**3. DESIGN**

**3.1 Software requirement specification**

Software sometimes can be a complex entity. Its development usually follows what is known as Software Development Life Cycle (SDLC). The second stage in the SDLC is the Design Stage. The objective of the design stage is to produce the overall design of the software. The design stage involves two sub-stages namely:

a. High-Level Design

b. Detailed Design

In the High-Level Design, the Technical Architect of the project will study the proposed applications functional and non-functional (qualitative) requirements and design overall solution architecture of the application, which can handle those needs.

High Level Design precisely discusses an overall view of how the system should work and the top-level components that will form the system that work to achieve the proposed solution. It should have very little detail on implementation, i.e. no explicit class definitions, and in some cases not even details such as database type (relational or object) and programming language and platform.

In our project, the high-level design stage consists of decomposing the project into top-level modules of getting the search string, getting ambiguous paragraphs, formatting the search string, training the key words and eliminating ambiguity for various contexts .which are explained in further parts of the chapter.

**3.1.1Overall description**

The Issues that we need to address before considering a design for implementing our system are:

* Mapping of qwerty keyboard into alphabets
* Length of the Search string.
* Training of various keywords for all the contexts.
* Searching the keyword in the paragraph for specific context.
* Displaying the meaning of the selected keyword according to the context.

3.1.1.1 Assumptions and Dependencies

• We are using word -net dictionary to develop our application along with hash table for database design.

• The Operating System used for testing is Windows XP and higher versions of windows.

**3.1.2 Specific Requirements**

Hardware requirement involves:

* Pentium 4 processor
* 128 MB RAM
* 50 MB hard disk space (the size increases when more and more information is populated)

The end user is assumed to have a little knowledge about how to operate a computer, GUI’s are provided for context based information retrieval for users. The application being developed right now needs no change, but future changes will be documented.

**3.1.3 Design constraints**

3.1.3.1 General Constraints

The general constraints of the system are as listed below.

• The system in which our application is installed is expected to have minimum of 50 MB hard disk space, this is because we are storing the database consisting of keyword along with corresponding meaning.

• Graphical User Interface is being provided, and every care is taken to see that the UI is easy to understand and manageable by even a computer layman

• To improve the disambiguating capabilities in our application, it is necessary to have a fast processor (1.7 GHz P4 Processor)

• As mentioned above, we require a minimum of 128 MB RAM and 50 MB Hard disk space to retrieve information at faster pace

3.1.3.2 Development Method

Although we did consider many strategies for developing this system, the one that suited us best was the functional development approach. In this method, we list out the different functions that we can have in our application and start developing them. The functional dependencies are maintained throughout this process. Also to be noted is the input –output data that flows from one function to another function remains same. To Aid us in functional development of the system, we have designed the Data Flow Diagrams and the Structured Charts, which are mentioned in the section 3.4.

**3.1.4 Interfaces**

We have developed the system using the Top-Down approach. In this method, the entire system is visualized a whole, latter each component of the system is split into modules, which, in the next level are further divided into sub-modules and so on.

* Eclipse-IDE has been used to develop the application.
* Making an application platform independent requires the usages of higher level languages, which can achieve platform independence using the Java programming language.
* User Interface developed in java swing enables a layman to easily carry out the search function and retrieve corresponding context based meaning needed.

**3.2 High level design**

**3.2.1 Design considerations**

3.2.1.1 System Architecture

The approach used in our application development is a function-based approach, where in we classify the system into different functions, which can perform certain operations on the given input and give us the required output. Thus based on this approach, we classify our World Sense Disambiguation software into three basic functionalities namely

* Pre-Processing of the Input provided by the user.
* Training of keywords for various contexts.
* Processing of the entire paragraph for mapping of context.
* Searching of relevant meaning for the entered input at the back end

Pre-processing is the stage where we take users input which is a keyword compact the provided input by a process called as morphological analysis.

Training is the stage where we take the meaning from the word net dictionary and provide various clues for artificial intelligence of recognizing the various contexts. The keyword inputted by the user is converted in to a morphs. Searching of relevant contexts based on the clues provided by trained data.

3.2.1.2 System Architecture (DFDs / Structured Charts / Use Cases)

The Architecture of the system can be explained with the aid of Data Flow Diagrams, Structured Charts and Use Cases. The Data Flow Diagrams indicate the movement of data within the system and the transformations that act on the data. The Structured Charts indicate the decomposition of the system into various modules and also show the interactions between those modules. The Use case diagrams show how users use the system. The use case diagram presents the activity to the users.

**3.2.2 Data Flow Diagrams**

The Data Flow Diagrams indicate how the data in the system flows from one module to another and what are all the transformations that takes place on the data. In Data flow diagrams the importance is given to the data and not on the control .The notations used in the dataflow diagram are:

* Rectangles that denote external entity
* Circles or bubbles that indicate the function or module
* Labeled Arrows that indicate data flow from one module to another module.

The DFDs for a system can be decomposed into several levels. The ‘level 0’shows the overall modules in the system and the data flow between these modules. Level1 DFD for the system shows data flow within a module, Level 2 DFD for the system shows data flow in sub-module and so on. The level 0 DFD for our system is shown below

Level 0 DFD:

Level 0 DFD for the system shows the major modules in the system. Input module takes the paragraph of words as the input and also takes the keyword as the search string. The context based information retrieval module using W.S.D generates the context based meaning of the selected keyword.

End User

End User

Context based meaning of the keyword

Paragraph and

keyword

Fig 3.1 DFD for World Sense Disambiguation System (Level 0)

Level 1 DFD:

Level 1 DFD for the system shows 4 modules. Input module takes the paragraph of words and the keyword as the input. The training module trains the clue words for mapping. The mapping and context analysis modules generate the context based meaning of the keyword.

Meanings based on context

List of Meanings

Paragraph

Search String

End User

End user

Database containing words and its meanings

Fig 3.2 DFD for World Sense Disambiguation System (Level 1)

Level 2 DFD:

Level 2 DFD shows the major modules. Tagging module tags each word in the paragraph to its respective parts of speech. Training module trains the clue words. The extraction module helps to extract meanings from the database. The mapping module maps the meanings and the W.S.D module helps to find the context based meaning.

End User

End User

Meanings based on context

Paragraph and keyword

List of meanings

Words with part of speech tag

Words with meanings

List of clue words

Words /meanings

Word net dictionary

Fig 3.3DFD for World Sense Disambiguation System (Level 2)

**3.3 Detailed design**

**3.3.1 Structured Charts**

A Structure Chart (SC) in software engineering and organizational theory is a chart, which shows the breakdown of the configuration system to the lowest manageable levels. This chart is used in structured programming to arrange the program modules in a tree structure. Each module is represented by a box, which contains the module's name. The tree structure visualizes the relationships between the modules.

Structured Charts are the tools that help in software design to decompose the system into a set of functional modules, and identify their interactions with one another. The structured charts are mainly useful to determine the work break down structure of modules within the system.

A structure chart is a top-down modular design tool, constructed of squares representing the different modules in the system, and lines that connect them. The lines represent the connection and or ownership between activities and sub activities as they are used in organization charts.

With the help of a structured chart of the design of the system following things can be inferred.

* the size and complexity of the system,
* number of readily identifiable functions and modules within each function and
* Whether each identifiable function is a manageable entity or should be broken down into smaller components.

A structured chart for the Word Sense Disambiguation system is as shown in figure in the next page.

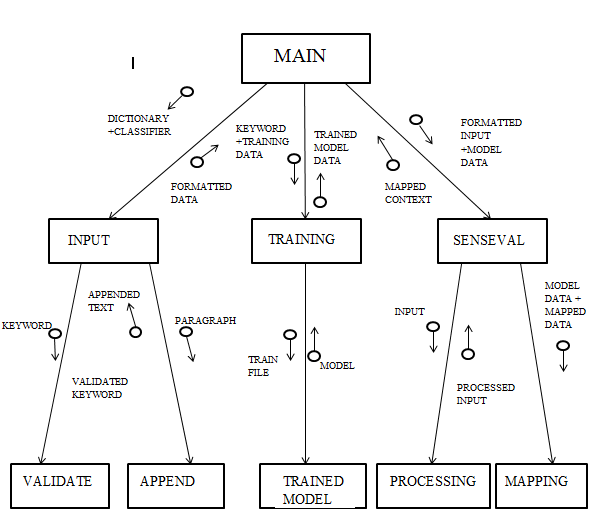


Fig 3.4 Structure diagram

**3.3.2 Module description**

3.3.2.1 Module-1: Input Module

The Input Module is an important module that takes all the needed input from the user. For this module Dictionary is passed as argument which contains definitions of lexical items and their senses. There's no top-level element, just a sequence of lexical elements.

The dictionary defines the words and their senses. For instance, the verb activate has five senses, numbered 38201 to 38205.The source of the word senses is indicated by the source attribute. The value for this entry indicates the senses are drawn from Word Smyth, which has the following entry for "activate".

Entries for English nouns have their source attribute set to wn, with word senses was drawn from Word Net. Other sources were used for the other languages. The synset attribute provides a list of synonyms for the word, drawn from the source indicated. The gloss attribute provides a dictionary style definition for the term.

Algorithm:

Start:

Print “enter the context”;

Read context;

Print “select keyword”;

Read Keyword;

While!=end of context

Begin:

Validate;

If(success) Move next;

Else

Begin:

print “error message”;

exit;

End

End

Append formatted text

Return;

End

Input:

A Context and a Keyword to be selected from Drop down menu.

Output:

The Input Module outputs the following:

* A validated Keyword.
* Appended Text.

3.3.2.2 Module-2: Training Module

Purpose:

The module trains the list of keywords for various senses and sets the mapping clues in the trained model.

Functionality:

The module accepts the input keyword and Training File as a input. Like the dictionary, there is no top-level element wrapping everything, as required in XML. Instead, for each word for which there is training data is enclosed in lexelt pseudo-tags.

Algorithm:

Start:

While!(end of training file)

Begin:

Tag the parts of speech

Explore the meaning from dictionary database

Create a model with specific clue words

End

Update the model

Return trained model

End

Input:

Training file and Keyword list.

Output

The Training Module outputs the following:

* Trained Model

### 3.3.2.3 Module-3: Senseval Module:

Purpose: The Module Maps the senses to corresponding context from the trained model and displays the specific meaning based on the usage of keyword.

Functionality:

The module accepts trained model and test data as arguments. Test data is formatted to the specific requirement. Then formatted input is sent to mapping along with training model containing clue words.

Algorithm:

Start:

Process the input

While!(end of test data)

Begin:

Explore the context of keyword appearance

Search for a specific match in the trained model

If(success)

map the corresponding sense and output the result

else

map it to default meaning of keyword and output the result

end

end

The Module Maps the senses to corresponding context from the trained model provides.

Input:

The Senseval Module accepts:

* Model Data
* Mapped Data

Output:

The Senseval Module outputs the meaning of the selected keyword which is generated depending upon the context of the entire paragraph.