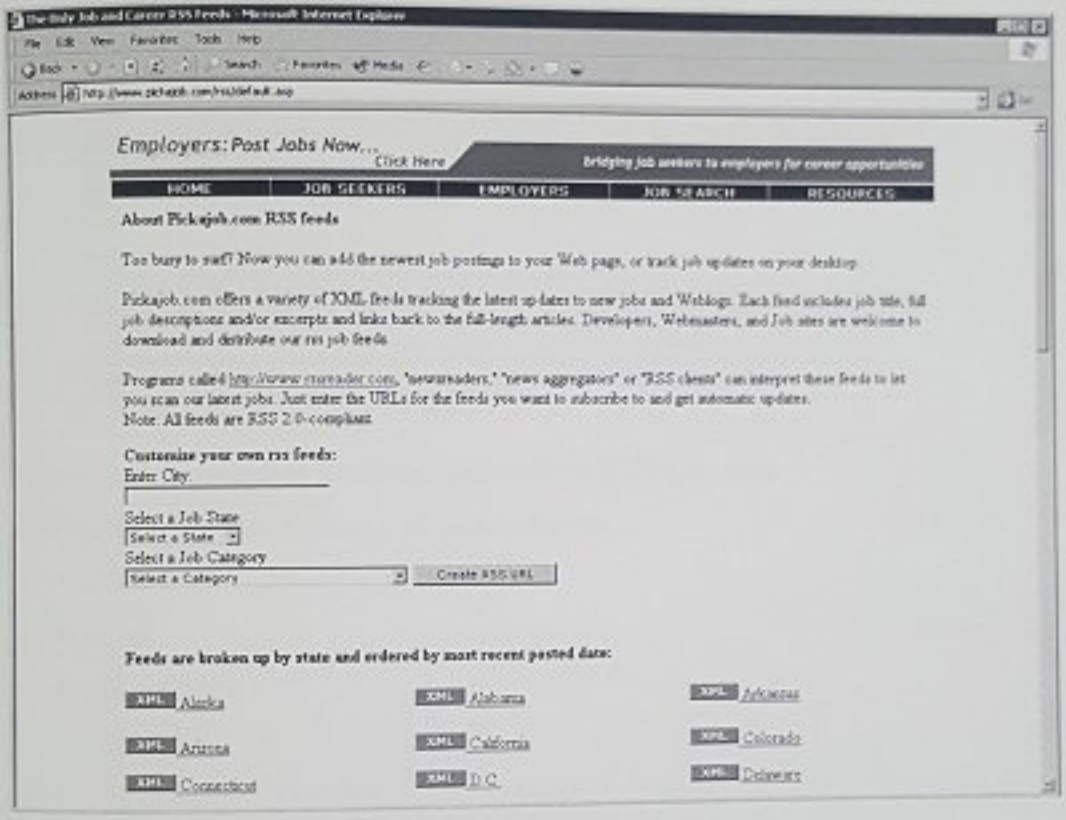

Figure 3.10 Screenshot job information pickajob.com

Pickajob.com is a subsidiary of ROI GROUP INC. based in San Diego. With years of experience in the employment management industry, Pickajob's management team have set out to revolutionize the job listing and recruiting experience. Our business model seeks to reduce not only the cost of running a company, but the cost of its products and services as well.[10]

Pickajob.com provide job information using RSS feeder, pickajob.com have their own style to display the information.

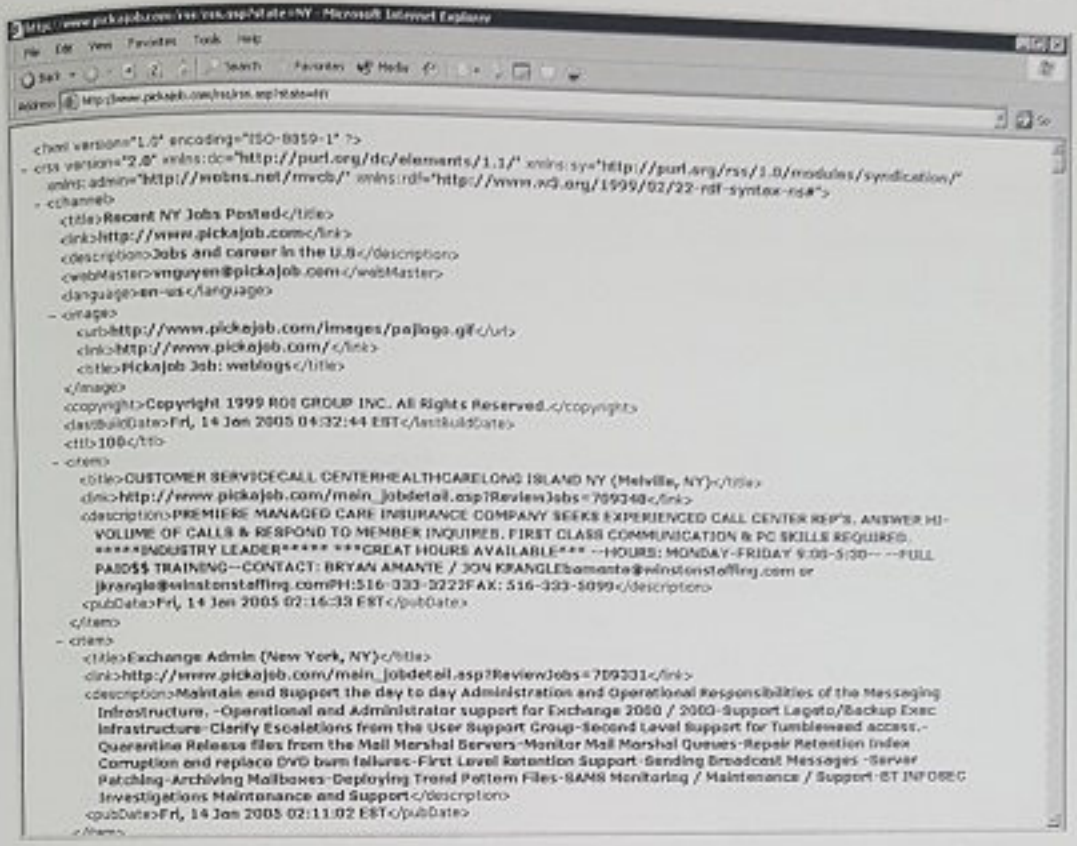

Figure 3.11 Screenshot job information from pickajob.com

Attributes from Job Vacancy at pickajob.com

- Title
- Link
- Description
- PubDate

CHAPTER 4 – RESULT AND DESIGN

4.1 Result Analysis

After gather the information from the Internet, the result is there is no formal standard for displaying about job information, everyone is free to decide what information will be displayed and this fact is amusing for visitor or even bots/search engine to get information.

UNESCO	SIEMENS	SAP	JOBSDB	PICKAJOB
• Post Title • Organisational unit • Duty station • Grade • Post number • Closing date • Purpose of Post • Main responsibility • Profile • Conditions of employment • How to apply • Notes	• Job Title • Job Number • Country • Region • Business area • Functional area • Function type • Mode of employment • Working Partners/Contacts • Tasks • Education • Knowledge/Language • Experience • Capabilities • Additional Information	• Job ID • Title • Position • Location • Relocation • Education • Experience • Description	• Emp. Ref. No • Jobsdb Ref • Company Name • Company Description • Job Title • Post Date • Job Description • Candidate Specification • Benefits • Closing Date • How to apply • Qualifications • Years Experience • State • Salary • Job Type	• Title • Link • Description • PubDate

Table 4.1 Result Analysis

Search engine will face the big problem when trying to grab job information from Internet because the information comes vary from one site to another. Search engine can only grab the whole page not the essential information, search engine have difficulty to find a pattern from one site to another.
Visitor/job seeker may feel confuse getting information from Internet because job information is different from one site to another.

The answer for the problems faced is creating pattern to display job information that can be used for everyone which use portable technology so it can exchange between

different types of computers using different types of operating system and application languages, actually there is RSS and RDF based on XML technology.

4.2 Job Information Grabber Prototype

This chapter will build a job information grabber prototype to retrieve job information from specific site using Java programming language.

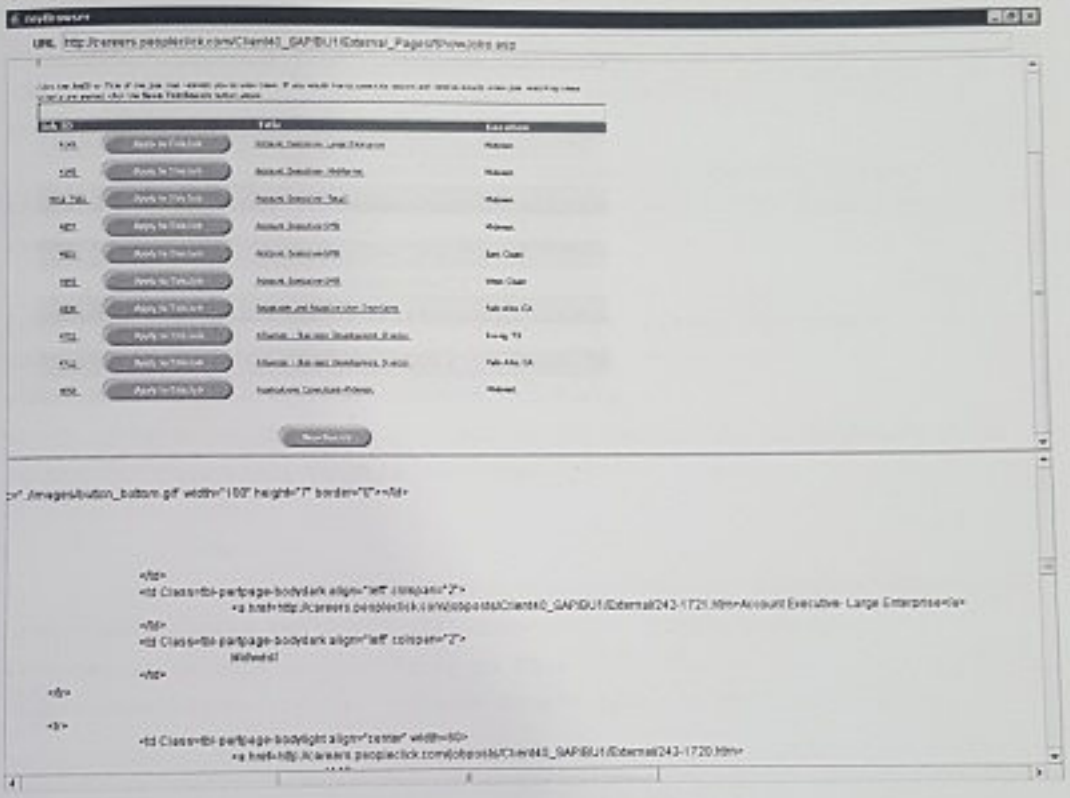

Figure 4.1 Browsing specific URL

To grab specific information from specific site require code analysis by human to find a pattern and parse the code to get our need. From the figure above we will find about job information and the link, the codes are:

```
<td Class=tbl-partpage-bodydark align="center" width=60>
                                                        <a
href=http://careers.peopleclick.com/jobposts/Client40_SAP/BU1/External/242-
1721.htm>

      4149
                                                        </a>
                                                    </td>
                                                    <td Class=tbl-
partpage-bodydark align="Left"  width=150>
```

```
      <table border="0" cellpadding="0" cellspacing="0">
            <tr>
                  <td rowspan="3"><a
href='NewCandidate.asp?Jobid=1721&Quest=Sales' class='form-button'
onMouseOver="window.status='Apply to This Job';return true"
onMouseOut="window.status='';return true"><img
src="../images/button_left.gif" width="16" height="27" align="top"
border="0" alt="Apply to This Job"></a></td>
                  <td align="center" valign="top" bgcolor="#6699cc"
height="7"><img src="../images/button_top.gif" width="100" height="7"
border="0"></td>
                  <td rowspan="3"><a
href='NewCandidate.asp?Jobid=1721&Quest=Sales' class='form-button'
onMouseOver="window.status='Apply to This Job';return true"
onMouseOut="window.status='';return true"><img
src="../images/button_right.gif" width="16" height="27" align="top"
border="0" alt="Apply to This Job"></a></td>
            </TR>
            <TR>
                  <td align="center" valign="middle" bgcolor="#6699cc"><a
href='NewCandidate.asp?Jobid=1721&Quest=Sales' class='form-button'
onMouseOver="window.status='Apply to This Job';return true"
onMouseOut="window.status='';return true" >Apply to This Job</a></td>

            </TR>
            <TR>
                  <td align="center" valign="bottom" bgcolor="#6699cc"
height="7"><img src="../images/button_bottom.gif" width="100" height="7"
border="0"></td>
            </tr>
      </table>


                                                      </td>

                                                      <td Class=tbl-
partpage-bodydark align="left" colspan="2">

                                                            <a
href=http://careers.peopleclick.com/jobposts/Client40_SAP/BU1/External/242-
```

```
1721.htm>Account Executive- Large Enterprise</a>

                                                                    </td>
                                                                    <td Class=tbl-
partpage-bodydark align="left" colspan="2">

                                                                          Midwest
                                                                    </td>

                                                              </tr>

                                                              <tr>

                                                                    <td Class=tbl-
partpage-bodylight align="center" width=60>
                                                                          <a
href=http://careers.peopleclick.com/jobposts/Client40_SAP/BU1/External/242-
1720.htm>

      4148
                                                                          </a>
                                                                    </td>
                                                                    <td Class=tbl-
partpage-bodylight align="Left"  width=150>




      <table border="0" cellpadding="0" cellspacing="0">
            <tr>
                  <td rowspan="3"><a
href='NewCandidate.asp?Jobid=1720&Quest=Sales' class='form-button'
onMouseOver="window.status='Apply to This Job';return true"
onMouseOut="window.status='';return true"><img
src="../images/button_left.gif" width="16" height="27" align="top"
border="0" alt="Apply to This Job"></a></td>
                  <td align="center" valign="top" bgcolor="#6699cc"
height="7"><img src="../images/button_top.gif" width="100" height="7"
border="0"></td>
                  <td rowspan="3"><a
href='NewCandidate.asp?Jobid=1720&Quest=Sales' class='form-button'
onMouseOver="window.status='Apply to This Job';return true"
onMouseOut="window.status='';return true"><img
src="../images/button_right.gif" width="16" height="27" align="top"
border="0" alt="Apply to This Job"></a></td>
```

```
            </TR>
            <TR>
                  <td align="center" valign="middle" bgcolor="#6699cc"><a
href='NewCandidate.asp?Jobid=1720&Quest=Sales' class='form-button'
onMouseOver="window.status='Apply to This Job';return true"
onMouseOut="window.status='';return true" >Apply to This Job</a></td>

            </TR>
            <TR>
                  <td align="center" valign="bottom" bgcolor="#6699cc"
height="7"><img src="../images/button_bottom.gif" width="100" height="7"
border="0"></td>
             </tr>
      </table>


                                                              </td>

                                                              <td Class=tbl-
partpage-bodylight align="left" colspan="2">

                                                                    <a
href=http://careers.peopleclick.com/jobposts/Client40_SAP/BU1/External/242-
1720.htm>Account Executive- MidMarket</a>

                                                              </td>
                                                              <td Class=tbl-
partpage-bodylight align="left" colspan="2">

                                                                    Midwest
                                                              </td>

                                                        </tr>
```

Figure 4.2 Chunks of Source Code

The result from codes above is:

4149	Apply to This Job	Account Executive- Large Enterprise	Midwest
4148	Apply to This Job	Account Executive- MidMarket	Midwest

Figure 4.3 Result of Source Code

Information needed is:

Title	Link
Account Executive-Large Enterprise	http://careers1.peopleclick.com/jobposts/Client40_SAP/BU1/External/242-1721.htm
Account Executive-MidMarket	http://careers1.peopleclick.com/jobposts/Client40_SAP/BU1/External/242-1720.htm

Table 4.2 Information Needed

Each website has their own code to represent job information and it has to analyze to get the information.

4.2.1 Flowchart of Job Information Grabber Prototype

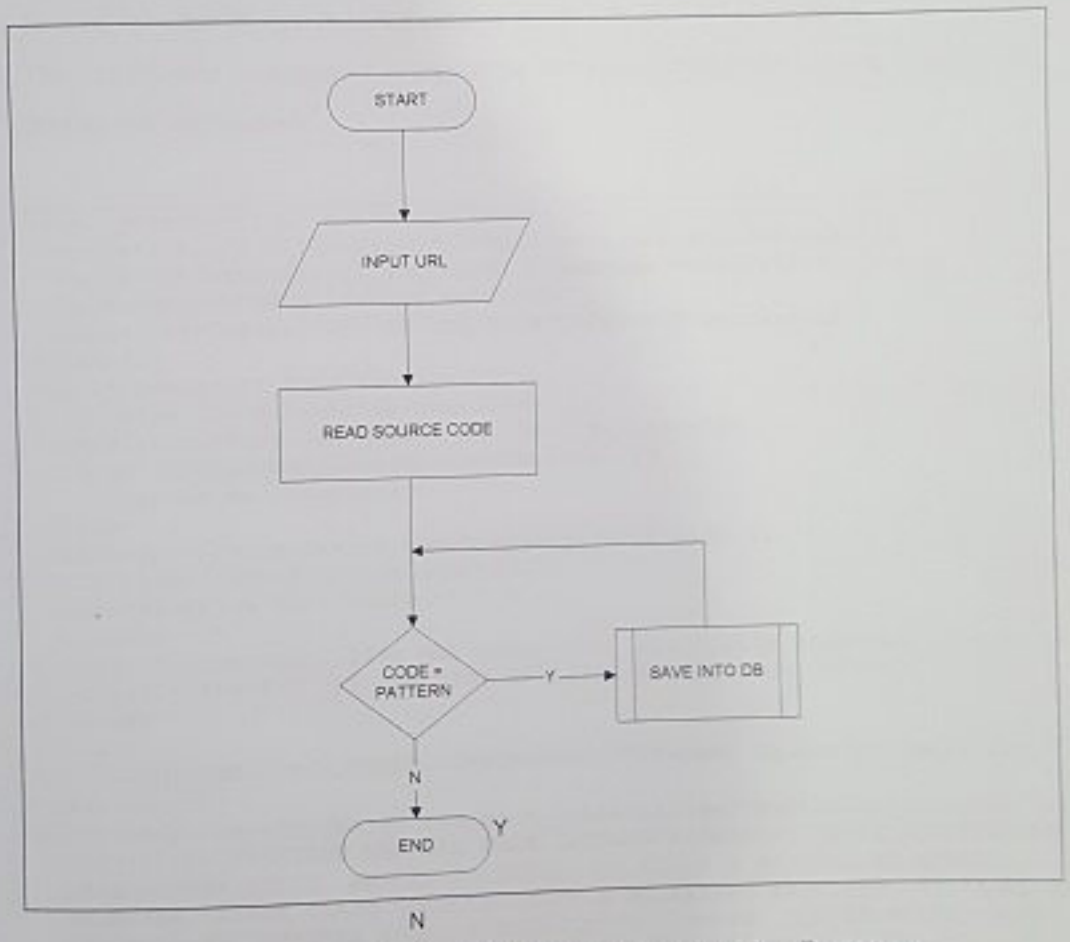

Figure 4.4 Flowchart of Job Information Grabber Prototype

4.2.2 Job Information Grabber Prototype Implementation

Software:

- Java SDK 1.5 (http://java.sun.com)
- Windows XP Professional (http://www.microsoft.com)

Application has been tested using:

- Toshiba Notebook Satellite A10 Pentium 4M 2.0GHz 512MB RAM

Disadvantage from the application is source code has to be analyzed by human to find a pattern, because every website have a different approach.

4.3 Design Pattern Using RSS

RSS is a format for syndicating news and the content of news-like sites, even RSS can be used for other things, the RSS schema doesn't fit for job information.

The code below is a part of source code from www.pickajob.com that use RSS to display job information.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
  xmlns:admin="http://webns.net/mvcb/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<channel>
<title>Recent NY Jobs Posted</title>
<link>http://www.pickajob.com</link>
<description>Jobs and career in the U.S</description>
<webMaster>vnguyen@pickajob.com</webMaster>
<language>en-us</language>
<image>
<url>http://www.pickajob.com/images/pajlogo.gif</url>
<link>http://www.pickajob.com/</link>
<title>Pickajob Job: weblogs</title>
  </image>
<copyright>Copyright 1999 ROI GROUP INC. All Rights Reserved.</copyright>
<lastBuildDate>Fri, 14 Jan 2005 07:12:13 EST</lastBuildDate>
<ttl>100</ttl>
<item>
<title>CUSTOMER SERVICECALL CENTERHEALTHCARELONG ISLAND NY (Melville,
  NY)</title>
<link>http://www.pickajob.com/main_jobdetail.asp?ReviewJobs=709348</link>
<description>PREMIERE MANAGED CARE INSURANCE COMPANY SEEKS EXPERIENCED
  CALL CENTER REP'S. ANSWER HI-VOLUME OF CALLS & RESPOND TO MEMBER
  INQUIRES. FIRST CLASS COMMUNICATION & PC SKILLS REQUIRED. *****INDUSTRY
  LEADER***** ***GREAT HOURS AVAILABLE*** --HOURS: MONDAY-FRIDAY 9:00-
  5:30-- --FULL PAID$$ TRAINING--CONTACT: BRYAN AMANTE / JON
  KRANGLEbamante@winstonstaffing.com or
  jkrangle@winstonstaffing.comPH:516-333-3222FAX: 516-333-
  5099</description>
<pubDate>Fri, 14 Jan 2005 02:16:33 EST</pubDate>
```

```
</item>
<item>
<title>Exchange Admin (New York, NY)</title>
<link>http://www.pickajob.com/main_jobdetail.asp?ReviewJobs=709331</link>
<description>Maintain and Support the day to day Administration and
  Operational Responsibilities of the Messaging Infrastructure. -
  Operational and Administrator support for Exchange 2000 / 2003-Support
  Legato/Backup Exec infrastructure-Clarify Escalations from the User
  Support Group-Second Level Support for Tumbleweed access.-Quarantine
  Release files from the Mail Marshal Servers-Monitor Mail Marshal
  Queues-Repair Retention Index Corruption and replace DVD burn failures-
  First Level Retention Support-Sending Broadcast Messages -Server
  Patching-Archiving Mailboxes-Deploying Trend Pattern Files-SAMS
  Monitoring / Maintenance / Support-BT INFOSEC Investigations
  Maintenance and Support</description>
<pubDate>Fri, 14 Jan 2005 02:11:02 EST</pubDate>
</item>
..
..
< /channel>
</rss>
```

Figure 4.5 Pickajob Source Code

This code is well formed we can easily grab the information, the main part of information is:

```
<item>
 <title> ..  </title>
 <link> .. </link>
 <description> .. </description>
 <pubDate> .. </pubDate>
</item>
```

From the code above the `<description>` tag holds a lot of content and plays a major function. Putting all information between `<description>` tag will making confuse for the visitor/reader and the RSS creator still free to decide what will be informed in `<description>` tag, so there is no standard.

Because RSS has their own specification (see chapter 2.3), we can't add any new attribute in RSS file.

4.4 Design Pattern Using RDF

The Resource Description Framework (RDF) is a language for representing information about web resources, such as content description, title, author, copyright information, availability schedules, and more.[5]

RDF is designed to be read by computers, not designed for being displayed to people, RDF seems to be a perfect tools for representing job information, RDF has been used for describing network objects, suitable for use by network search engines named The Dublin Core.

The Dublin Core is a set of predefined properties for describing documents.

Property	Definition
Contributor	An entity responsible for making contributions to the content of the resource
Coverage	The extent or scope of the content of the resource
Creator	An entity primarily responsible for making the content of the resource
Format	The physical or digital manifestation of the resource
Date	A date of an event in the lifecycle of the resource
Description	An account of the content of the resource
Identifier	An unambiguous reference to the resource within a given context
Language	A language of the intellectual content of the resource
Publisher	An entity responsible for making the resource available
Relation	A reference to a related resource
Rights	Information about rights held in and over the resource
Source	A Reference to a resource from which the present resource is derived
Subject	A topic of the content of the resource
Title	A name given to the resource
Type	The nature or genre of the content of the resource

Table 4.3 The Dublin Core RDF

This is representation of Dublin Core in RDF:[5]

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc= "http://purl.org/dc/elements/1.1/">
<rdf:Description rdf:about="http://www.w3schools.com">
<dc:title>D-Lib Program - Research in Digital Libraries</dc:title>
<dc:description>
W3Schools is large collection of web building tutorials
</dc:description>
<dc:publisher>Refsnes Data</dc:publisher>
<dc:date>1999-09-01</dc:date>
<dc:type>World Wide Web Home Page</dc:type>
<dc:format>text/html</dc:format>
<dc:language>en</dc:language>
```

```
</rdf:Description>
</rdf:RDF>
```

4.5 Job Information Pattern Design

Based on analysis, here are mandatory field to build job item information:

- Company
- Job title
- Job Description
- Posting date
- Closing date
- Location
- Job Type
- Education
- Qualification
- Experience
- How to apply
- Link
- Notes

Representation with RDF is:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:job="http://www.job.fake/job">
<rdf:Description
 rdf:about="http://www.job.fake/job/company">
  <job:title> </job:title>
  <job:description> </job:description>
  <job:postdate> </job:postdate>
  <job:closingdate> </job:closingdate>
  <job:location> </job:location>
  <job:type> </job:type>
  <job:education> </job:education>
  <job:qualification> </job:qualification>
  <job:experience> </job:experience>
  <job:apply> </job:apply>
  <job:link> </job:link>
```

```
 <job:notes> </job:notes>
</rdf:Description>
</rdf:RDF>
```

Representation in XML is:

```
<?xml version="1.0"?>
<!DOCTYPE item [
  <!ELEMENT item
(company,title,description,postdate,closingdate,location,type,educati
on,qualification,experience,apply,link,notes)>
   <!ELEMENT company          (#PCDATA)>
   <!ELEMENT title            (#PCDATA)>
   <!ELEMENT description      (#PCDATA)>
   <!ELEMENT postdate         (#PCDATA)>
   <!ELEMENT closingdate      (#PCDATA)>
   <!ELEMENT location         (#PCDATA)>
   <!ELEMENT type             (#PCDATA)>
   <!ELEMENT education        (#PCDATA)>
   <!ELEMENT qualification    (#PCDATA)>
   <!ELEMENT experience       (#PCDATA)>
   <!ELEMENT apply            (#PCDATA)>
   <!ELEMENT link             (#PCDATA)>
   <!ELEMENT notes            (#PCDATA)>
  ]>
  <item>
     <company></company>
     <title> </title>
     <description> </description>
     <postdate> </postdate>
     <closingdate> </closingdate>
     <location> </location>
     <type> </type>
     <education> </education>
     <qualification> </qualification>
     <experience> </experience>
     <apply> </apply>
     <link> </link>
     <notes> </notes>
   </item>
```

Both pattern above using XML based technology, it means we can easy to maintain the information and exchange between application, so it will simplify for visitor, search engine and site owner to:

- Display job information
- Crawl job information
- Exchange job information between site

4.6 Implementation Job Information Pattern

4.6.1 Flowchart Job Information Application

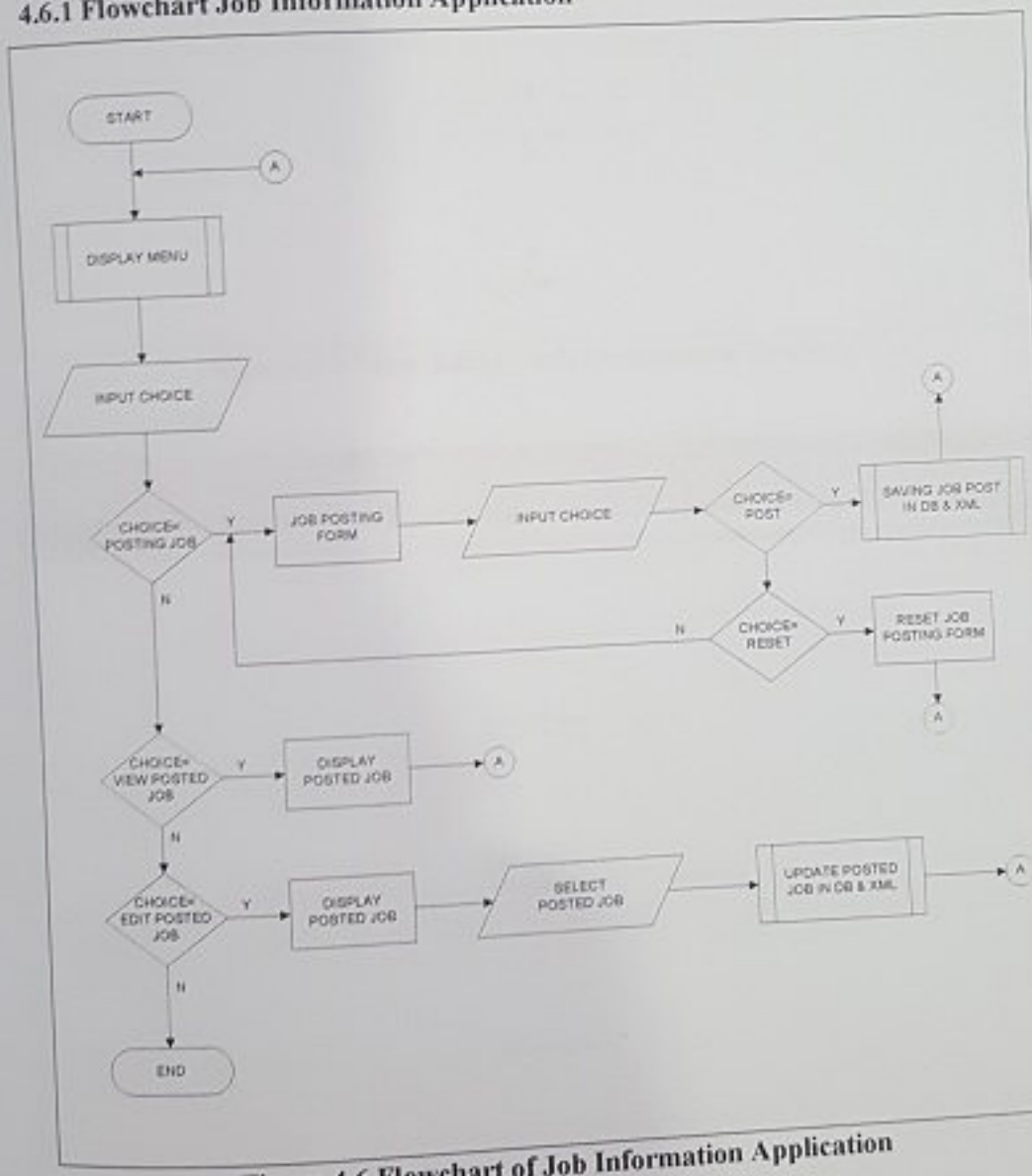

Figure 4.6 Flowchart of Job Information Application

4.6.2 Flowchart Job Information Grabber

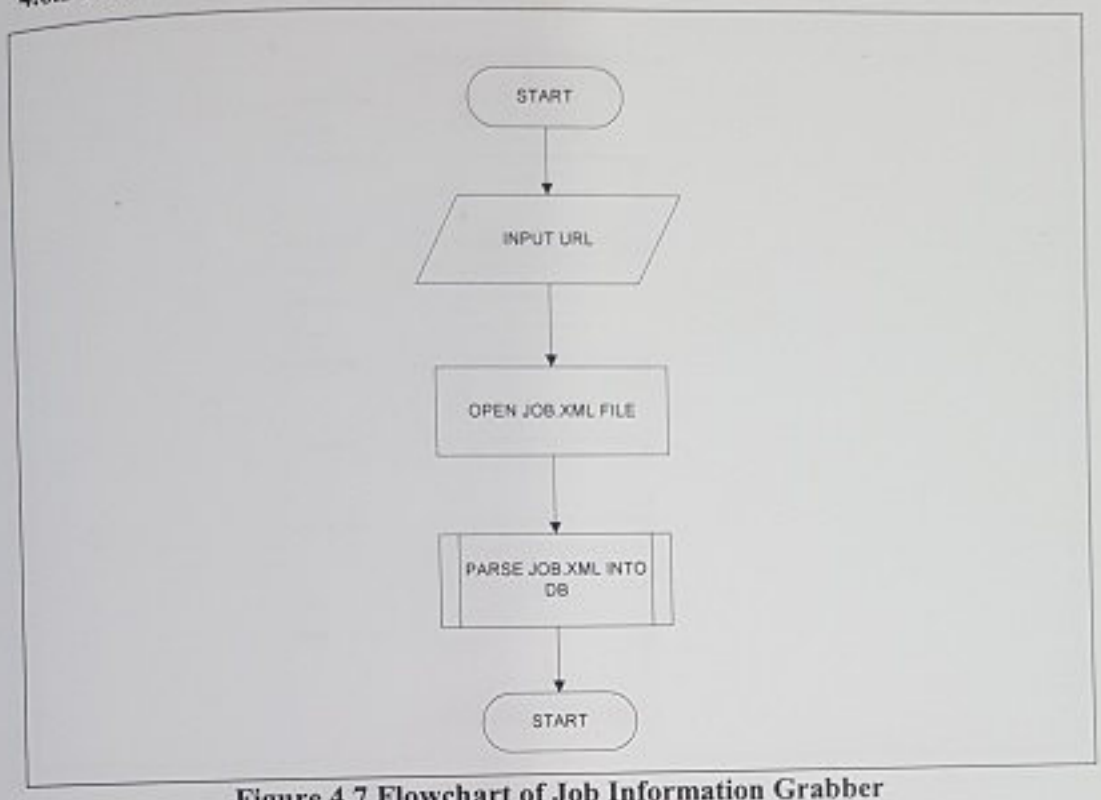

Figure 4.7 Flowchart of Job Information Grabber

4.6.3 Screenshot

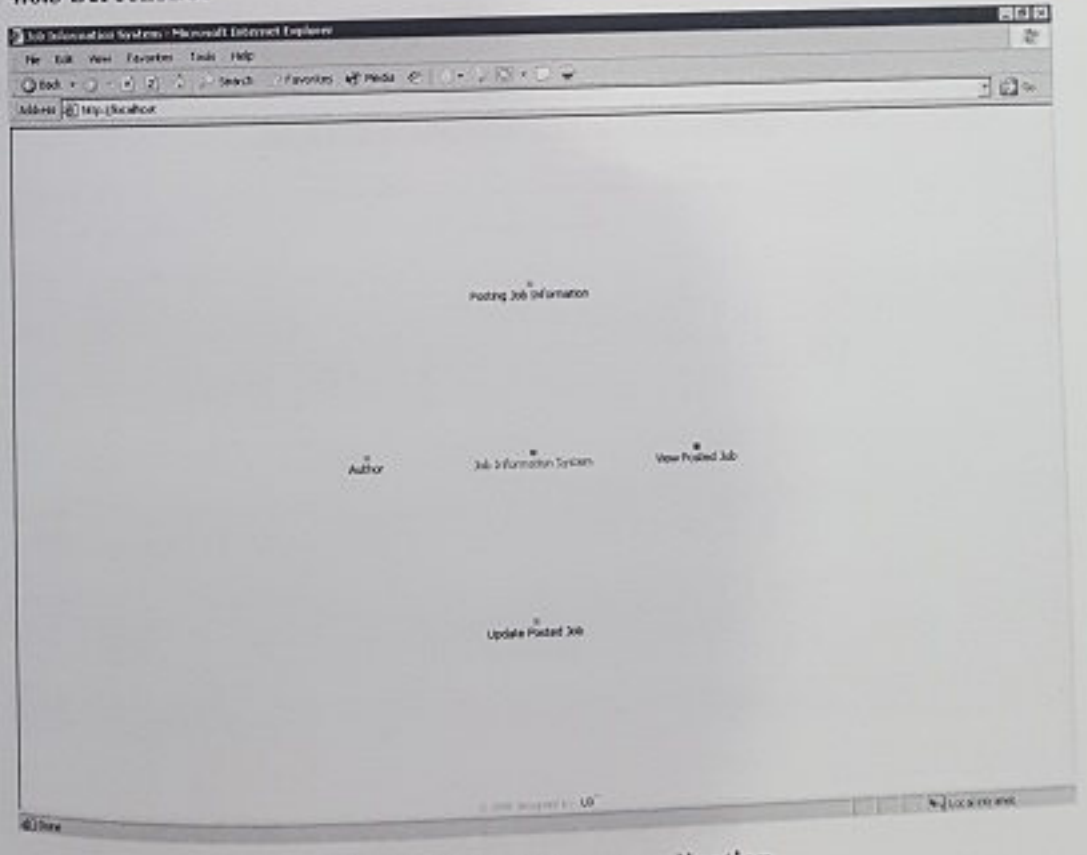

Figure 4.8 Main Page Application

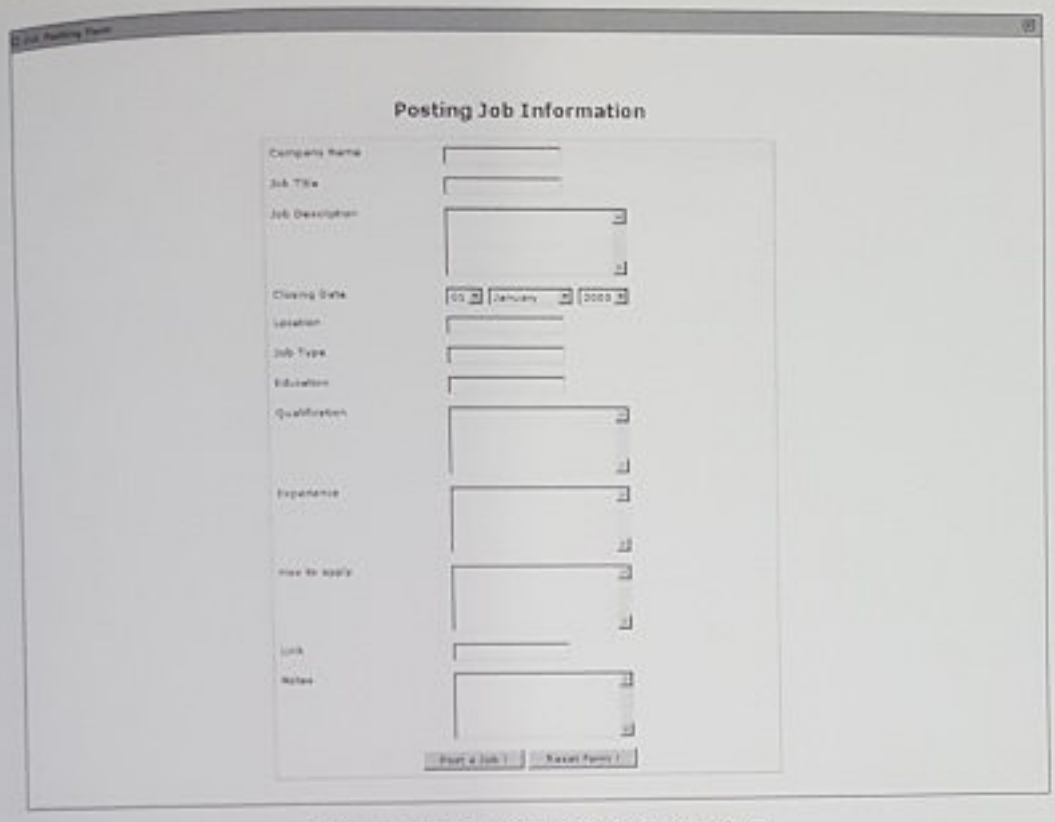

Figure 4.9 Posting Job Information

Figure 4.10 View Posted Job

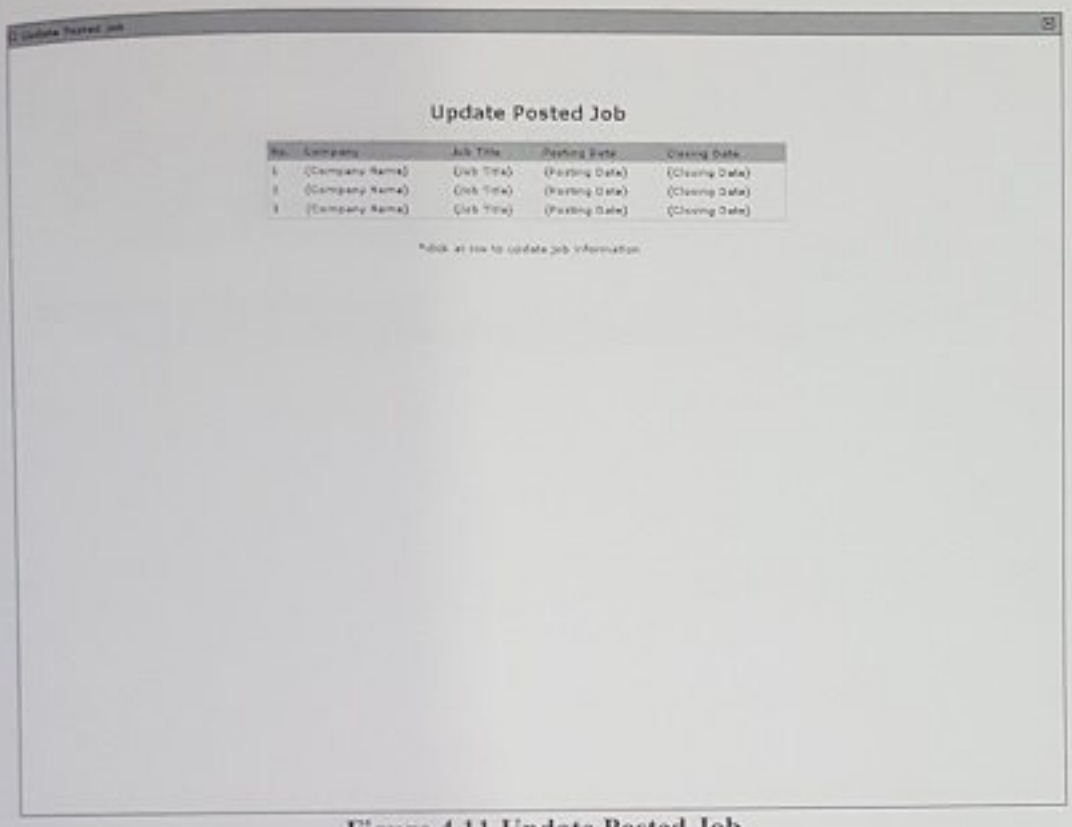

Figure 4.11 Update Posted Job

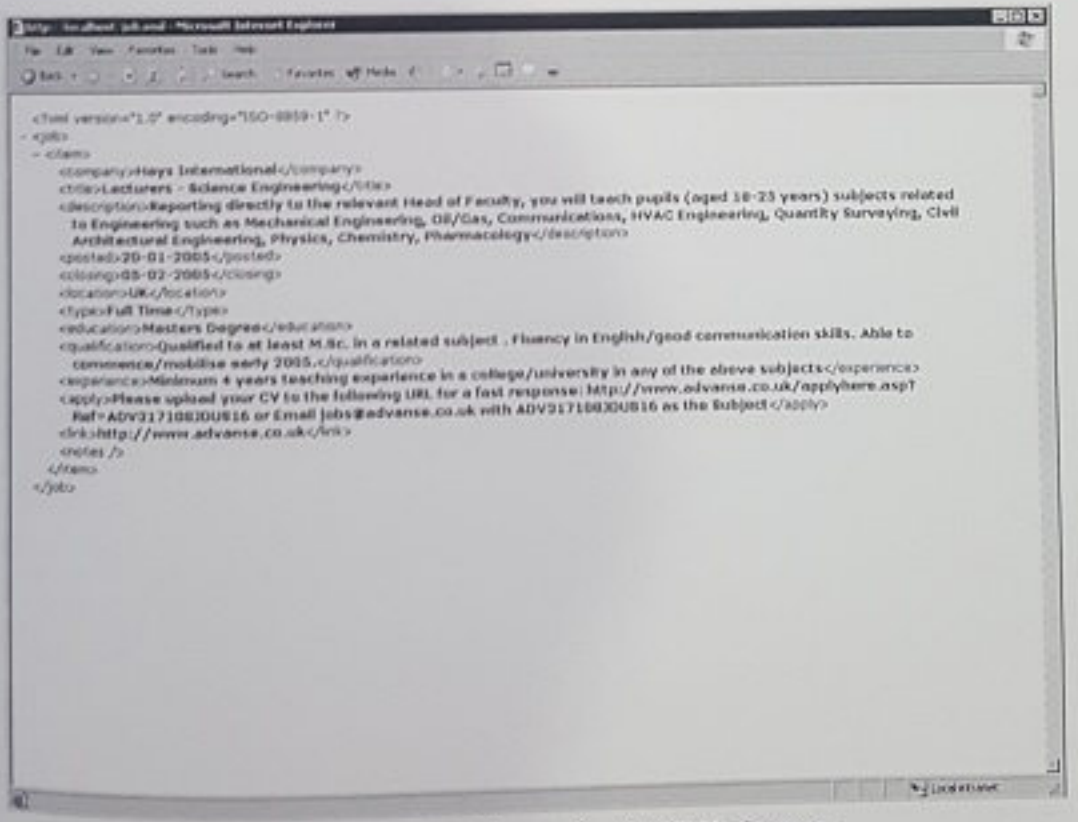

Figure 4.12 Job Information in XML format

4.6.4 Job Information Implementation

Software:

- Apache Web Server (http://apache.org)
- PHP version 4.3.4 (http://www.php.net)
- MySQL Database (http://www.mysql.com)
- Windows XP Operating System

Application had been tested using:

- Toshiba Notebook Satellite A10 Pentium 4M 2.0GHz 512MB RAM

CHAPTER 5 – CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

Internet provides a lot of information, there are many recommendation and standard from World Wide Web Consortium (W3C-http://www.w3.org) about displaying information in Internet.

Right now there are RSS and RDF. RSS stands for "Really Simple Syndication" and is a dialect of XML created to allow lists of information, known as "feeds", to be published by content producers and subscribed to by readers. [6]
RDF stands for "Resource Description Framework" a language for representing information about web resources.[5] Both based on XML Technology.

There are many information provided by website that different from each other, e.g job information and there is no standard to display job information, this is quite interesting because job information is one of important information that looking for by visitor in Internet. What happened is people get different information about job from one site to another, many site that provide content about job information can't exchange their information to another, special search engine which crawl job information have to hard work to crawl the information, this is because there is no formal standard to handle this situation.

One of solution is define new standard or extend from the legacy standard to handle job information display to make easier for people gathered job information using XML based technology and what is done this thesis are:

- Create JIP (Job Information Pattern) and build the application for JIP
- Build JIGrab protoype (Job Information Grabber) for grabbing XML from site who implement JIP

5.2 RECOMMENDATION

This thesis is only analyze job information from several information and tries to design pattern for displaying job information based on XML technology.

For further work there must be a group of company or organisation to research about displaying job information and define a new standard that can be used together, like dublin core [4].

For further work, creating DTD and XML schema for information job and curriculum vitae is recommended, so it will become a standard.

GLOSSARY

ASP : An Active Server Page (ASP) is an HTML page that includes one or more scripts (small embedded programs) that are processed on a Microsoft Web server before the page is sent to the user. An ASP is somewhat similar to a server-side include or a common gateway interface (CGI) application in that all involve programs that run on the server, usually tailoring a page for the user. Typically, the script in the Web page at the server uses input received as the result of the user's request for the page to access data from a database and then builds or customizes the page on the fly before sending it to the requestor. [11]

DTD : A Document Type Definition (DTD) is a specific document defining and constraining definition or set of statements that follow the rules of the Standard Generalized Markup Language (SGML) or of the Extensible Markup Language (XML), a subset of SGML. A DTD is a specification that accompanies a document and identifies what the funny little codes (or markup) are that, in the case of a text document, separate paragraphs, identify topic headings, and so forth and how each is to be processed. By mailing a DTD with a document, any location that has a DTD "reader" (or "SGML compiler") will be able to process the document and display or print it as intended. This means that a single standard SGML compiler can serve many different kinds of documents that use a range of different markup codes and related meanings. The compiler looks at the DTD and then prints or displays the document accordingly.[11]

HTTP : HTTP (Hypertext Transfer Protocol) is the set of rules for transferring files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. As soon as a Web user opens their Web browser, the user is indirectly making use of HTTP. HTTP is an application protocol that runs on top of the TCP/IP suite of protocols (the foundation protocols for the Internet). [11]

ISO 8879 : ISO (International Organization for Standardization). *ISO 8879:1986(E). Information processing -- Text and Office Systems -- Standard*

Generalized Markup Language (SGML). First edition -- 1986-10-15. [Geneva]: International Organization for Standardization, 1986. [1]

JSP : JavaServer Pages (JSP) technology enables Web developers and designers to rapidly develop and easily maintain, information-rich, dynamic Web pages that leverage existing business systems. As part of the Java technology family, JSP technology enables rapid development of Web-based applications that are platform independent. JSP technology separates the user interface from content generation, enabling designers to change the overall page layout without altering the underlying dynamic content [13]

PHP : PHP (recursive acronym for "PHP: Hypertext Preprocessor") is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. [12]

RDF : The Resource Description Framework (RDF) is a language for representing information about web resources, such as content description, title, author, copyright information, availability schedules, and more [5]

RSS : RSS is a format for syndicating news and the content of news-like sites, including major news sites like Wired, news-oriented community sites like Slashdot, and personal weblogs. But it's not just for news. Pretty much anything that can be broken down into discrete items can be syndicated via RSS: the "recent changes" page of a wiki, a changelog of CVS checkins, even the revision history of a book. Once information about each item is in RSS format, an RSS-aware program can check the feed for changes and react to the changes in an appropriate way.

XML : Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.[1]

REFERENCES

[1]. World Wide Web Consortium http://www.w3.org, 2005

[2]. A Technical Introduction to XML by Norman Walsh http://www.xml.com, 2005

[3]. What is RDF? by Tim Bray http://www.xml.com/pub/a/2001/01/24/rdf.html, 2005

[4]. Dublin Core Metadata Initiative http://dublincore.org/resources/faq, 2005

[5]. RDF Tutorial http://www.w3schools.com/rdf, 2005

[6]. What is RSS http://www.userland.com, 2005

[7]. RSS 2.0 Specification http://blogs.law.harvard.edu/tech/, 2005

[8]. SAP http://www.sap.com, 2005

[9]. JOBSDB http://www.jobsdb.com, 2005

[10]. http://www.pickajob.com, 2005

[11]. http://www.whatis.com, 2005

[12]. PHP http://www.php.net, 2005

[13]. SUN MICROSYSTEM http://java.sun.com, 2005

[14]. UNESCO http://www.unesco.org, 2005

[15]. SIEMENS AG, http://www.siemens.com, 2005

CURRICULUM VITAE

Name	: Utomo Budiyanto
Place, Date of Birth	: Jakarta, 15 November 1973
Gender	: Male
Email	: tomo@bl.ac.id

EDUCATIONAL BACKGROUND

2003 – 2005	: Swiss German University majoring Information Technology
1991 – 1996	: STMIK Budi Luhur Jakarta majoring Information Management

WORKING EXPERIENCE

2004 – now	: Head of Information System Development Division at Information System Bureau in Budi Luhur University
2001 – 2004	: Web Developer at Budi Luhur University
1999 – 2001	: Head of Computer Laboratory Academy of Secretary Budi Luhur
1996 – 1999	: Programmer and System Analys at Electronic Data Processing STMIK Budi Luhur

SKILLS

Programming	: BASIC, PASCAL, C/C++, Java, VB
Database	: XBASE, MySQL, ORACLE, ACCESS
Scripting	: HTML, XHTML, XML, ASP, PHP, JSP, Javascript
Operating System	: DOS, Windows, Linux
Others	: Photoshop, Flash, Dreamweaver, CorelDraw, Video Editing

PORTOFOLIO

- Budi Luhur University website (http://www.bl.ac.id)
- Budi Luhur Career (http://www.bl.ac.id/karir)
- New Student Online Registration System (http://www.bl.ac.id/registration)
- Budi Luhur Alumni website (http://alumni.bl.ac.id)
- Budi Luhur Student Center (http://www.bl.ac.id/student)