Visualization module of Vijjana, a Pragmatic Model for Collaborative, Self-organizing, Domain Centric Knowledge Networks

Sasanka Babu Gottipati

Problem Report submitted to the College of Engineering and Mineral Resources at West Virginia University in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

Approved by Dr. Yenumula. V. Reddy, Ph.D., Chair Dr. James D Mooney, Ph.D. Dr. Sumitra Reddy, Ph.D.

Lane Department of Computer Science and Electrical Engineering Morgantown, West Virginia 2008

Keywords: Vijjana, Visualization, Knowledge base, ODP, DOMZ, Prefuse, Hypergraph
ABSTRACT

Visualization module of Vijjana, a Pragmatic Model for Collaborative, Self-organizing, Domain Centric Knowledge Networks

Sasanka Babu Gottipati

The World Wide Web has become an integral part of today's life. There is an exponential growth of information stored in the World Wide Web and it has now become the preeminent source for finding information in any field with the help of some search engines like Google. However, because of the inherent limitations of keyword based searching, the user is still obliged to wade through a number of irrelevant web pages before finding the desired information. Thus, the World Wide Web demands an implementation of some kind of Semantic Web [1] enabling automated programs to process the web content and Vijjana is a solution to that. Vijjana [2] is defined as a Pragmatic Model for Collaborative, Self-organizing, Domain Centric Knowledge Networks.

In this report we explain the concept of Visualization of Vijjana with a module. We used Open Directory Project [3] data as a knowledge base which provides plenty of classified data for manipulation into categories based on its own features. Open Directory project data is a RDF (Resource Description Framework) [4] type, and the Meta data of that RDF is transformed to a MySQL relational database using an open source tool phpODPworld [5]. Different XML’s are generated connecting to the database following the schema rules of visualizations, and they were used as an input to the Prefuse [8] and Hypergraph [7] tools to get visual applets of Treeview, Radialgraph [6] & Hypergraph [7]. Applets thus generated were used by the users of Vijjana for visual search and also to add content (URL’s Documents etc.) to the knowledge base.
Acknowledgments

I gratefully acknowledge my Committee Chair, Dr. Yenumula V. Reddy and am very thankful for his supervision, constant support, and proper guidance throughout the project. He always encourages me to come up with new ideas and helped me a lot in implementing those ideas. This project is originated from his ideas and evolved from the insights and suggestions of Vijjana team.

I would like to extend my thanks to Dr. James Mooney and Dr. Sumitra Reddy for serving on the committee and supporting the project with their suggestions and encouragement.

I would like to thank all my friends and project mates who helped me with their suggestions and always being supportive in completing this project.

Finally, I am deeply thankful to my beloved parents and sister for their encouragement and support in all the things I do.
# Table of Contents

Acknowledgments ........................................................................................................................................ iii

Table of Contents ....................................................................................................................................... iv

Chapter 1 Introduction ................................................................................................................................. 1

1.1 Background .............................................................................................................................................. 1

1.2 Problem Statement .................................................................................................................................. 2

Chapter 2 Vijjana Model ............................................................................................................................... 3

2.1 Introduction .............................................................................................................................................. 3

2.2 Vijjana Agents ......................................................................................................................................... 4

   2.2.1 Taxonomy and the Semantic Net of Knowledge (T and R) ......................................................... 4

   2.2.2 The Discovery Agent (dA) ........................................................................................................... 5

   2.2.3 The Organizing Agent (oA) ......................................................................................................... 6

   2.2.4 The Consistency and Completeness Agent (cA) ......................................................................... 7

   2.2.5 The Search Agent (sA) ................................................................................................................. 8

   2.2.6 The Rating Agent (rA) ............................................................................................................... 9

   2.2.7 The Visualization Agent (vA) ..................................................................................................... 9

2.3 The Vijjana Architecture [2] ................................................................................................................ 11

   2.3.1 Vijjana Framework Architecture ............................................................................................. 11

   2.3.2 Vijjana Client Architecture ..................................................................................................... 12

Chapter 3 Vijjana Database .......................................................................................................................... 13

3.1 Introduction ............................................................................................................................................. 13

3.2 Implementation: ..................................................................................................................................... 13

   3.2.1 Resource Description Framework ............................................................................................ 15

   3.2.2 Relational Database Advantages ........................................................................................... 17

3.3 RDF to Relational Database (MySQL) ................................................................................................. 18

   3.3.1 ODP/dmoz Database installation .......................................................................................... 19

   3.3.2 Dependencies using phpODPmyworld .................................................................................... 20

Chapter 4 Visualization ............................................................................................................................... 21

4.1 Introduction ............................................................................................................................................. 21

4.2 Vijjana Visualizations ............................................................................................................................ 22

   4.2.1 Tree View Visualization ......................................................................................................... 22

   4.2.2 Radialgraph Visualization ...................................................................................................... 22

   4.2.3 Hypergraph Visualization ...................................................................................................... 22

4.3 XML in Vijjana ...................................................................................................................................... 23

   4.3.1 Why do we need XML? .......................................................................................................... 23

   4.3.2 XML Introduction ................................................................................................................... 24

   4.3.3 XML Schema Introduction .................................................................................................. 25

   4.3.4 XML Schema’s Used in Vijjana ............................................................................................. 27

      4.3.4.1 Sample XML schema of TreeView ............................................................................... 27

      4.3.4.2 Sample XML Schema of RadialGraph ........................................................................... 28

iv
4.3.4.3 Sample XML Schema of HyperGraph: ................................................................. 28

4.4 Introduction to Prefuse & Hypergraph Toolkits .................................................. 30
  4.4.1 Prefuse Toolkit ........................................................................................................ 30
    4.4.1.1 Introduction ........................................................................................................ 30
    4.4.1.2 Prefuse Architecture ......................................................................................... 31
    4.4.1.3 Source Data ......................................................................................................... 31
    4.4.1.4 Data Tables .......................................................................................................... 32
    4.4.1.5 Visual Abstraction ............................................................................................. 32
    4.4.1.6 User Interface ...................................................................................................... 32
    4.4.1.7 User Controls ..................................................................................................... 32
  4.4.2 Hypergraph Tool ................................................................................................. 34

4.5 Pros & Cons of selected Visual Views of Vijjana ................................................. 34

Chapter 5 Implementation of Visualization of Vijjana ............................................. 36
  5.1 Prefuse Toolkit & Hypergraph ................................................................................ 36
  5.2 PHP Scripts & MySQL Database ........................................................................... 36
  5.3 NetBeans IDE .......................................................................................................... 36
  5.4 Eclipse ..................................................................................................................... 37
  5.5 Platform .................................................................................................................. 37

Chapter 6 Results ...................................................................................................... 38
  6.1 Results of Vijjana Visualization ............................................................................. 38
    6.1.1 TreeView ............................................................................................................... 38
    6.1.2 RadialGraph View .............................................................................................. 40
    6.1.3 HyperGraph View .............................................................................................. 42

Chapter 7 Future Work and Conclusion ................................................................. 44
  7.1 Future Work .............................................................................................................. 44
  7.2 Conclusion ............................................................................................................... 45

References .................................................................................................................. 46
List of Figures

Figure 1: Sample Semantic net of Vijjana Visualization Views. .................................................4
Figure 2: A proposed View of Vijjana Knowledge Network Web Interface adding a JAN (Sri
Ram Devalapalli work - a part of Vijjana Project) .................................................................6
Figure 3: A proposed View of the User’s Vijjana Knowledge Network after a new JAN was
added (Suman Matcha’s work - a part of Vijjana Project) ..........................................................7
Figure 4: A proposed view of Search agent of Vijjana ...............................................................8
Figure 5: Rating agent proposed page .........................................................................................9
Figure 6: Radialgraph and Hypergraph Views of Computer Science domain Vijjana .......... 10
Figure 7: The Vijjana Architecture .............................................................................................11
Figure 8: Vijjana Client Architecture .........................................................................................12
Figure 9: A generalized triple (subject, predicate and object) of RDF. ..................................16
Figure 10: Database Structure of ODP using phpODPWorld tool ...........................................18
Figure 11: Sample Treeview .....................................................................................................27
Figure 12: Sample RadialGraph View .......................................................................................28
Figure 13: Sample HyperGraph view .......................................................................................29
Figure 14: The relation between classes to the Infovis reference model and different Prefuse
packages is shown [6]. .............................................................................................................31
Figure 15: List of actions to be done during the visualization process from input abstract data
to the transformation of user controllable visual (view). .........................................................33
Figure 16: Treeview of Computer Science domain showing the categories related to that ....... 38
Figure 17: Treeview when the Database Theory category is extended .....................................39
Figure 18: Radialgraph of Computer science domain ..............................................................40
Figure 19: Radialgraph of Computer science domain limited to 100 nodes ......................... 41
Figure 20: Hypergraph showing Computer science domain and the URL which directs to the
index of URL’s of Computer Science domain ....................................................................42
Figure 21: Hypergraph showing the navigation towards Conferences from Computer Science. .43
Chapter 1 Introduction

1.1 Background

If tourists are lost in a city, they will surely dislike it. It is the same in the case of the World Wide Web; being lost or uncertain is quite common while we are searching to find some information. There are several search engines like Google, Yahoo, etc, that will search the information on the World Wide Web based on the keywords or the inputs give to the search engine. In this search process we will come through a lot of unnecessary information or web pages and also we not sure what are the related things to those keywords are.

“Navigation should let us know where they are, how they got there and where they can go next” [Wroblewski, 2002: 48]. If we consider directions to cities or towns, to find the easiest way Maps and Signs are used – similarly the navigation system of World Wide Web could be optimized too. The layout or the structure of the cities and towns is almost static or there are very few changes but the structure of the World Wide Web is huge and there is no common pattern or right way to deal it. We need to implement a Semantic Web [1] enabling automated programs to process web contents to optimize the information stored in the World Wide Web as that information is growing exponentially.

Vijjana [2], a solution for the semantic web, in brief is defined as a Pragmatic Model for Collaborative, Self-organizing, Domain Centric Knowledge Networks. Vijjana is a framework for the development of Semantic web by storing the Meta data of Web Pages in a centralized database which acts as a Knowledge Base. This Meta data of each web page is defined as a JAN in Vijjana, so a knowledge base is a collection of JAN’s. On the top of the Knowledge Base the
Vijjana provides an interface for the users to access those Web Pages and related information (Web Pages) by visual means. The users can also contribute their own WebPages or some relative content to Vijjana. The Vijjana can be a Domain centric for particular organization or it can be a universal for the entire World Wide Web i.e., a centralized structure obtained by the integration of Vijjana Domain’s in all the fields.

Vijjana knowledge base needs a huge data set of webpage’s having some hierarchical categorization, so that it is easy to find the category to which the webpage belongs to and also the semantics related to that webpage. Open Directory Project (ODP) [3], most comprehensive human-edited directory of the Web, is having the similar properties as a Vijjana knowledge base and it is used as Vijjana data for the demonstration of Visualization module. The Open Directory Project data is available for download and it is in RDF [4] (Resource Description Framework is a framework for Meta data) format. The RDF of Open Directory Project is transformed to MySQL relational database and using this database as Knowledge base we can demonstrate the visualization module of Vijjana.

1.2 Problem Statement

To demonstrate the Visualization module of Vijjana [2], XML’s (based on the schema rules of visuals) are generated for a particular domain (we choose Computer Science as our domain of interest) from the knowledge base having standard taxonomy. Before that a database that acts as a knowledge base of Vijjana is created by the transformation RDF of Open Directory Project data (having a predefined taxonomy and collection of URL’s to MySQL database. The generated XML’s are given as an input to Prefuse and Hypergraph Visualization toolkits to generate visual applets of Treeview, Radialgraph [8] and Hypergraph [7].
Chapter 2 Vijjana Model

2.1 Introduction

The Vijjana model [2] is an integration of agents so as to get an Intelligent Domain Centric Knowledge Database.

We define Vijjana as

\[ \text{Vijjana-X} = \{ J, T, R, dA, oA, cA, vA, sA, rA \} \]

where

\( X \) = the domain name

\( J \) = the collection of JAN’s in the Vijjana-X

\( T \) = the Taxonomy used for classification of JAN’s

\( R \) = the domain specific relations

\( dA \) = the discovery agent which find relevant JAN’s

\( oA \) = the organizing agent which interlinks the JAN’s based on R

\( cA \) = the consistency/completeness agent

\( vA \) = the visualization agent

\( sA \) = the search agent

\( rA \) = the rating agent

Each agent has its own importance and in the remainder of this section we explain each of these components of the Vijjana model.
2.2 Vijjana Agents

2.2.1 Taxonomy and the Semantic Net of Knowledge (T and R)

The basic step in developing a Knowledge Base is the classifications of its constituent JAN’S and interlinking or finding relation between them to form a Semantic net. For this, we first define taxonomy that suits the domain for which we are going to develop Vijjana Knowledge Base. Since the same knowledge could be organized using a wide variety of taxonomies, the Vijjana model is designed to work with any taxonomy and associated relational semantics, by treating them as parameters that could be modified as the knowledge base evolves. This makes Vijjana completely portable and we can import and export the knowledge base if the Knowledge base of Vijjana’s has the same taxonomy and relational semantics.

![Sample Semantic net of Vijjana Visualization Views.](image-url)

*Figure 1: Sample Semantic net of Vijjana Visualization Views.*
The first and foremost thing we need to agree upon is on the Taxonomy that we use to build the Knowledge Base, if we employ Vijjana to build large-scale knowledge networks involving multiple enterprises, and then we need to consider relational semantics. The following Figure 1 shows an example of the taxonomy in two visual views and relational semantics used in building the Vijjana-CS used throughout this paper. As new JAN’s are added, this knowledge network grows; it shrinks as dead links are removed.

2.2.2 The Discovery Agent (dA)

The Vijjana model is developed by defining all the services using an agent paradigm. The Discovery Agent service is of two types, the first one is by human initiative and the second one is an automated program that collects JAN’s related to that file which is invoked by a program or a script. In the first version of Vijjana model, there is only service type which is human initiative type. The users who registered under that Vijjana knowledge base, if they found an interesting JAN, they will make up that piece of information or webpage and then using the taxonomy visual view, they will throw that JAN to the corresponding category (field of Interest). Here “markup” involves indicating the actual node type the discovered JAN belongs to by “clicking” on the appropriate node-type on a graphically displayed taxonomy.

In addition to that, in future we can use subscription of RSS feeds from important websites related to that category and incorporate automatic addition of JAN’s to the knowledge base. The parsing and addition of JAN’s to the knowledge base is automated Since RSS feeds are based on a standard protocol. We can also use Email so that the collaborators can contribute relevant JAN’s by following a predefined syntax of the message to enable automatic parsing of Email.
2.2.3 The Organizing Agent (oA)

Once a JAN is obtained through one of the four possible ways: (1) a RSS feed, (2) Email, (3) by “clicking” on the “markup” button installed by a user on his or her browser, or (4) through the Vijjana client interface, it is handed-off to the organizing agent which first ensures that the JAN represents a genuine link - one that is not broken or submitted by an unreliable source. It then examines the markup information, which is used for classification and interlinking. This information is also used for creating inverse links. For example if we discover a JAN of the type “syllabus” which is “based-on” a “textbook”, the oA will link this JAN with the indicated “textbook” via the “based-on” relational link. In addition, the oA will also link the “textbook” with the “syllabus” using the inverse link “has-syllabus” which represents the fact that this “textbook” has the specified “syllabus”.

Figure 2: A proposed View of Vijjana Knowledge Network Web Interface adding a JAN (Sri Ram Devalapalli work - a part of Vijjana Project)
Thus as new JAN’s of the type “syllabus” enter the Vijjana knowledge network; they will be automatically linked using both the link and inverse link. In addition to organizing the link into the evolving Vijjana, it logs the information into a provenance file which will be later used for generating the history of any JAN. The oA could also notify users who signed-up for automatic notifications about the incorporation of a new JAN into the Vijjana. The Organizing agent also stores the number of times a particular JAN was visited by the user, which could be used by the rating agent (rA) in computing the rating heuristic value.

![A White paper added as a JAN](image)

Figure 3: A proposed View of the User’s Vijjana Knowledge Network after a new JAN was added (Suman Matcha’s work - a part of Vijjana Project)

### 2.2.4 The Consistency and Completeness Agent (cA)

The Consistency and Completeness Agent (cA) is responsible for maintaining the integrity of the Vijjana knowledge network. This is accomplished by the cA by periodically visiting all the URLs of the JANs to make sure they are still “alive”. The cA will also find out all
the relational links that are not set (i.e., incomplete) which will be posted to an agenda file so that the users can manually fill them as information becomes available. These nodes with incomplete relational information will also be color coded so that the visualization agent can display them thus attracting the attention of the users who may take steps to correct the problem.

2.2.5 The Search Agent (sA)

The search agent implements a variety of search mechanisms including keyword-based search and more advanced search mechanisms such as using relational algebra. For example, we can issue a command such as “find all textbooks that have associated syllabi and include lecture notes” which will return all the JAN’s that satisfy the stated criteria. This however is only possible if the semantic links are completely specified. We include this feature for the sake of completeness and to cope with large knowledge networks where the user is already familiar with the network topology but needs the results quickly without browsing.

![Figure 4: A proposed view of Search agent of Vijjana.](image-url)
2.2.6 The Rating Agent (rA)

The Rating agent (rA) is another important feature of Vijjana, which is intended to bring the attention of the user to high value JAN’s. High value JAN’s may be determined through a rating mechanism which uses a combination of user specified values, number of “hits”, and the number of fully specified relational links. By incorporating a coloring scheme in the display of the knowledge network, the user’s attention is quickly drawn to valuable JAN’s. For example, if a JAN representing a “textbook” has a very large number of “has-syllabus” links, it should be intuitively obvious that the textbook represented by that node is a high value item. We plan to experiment with a variety of heuristics and incorporate user feedback into this agent.

![Figure 5: Rating agent proposed page.](image)

2.2.7 The Visualization Agent (vA)

The visualization agent is responsible for displaying the Vijjana knowledge network in a variety of forms based on user preferences. One of the most useful ways to display a knowledge network is through a variety of visual representations, such as graphs, tables, and maps. The visualization agent allows users to customize these displays according to their preferences, providing a more intuitive and user-friendly experience. By incorporating advanced visualization techniques, we hope to enhance the user's ability to explore and understand the knowledge network.
network is by drawing a hyper tree shown in Figure 1 (pioneered by Xerox Corporation through its InXight.com website), or as a radial graph where nodes at each level at the end of radii are equally spaced. The user can browse the lower level nodes by “clicking” on the nodes as the information at lower levels unfolds.

In this view, the user gets a bird’s eye view of the whole knowledge network from which the user can navigate to the area of interest as it unfolds with more and more detail as you near the target. This is perhaps one of the biggest advantages of a Vijjana compared to the social networking bookmark sharing websites. In addition to the hyper tree model, we intend to provide a variety of other representations including hierarchies and network traversal following predefined patterns.

Figure 6: Radialgraph and Hypergraph Views of Computer Science domain Vijjana.
2.3 The Vijjana Architecture [2]

2.3.1 Vijjana Framework Architecture

On the left hand side of Figure 7, administrator interface is shown, which is used for checking consistency and organizing JAN’s into the knowledge network. On the right hand side is shown the general user interface where users can contribute JAN’s, import or export entire knowledge networks or visualizes the network itself using the vA. In addition, it shows an agent intended to manage social networking.

![Figure 7: The Vijjana Architecture](image)

For example, a user browsing the knowledge network may become aware of another user who is also interacting with the knowledge network and then may decide to open a communication channel which may result in arriving at rating for a node collaboratively. In the
bottom we show the interface between the discovery agent and the web through which the JAN’s migrate from the web to the Vijjana knowledge network. On the top is shown the database that contains the taxonomies, link relationships and the actual URLs (Uniform Resource Locators) representing the JAN’s in individual knowledge networks.

2.3.2 Vijjana Client Architecture

The Vijjana system may be used using a dedicated client interface shown in Figure 8 or using a web interface shown in Figure 3. Figure 8 illustrates the overall architecture of the Vijjana client program. As can be seen in Figure 8 a JAN may be sent to Vijjana by Email or via a markup by the user. When the JAN arrives via Email the message is parsed to discover the markup information for organizing the JAN into its proper place in the evolving knowledge network using the organizing agent. In later implementations, we plan to expand this interface to include RSS feeds.

![Figure 8: Vijjana Client Architecture.](image-url)
Chapter 3 Vijjana Database

3.1 Introduction

The Vijjana project is aimed to use logical network view to illustrate the complex relations among large growing knowledge which publishes 100 times faster than 20 years ago. Even though the knowledge creator, owner and user can be different objects such as people, website, or companies, the intermediate is just in one form, broadcasting by several fast developing technologies.

The networking or Semantic Linking is the most obvious and prestigious one. More and more net users are going to become thirsty for the distilled knowledge which has been ordered by category and searchable by its features. All these user requirements contribute a lot for the future development of Internet.

The Vijjana is aimed to provide an expert system of organizing large amounts of knowledge according to the several of user's requirements. This rudiment requirement calls for a scientific way to deal with the raw data. Let’s see the implementation part of Vijjana Database.

3.2 Implementation:

The Vijjana Database consists of three major parts, basic raw data, user data and website data. The raw data is gathered from the biggest open source data store, Open Directory Project. It provides plenty of data for manipulating which is classified into categories based on its own features. This project data acts as a directory standard of World Wide Web has been widely used
in many areas, providing data for many big search engines, such as Google, the biggest worldwide Linux-based search engine, and yahoo also.

For the user purpose, the Vijjana system can be regarded as large scale distributed system. To fulfill this requirement, the well structured user database is necessary for the project smooth running. This user database is used to maintain the basic user information and his recent activities based on changeable interest in future. Taking the track of Users interest history is necessary to find their appropriate cut off point; therefore, a large and scalable database would be important part in the Vijjana project.

Compared with the above two, the Vijjana website database is easier but also takes a big important point. To provide an efficient and effective service, a well and easy maintainable website database is crucial. So the Vijjana database is set to have these three major parts, but would have more supplement features added. The following is what we did to gather the raw data used to organize the knowledge.

Open Directory Project (ODP) is a famous open source project that provides more than 3 million records of knowledge data which is widely used as a directory standard of World Wide Web by several of search engines, including the most famous Linux-based search engine, Google. The ODP calls their data as "RDF dumps" and they use RDF format. The basic introduction to the RDF is as follows:
3.2.1 Resource Description Framework

"RDF" is an abbreviation for Resource Description Framework. It is an XML-based standard (or recommendation) to manage metadata.

According to W3C RDF Primer [9] "It is particularly intended for representing metadata about web resources, such as the title, author, and modification date of a Web page, copyright and licensing information about a web document, or the availability schedule for some shared resource. RDF is intended for situations in which this information needs to be processed by applications, rather than being only displayed to people."

RDF standard was published by W3C as early as 1999. But in 2004 it was designed and presented as a whole new standard. Today we are using the same standard, and also it’s been used in several new areas, including RSS [10] and FOAF [11]. RDF is part of W3Cs "Semantic Web project" [1]. With RDF, you can enter the semantic content in three way format: subjects, objects and predicate. In order to describe the linguistic content of these three, let’s see the following sentence:

*Vijjana is about Semantic Web.*

In the above sentence, "Vijjana" is the subject, the thing which has a characteristic, "is about" is predicate and “Semantic Web” is the object. Predicate defines the property in the broad sense. Here the property between the subject (the person making any) and the object is referred by “is about".
Figure 9: A generalized triple (subject, predicate and object) of RDF.

With RDF, you can point out some relevant data, such as the author of an article, with whom the article deals with, the records in a table, the goods in a shopping cart. One can also talk about the subject in many ways, how something is being prescribed or when something happens (= objects). The difficulty with RDF is that it describes a unique thing, since we like to mix unique descriptions since a generalization RDF is not useful. With the "unique description" what it means is that one thing points out one item, while a generalization means one thing points out to a group or category.

Let’s see one more example “Dr. Reddy is the Chair of Vijjana project and the URL of the project is found at www.vijjana.csee.wvu.edu” as a statement. In this case the URL finding the resource we want to talk about is www.vijjana.csee.wvu.edu and this URL is the subject of our statement. Its predicate is “Chair” and the object is “Dr. Reddy”. The URL www.csee.wvu.edu/yreddy.html that gives us the information of Dr. Reddy is the object resource instead.

From the above, we can simple say that the basic building blocks of an RDF Statement are Resources and literals. A literal is simply a character string and anything that can be reached by a URL is a resource. An object can be either resource or literal but the Subjects and predicates always need to be resources. Before going to the RDF to Database transformation let’s see the advantages of RDF as Relational database over standard XML representation.
3.2.2 Relational Database Advantages [12]

There are many advantages if we transform RDF as a Relational Database and I will list the advantages in the coming paragraph. Most of the standard relational database (Oracle, MySQL, SQL Server etc) provides similar features and hence this is a generalization to all the databases.

- Easy Management of statements: It is easier to retrieve a subset of statements from a huge number of statements using SQL queries. The realization of one entity, the concept of grouping several statements together, is very easy in a database where as XML syntax causes problems for doing the same.

- Transparency: The complexity of the SQL queries is reduced by the Views. By hiding the implementation details the Stored Procedures provides a user interface that encapsulates more complex tasks involving several SQL queries.

- Safety and Recovery: The state of database is always consistent. Consider a case that the model fails at some point while executing a stored procedure for deleting a complete RDF. In this scenario if we use database then we can undo the whole transaction unless it has be committed after successful execution. Also the parsing process results are stored in the database and so there is no need of reverse parsing.

- Performance: Parsing a long RDF data to get a resulted is tedious job than using a single SQL query to retrieve the same results. The database’s query optimizer and caching mechanisms is very fast compared to parsing an XML, especially when we execute a lot of similar queries are in a sequence.
Compact Storage: There is a good chance of occurrence of same character strings within a piece of RDF metadata and we can avoid those duplicated if you use relational database.

Thus we prefer transformation of RDF Dump to a relational database and let’s see the procedure to implement it.

3.3 RDF to Relational Database (MySQL)

It is clear that RDF file uses tags to distinguish different resource categories, to maintain a high-changeable RDF file is not easy for its growing cumbersome size. The best way to maintain such big amount of data is to store them into database and maintain daily increment on it. The editor of Open Directory Project created a set of tool named “ODP/dmoz" to do such job. A famous tool for this is called "PhpODPWorld" [5]. It design a database containing three basic table "structure", "resources” and "xurls". The basic structure is as following:

![Database Structure of ODP using phpODPWorld tool.](image)

The script used to import the RDF file into database is developed by Steve who is one of open directory project editor. He is also responsible for fetch the bugs and errors existing in the
RDF file. Based on the tools he designed, we can easily manipulate data by searching by their categories and their own features. We also apply different searching algorithm to generate keywords on them.

The following procedure explains in steps to do the RDF to Database transformation. The installation procedure and the dependencies using the tool are given below [5].

3.3.1 ODP/dmoz Database installation

1. To the wanted directory on the web server, extract and move all the unpacked files of downloaded phpODPWorld package.

2. Complete the following steps
   - Create "logs" directory, if you enabled logging in the config file and this directory must be writable by the web server
   - Create "smarty/cache" directory and "smarty/compiled", if you enabled smarty in the config file and this directory must be writable by the web server

3. Step three
   - Create a database and a user either with password or without.
   - The database table defined in "tools/db.sql" is created.

4. Wanted categories and references, database settings are reflected by editing "config.inc.php" and "tools/config.pl".

5. Downloaded either the complete content or structure RDF.

6. Use the Perl script "tools/extract.pl" to extract your categories from the RDFs. (Perl module DMOZ-ParseRDF-0.14 should be installed now)
7. Using Perl scripts insert your categories (from the RDFs) into the database.

8. Do the following for initial run to update the count (of sites) for each category
   - Turn on maintenance mode in "config.inc.php".
   - Turn off maintenance mode in "config.inc.php".

9. The above steps should make you build your site.

Some other tasks

- Repeat step 5 to 8 when you want to install a new RDF.

3.3.2 Dependencies using phpODPworld

1. A database - MySQL, Postgresql, or any other supported by Pearl DB.

2. A PHP (with PHP 4.3 or better) enabled web server having (It may work with older versions of PHP.):
   - gettext (without which only English language is supported) support to have working translations.
   - Install Pear DB and preferably PHP Smarty.

3. Install DMOZ-ParseRDF (and DBI) with Perl.

   A Database with standard taxonomy and URL’s corresponding to each category is created. From Figure 10, the table STRUCTURE table contains the taxonomy of all the domains and we can choose a domain of our interest (Computer Science). XURLS table contains the resources (URL’s) related to the category. Our problem of interest is using STRUCTURE table so that we can visualize the taxonomy of computer science domain.
Chapter 4 Visualization

4.1 Introduction

Wiki’s Definition, Visualization is any technique for creating images, diagrams, or animations to communicate a message. Visualization through visual imagery has been an effective way to communicate both abstract and concrete ideas since the dawn of man.

There are two types of Visualization

- Scientific visualization (model-based, sampled)
- Information visualization (no model)

Our point of interest is on Information Visualization and it is defined by Card et al. (1999) as “The use of computer-supported, interactive, visual representation of abstract data to amplify cognition”

Visualization of Vijjana is the most important part of all the agents that we have in Vijjana model. It is the users interface to interact with the knowledge base to find the semantics and also to obtain relevant information in a particular field by means of user friendly navigation. In the first stage Visualization of Vijjana is done using Prefuse toolkit [6] and Hypergraph [7]. We used Prefuse toolkit to get two visual views and Hypergraph for hyperbolic view. Prefuse and Hypergraph, both are open sources and they provide several ways to visualize data in which we use three visual views for our module, TreeView, RadialGraph & HyperGraph. Let’s see the basics of Treeview, Radialgraph [8] & Hypergraph.
4.2 Vijjana Visualizations

4.2.1 Tree View Visualization

Tree view Visualization is analogous to a hierarchical family tree layout. Visualization can be done by highlighting the tree dataset which is shown in Data Manager Window, followed by clicking visualization tree window; a similar window will appear displaying tree view visualization. Zooming in and out of the tree can be done by holding the right or middle mouse button.

A search button can be found at the right corner of the Tree Map. The search will highlight the query result.

4.2.2 Radialgraph Visualization

The Radial graph layout is implemented by focus + context technique for visualizing and manipulating large hierarchies.

The focused node will be displayed at the center and all other nodes are rendered around the focused node. The distance from the center node will determine the size at which the node is rendered. This will help us in displaying very large hierarchies.

4.2.3 Hypergraph Visualization

The techniques employed in Hypergraph Visualization:

Radical Layout: The trees are displayed on a Euclidean plane with the root at the top and children below parents. Roots are connected to their parents with edges in hyperbolic layout. Roots are at the center while children at outer ring to their parents. The circumference increases with radius.
Distortion Technique: Nonlinear technique is used to accommodate focus for large number of nodes.

Non Overlapping: Open angle is given to ensure nodes do not overlap with each other. All children of one node are laid out in open angle.

Refocusing: transformations are implemented to allow fluent node repositioning. In order to move a node to the center, click on that node, grab and reposition a single node.

Now for the Visualization of Vijjana we have to first connect to the database of Vijjana and generate XML’s, which are given as an input to the Prefuse and Hypergraph tools and thereby we will get corresponding User Controllable visuals. Before dealing with generation of XML, I will introduce XML, XML schemas and the XML schema’s used for different visual views.

4.3 XML in Vijjana

4.3.1 Why do we need XML?

The raw data from the database is to be given in an abstract form to the Prefuse & Hypergraph toolkits, so an XML having the structured information about the Vijjana semantic net or taxonomy is required. There are several XML schemas that are related to the corresponding visual views. For example for Hypergraph we use an XML that has the schema of GraphXML. So based on the type of Visualization that we generate, first we have to study the XML schema and then we have to transform the raw data to that form of XML.

The XML file is easy to transfer because of its small size but, hard to interpret because of its flexibility. Actually, what we are going to implement is to create the XML data schema based
on the database table we have, and create an automatically intelligence system to generate the XML file based on the search result. The result is large XML file which can be interpret based on several of data schema, and it will return different small xml files based on them. This XML file can be interpreted as the raw data file for the visualization and other purposes.

4.3.2 XML Introduction

A markup language containing structured information for documents is termed as XML. Structure information is the information that contains content (words, pictures, etc.) and the role it plays (to say, “a word document contains header on the top and footer on the bottom of the page” is a structure information). We can say almost all the Documents are structured. To identify the structure of the documents we use markup language and a standard way to add markup to the documents is defined by XML Specification.

The technology is not new but it was developed from Standard Generalized Markup Language (SGML), which is an ISO standard. SGML can be seen as the father of both XML and HTML. XML is a subset of SGML and HTML is an application created from SGML. All three standards are intended to be used over different platforms and environments.

SGML has been there even before the World Wide Web came in existence. The Web has largely reached its broad audience because it is cheap and easy to use and all the leading browsers that we are using now are free. SGML is not the shorter version of HTML but SGML is simultaneously a very broad and general standard. HTML has a number of shortcomings that make it inappropriate to describe and manage information in an efficient manner. Among other things, HTML language has a fixed set of tags that does not describe the information that is
created in a good way. Moreover HTML is more focused on presentation than the logical structure of the information.

In XML, there is a regulatory framework of how information should be organized in the form of a Document Type Definition (DTD). The following is an example of XML which stores the address information of an employee.

```xml
<employee>
    <name> Rama </name>
    <address>
        <street> Banjara Hills, Door No 4-12-6 </street>
        <city> Hyderabad </city>
    </address>
</employee>
```

In this way we can put the information in a context and make it more understandable. XML is a readable format for both man and machine. A machine requires a well-defined structure to be able to manage the information we create and that is precisely what gives XML.

### 4.3.3 XML Schema Introduction

When creating structured information in XML we have to use Document Type Definition (DTD), rules to be defined and applied to organize XML. The rules are nothing but the order in which various elements may enter the document, or whether an element must occur or not. These rules work as an aid in writing as well as to check the correct document.

DTD ensures that the data is simple and the document is correct. DTD is a legacy of SGML, which became an ISO standard in 1986, and is very useful in the applications where
SGML is used traditionally. With the advent of XML, a world of new opportunities for exchange of information is opened. This leads to frame even more stringent rules for the XML documents to get a better control and in order to fill those needs XML Schema is developed.

A schema is itself an XML document. This makes various tools which already interpret XML documents can also interpret Schemas. Now in addition to the properties contained in a DTD we have to frame a number of new rules. These rules are divided into two parts, for the structure and data types.

Now coming to the other part, the most important property is to define data types. These data type rules help processing of data in an easier way for a host application. To get a clear picture on this, a price must be a numerical value instead of a text string, since it may be used to make calculations. In addition to that, you can make a more accurate monitoring of data, which is also important for a host application. There may be some values which have a predetermined form, such as ZIP code, and with the XML schema, we can make a rule that the zip code will consist of figures and is five characters long.

All the above rules or properties were not possible with a DTD. A DTD controls whether we have a right structure or not whereas a Schema allows addition of controls so that contents of each element is in a meaning full way.

XML Schema is particularly important for the exchange of information. In Business Software, many transactions over the network will automatically be placed in different
organizations and also the data can be business-critical which leads to the importance of XML Schema.

4.3.4 XML Schema’s Used in Vijjana

Now for the visualization of Vijjana, in the first stage, we have to define the logic in retrieving data to follow the rules of XML schema’s of corresponding visualizations. Let’s see the three samples of XML Schema that we used for visualization.

4.3.4.1 Sample XML schema of TreeView

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<tree>
  <declarations>
    <attributeDecl name="name" type="String"/>
  </declarations>
  <top>
    <branch>
      <attribute name="name" value="Computer Science"/>
      <branch>
        <attribute name="name" value="Software"/>
      </branch>
      <branch>
        <attribute name="name" value="Hardware"/>
      </branch>
    </branch>
  </top>
</tree>
```

![Sample Treeview](image)

*Figure 11: Sample Treeview*
4.3.4.2 Sample XML Schema of RadialGraph

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<GraphXML xmlns="http://graphml.graphdrawing.org/xmlns">
  <graph edgedefault="undirected">
    <key id="name" for="node" attr.name="name" attr.type="string"/>
    <node id="1">
      <data key="name">Computer_Science</data>
    </node>
    <node id="2">
      <data key="name">Software</data>
    </node>
    <node id="3">
      <data key="name">Hardware</data>
    </node>
    <edge source="1" target="2"/>
    <edge source="1" target="3"/>
  </graph>
</GraphXML>
```

![Sample RadialGraph View](image)

Figure 12: Sample RadialGraph View

4.3.4.3 Sample XML Schema of HyperGraph:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE GraphXML SYSTEM "GraphXML.dtd">
<!-- Comment
-->
<GraphXML>
  <graph id="RUB">
    <style>
      <line tag="node" class="x" colour="blue"/>
    </style>
    <node name="1" class="x">
```

28
Let’s consider the XML Schema for TreeView and you can see the tags of that XML.

The first line gives us the information about the XML version and encoding style of that XML. The second line with tag <tree> indicates the root element of the document, the tree content for visualization is found within this tag. Now all the elements inside the root tag are all the child elements to that root element. The first child element with tag <declarations> is for the
declaration of the attributes that we were going to use throughout the XML; here in this case it’s a string type with variable name “name”. The next element indicates that the branching of the tree starts from that tag. The actual branching of the tree starts with element name <branch> and the attribute name used in that element is “name”, and we need some string value defined to that attribute. You can easily understand the branching that under Computer Science branch we have two child’s Software and Hardware. And finally we can see that the XML end’s with end of the root element.

For both Radialgraph and Hypergraph, the schema used is GraphXML. GraphXML is a graph description language in XML that can be used as an interchange format for graph drawing and visualization packages [16]. In this XML each node is given an ID and if there exists any child nodes to that node then they fall under target to that source ID (source is linked to the target). Each node has some properties like node name, hyperlink to that node and to which class the node fall under etc. All properties were defined in the DTD of the GraphXML. In this report we don’t do any automated classification to the nodes of the group in which they come under. The generated XML’s that follow the schema rules of Visuals are given as an input to the Prefuse and Hypergraph toolkits. Let’s see about those toolkits.

4.4 Introduction to Prefuse & Hypergraph Toolkits

4.4.1 Prefuse Toolkit

4.4.1.1 Introduction

The Prefuse visualization toolkit [6] is a comprehensive package for the visualization of data. It is so powerful tool to visualize that the data can include everything from a simple count...
of nodes to some millions of nodes. Prefuse is an Open Source Java Interface for constructing interactive information for visualization applications. Visualization, Animation, and Interaction were supported by Prefuse. By stringing fine-grained, reusable components together, applications are developed.

4.4.1.2 Prefuse Architecture

The broad way in which data is visualized is shown below in Figure 14. The various components are explained in detail. A part that does not belong to the visualization model, but an important part of Prefuse, is the Control component. This feature allows users to interact with the visualization.

![Diagram showing the relation between classes to the Infovis reference model and different Prefuse packages](image)

*Figure 14: The relation between classes to the Infovis reference model and different Prefuse packages is shown [6].*

4.4.1.3 Source Data

Eventually, Source Data is the one has to be visualized. This source data is in various formats, it may be formatted like GraphML or TreeML etc or it could be a table of figures, or any other data set.
4.4.1.4 Data Tables

By carrying out Data Transformations the Prefuse converts Source Data to a readable format (Data Tables). Prefuse supports three basic types of data representation: Graphs, Trees and Tables. In the three basic types of the data, each has its own significance. If these basic types are not sufficient for the representation of the data, it is possible for these data types to be used like that or with some extension transformation.

4.4.1.5 Visual Abstraction

After defining the data in the Data Tables now it’s time to give a face to the data using Visual Mapping. Here is the strength of Prefuse. This step is to give visual characteristics attributed to different types of the items in the Data Tables. It is about to represent the colors, shapes and layout of the elements in the final visualization part of the data from the Data Tables.

For example in case of a Table with different columns, all the cells from a particular column are visualized as a red quadrangle. To exemplify with a Table having a column of numbers, it is possible for certain colors or positions to be visualized as different groups.

4.4.1.6 User Interface

The visual representation of the data is ultimately reflected in a View and the user should have a good interface to interact with it. The visual representation should be in such a way that the user interface should not transform the data when we rotate, scale or shift the image.

4.4.1.7 User Controls

A separate part of Prefuse, but one of the important parts, is the Control component. With Prefuse it is possible for the user to control the visualization of data. The controls can be at
various levels in the Prefuse reference model (Figure 14). Simple control takes place in the View component. A complex form of control occurs at the Visual Abstraction level.

It is possible to change the visual characteristics of certain elements too. For example, by dragging the mouse we can either move a single node in RadialGraph or can change the color by selecting a node (which shows that the node was selected) or there may be another type of control to sort data into groups by dragging elements to a magic spot on the screen.

Now the Prefuse package guide, Figure 14 is transformed to a simplified architectural version of Prefuse Toolkit shown in Figure 15.

![Diagram](image)

*Figure 15: List of actions to be done during the visualization process from input abstract data to the transformation of user controllable visual (view).*

At the present stage we manually input XML data to the Prefuse toolkit to get the visual applets of Radialgraph and Treeview. Let’s see the introduction to Hypergraph tool:
4.4.2 Hypergraph Tool

More information than standard tree representations can be visualized by a hyperbolic tool [7]. The java code in Hypergraph provides the opportunity to work with hyperbolic trees. Hyperbolic trees were developed from non-Euclidean plane Poincare's model.

Hyperbolic graph layout uses Content + Focus technique to manage huge tree hierarchies on a relatively small screen size. John Lamping, Ramana Rao and Peter Pirolli rediscovered hyperbolic spaces in 1995 for information visualization, Experience non-Euclidean Geometry with Paul Garrett's applet [17]. A 3D hyperbolic viewer was developed by Tamara Munzner at Stanford University in her Ph.D. thesis [18].

4.5 Pros & Cons of selected Visual Views of Vijjana

Hyperbolic trees can be lucidly used to present data in hierarchical structures like organization and classification hierarchies, file directories websites etc. Methods like Zooming are used to locate parts of information at particular locations whereas hyperbolic trees show context and information at once. Many hierarchies at once cannot be showed entirely on a screen. Radial trees display potentially very large rooted or unrooted trees. Clients can work on parts of those trees without losing context. Clients can be confused by edge crossings at the time of animated focus change. With TreeView we can easily represent more than 6, 00,000 nodes at a time [7]. But there is a disadvantage in this method as we cannot interlink different nodes of different depths using Treeview. Using Radialgraph we can easily represent interlinking of nodes but the visual view is limited to only thousands of nodes (with more number of nodes at a particular depth the nodes will overlap with one another and the user cannot distinguish parent nodes).
The generated XML’s were given as an input to these to get the Visualizations of the Vijjana. The Radialgraph and Treeview at the present stage are used to view the taxonomy of the Knowledge base. Some research has to done to make other visuals work as Hypergraph so that each node can have a hyperlink. In the Hypergraph visual each node is having a hyperlink with will either directs to a webpage that call search applet that has the list if all the related URL’s related to that category or it will call discovery agent (dA) which will invoke an applet shown in Figure 2.
Chapter 5 Implementation of Visualization of Vijjana

In the process of transformation of RDF to visual views there are so many tools used in this module of Vijjana Model. To make Vijjana as a platform independent and keeping cost effectiveness in mind, all the tools used in Vijjana are open source and developed by Java Programming, except we used PHP scripts to transform RDF in to MySQL database.

5.1 Prefuse Toolkit & Hypergraph

Prefuse Beta release 2007.10.21 is used for the visualization of Vijjana for Treeview and Radialgraph. Hypergraph 0.6.3 release in October 2005 is used for the Hypergraph Visualization.

5.2 PHP Scripts & MySQL Database

phpODPworld tool is used to run PHP scripts, for the transformation of RDF to MySQL relational Database. We use phpMyAdmin 2.10.0.2 tool written in PHP intended to handle the administration of MySQL 5.0.22 version web server. MySQL 5.0.51b-community-nt via TCP/IP is used as a local server to generate XML files. (We import the database to local server as we are getting some problems connecting to the restricted ports because of university firewall restrictions).

5.3 NetBeans IDE

NetBeans 6.1 Integrated Development Environment is writing Java application that connects to the local database and generates XML of our needs.
5.4 Eclipse

Eclipse 3.3.1.1 is used to generate applets of Treeview and Radialgraph visuals by giving the generated XML’s as an input to the Prefuse tool.

5.5 Platform

We used Windows XP standard platform to run all the above programs. As we already said, all these programs are platform independent and can run on all the available platforms.
Chapter 6 Results

6.1 Results of Vijjana Visualization

6.1.1 TreeView

Figure 16: Treeview of Computer Science domain showing the categories related to that.
Figure 17: Treeview when the Database Theory category is extended
6.1.2 RadialGraph View

![Radial Graph of Computer Science Domain](image)

**Figure 18:** Radialgraph of Computer science domain.
Figure 19: Radialgraph of Computer science domain limited to 100 nodes
6.1.3 HyperGraph View

Figure 20: Hypergraph showing Computer science domain and the URL which directs to the index of URL’s of Computer Science domain.
Figure 21: Hypergraph showing the navigation towards Conferences from Computer Science.
Chapter 7 Future Work and Conclusion

7.1 Future Work

Vijjana model is in the early stage of development. Visualization module of Vijjana in this report is done using three user controllable visuals. But all the three visuals are having some limitations, either with the number of nodes or with semantic representation of them, and extensive research are necessary to find out the best way to visualize Vijjana. As we are dealing with semantic web, there might be millions of groups or categories in a single knowledge domain and we have to find a best way of representing all those groups on a single page, where the user can easily navigate from the first node to the last node.

Coming to the Hypergraph, we used GraphXML for the visualization and in future we can implement Hypergraph with GraphXML automated grouping of categories of same depth and showing the semantics between different nodes by assigning different colors to them. We can also show the semantics related to a category by the variation of color in Radialgraph. In this report we generated XML of the entire domain at a time and in future we have to work on clustering of data and generation of XML’s according to the user choice (clusters of data). The SQL query to generate the XML is static in future we have to make it dynamic so that the visuals are of user choice. Like Hypergraph, for Radialgraph and Treeview we need to work on source code of them so that they can also have some reference hyperlinks that can call some methods or applets at each node.
7.2 Conclusion

The visualizations, Hypergraph, Radialgraph, and Treeview of Vijjana, at this early stage help us to build a standard framework for Vijjana model Module. The visuals can be easily integrated in a webpage and thereby allowing access to all the users. Once if the User Interface is ready and if the users contribute URL’s, we can straight away direct that to search agent through Hypergraph, where the entire list of URL’s of a particular category results were displayed on that page. The standard taxonomy and the semantics that were manually added can be shown with these visualizations. This report can acts as a base for the implementation of visualization agent and new features of Vijjana Model.
References