Chapter 4
Implementation

4.1 Software Design Methodology

Object-oriented methodology of software development is selected for this system. This method of software development was first proposed in the late 1960s. However it took almost 20 years for object technologies to become widely used. During the first half of the 1990s object-oriented software engineering became the paradigm of choice for many software product builders and information system professionals. As time passes, object technologies are replacing classical software development approaches. Object technologies do lead to a number of inherent benefits that provide advantages at both the management and technical level.

Object technologies lead to reuse, and reuses of program components lead to faster software development and higher-quality programs. Object-oriented software is easier to maintain because its structure is inherently decoupled. This lead to fewer side effects when changes have to be made and hence less frustration for the software engineer and the customers. In addition, object-oriented systems are easier to adapt and scale (ie. Large systems can be created by assembling reusable subsystems).

An object-oriented model of computer software exhibit data and procedural abstractions that lead to effective modularity. A class is an OO concept that encapsulates the data and procedural abstractions that are required to describe the content and behavior of some real world entity. The data abstractions (attributes) that describe the class are enclosed by a wall of procedural
abstractions. The only way to reach the attributes is to go through one of the procedural abstractions that comprise the wall. Therefore the class encapsulates the data and the processing that manipulates that data. This achieves information hiding.

4.2 Classes of the System

The classes identified for the general meaning representation system, machine translation system and natural language interface for databases are given in this section.

4.2.1 Meaning Representation System

The important classes identified for the system are given in fig 4.1.

<table>
<thead>
<tr>
<th>No</th>
<th>Name of classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Morphological-analyser</td>
</tr>
<tr>
<td>2</td>
<td>Local word grouper</td>
</tr>
<tr>
<td>3</td>
<td>Parser</td>
</tr>
<tr>
<td>4</td>
<td>Noun</td>
</tr>
<tr>
<td>5</td>
<td>Pronoun</td>
</tr>
<tr>
<td>6</td>
<td>Verbs</td>
</tr>
<tr>
<td>7</td>
<td>Modifiers</td>
</tr>
</tbody>
</table>

Fig 4.1 list of classes used in the general meaning representation system

The class definitions are given below.
Class morphological analyser
{
    protected:
        char query [ ];
        char querywords [ ][ ];
        char wordcategory [ ][ ];
        int wordpointer [ ];
    public:
        getquery ( );
        separatewords ( );
        getwordcategory ( );
};

Class localwordgrouper : public morphological analyser
{
    Protected:
        char wordgroups[ ][ ];
    Public:
        Wordgrouping ( );
};

Class parser : public localwordgrouper
{
    protected:
        char Vroot[ ];
        int Verb form[ ];
        char Tense[ ];
}
int Vperson;
char Vmodifier [ ];
char KKroot[ ];
int KKcategory;
char KKmvalue[ ];
int KKmcategory;
int KKnumber;
int KKgender;
int KKperson;
char KRroot[ ];
int K Rcategor y;
char KRmvalue[ ];
int K Rmcategor y;
int KRnumber;
int KRgender;
int KRperson;
char SKroot [ ];
int SKcategory;
char SKmvalue[ ];
int SKmcategor y;
int SKnumber;
int SKgender;
int SKperson;
char SMroot[ ];
int SMcategory;
char SMmvalue[ ];
int SMmcategor y;
int SMnumber;
int SMgender;
int SMperson;

    char HKroot[ ];
    int HKcategory;
    char HKmvalue[ ];
    int HKmcategory;
    char AKroot[ ];
    char AKdestination[ ];
    char AKtime[ ];
    char AKdate[ ];

public:
    get_properties_of_words();
    fill_slots();
    show_structure();

};

Class noun
{
    private:
        char root[ ];
        struct form
        {
            char suffix[ ];
            char vibakthi[ ];
        }ST;
        int gender, number, person;
        char semantic properties[ ];

public:
add_values_to_lexicon();
search_value();
}

Class pronoun
{
private:
  char root[];
  struct form
  {
    char suffix[];
    char vibakthi[];
  }ST;
  int gender, number, person;
public:
  add_values_to_lexicon();
  search_value();
};

Class Modifier
{
private:
  char root[];
  int category;
public:
  add_values_to_lexicon();
  search_value();
};
Class Verbs
{
    private:
        char root[];
        struct form
        {
            char suffix[];
            int verbform;
        }ST;
    public:
        add_values_to_lexicon();
        search_value();
};

The algorithm of the meaning representation program is as given below.

1. Display the start screen.
2. Read query.
3. Separate the words in the query.
   Get categories of words.
   Get the pointers to word properties.
   Make word grouping.
   Get the properties of the words.
4. Fill the slots of the meaning representation structure.
5. Display the meaning representation structure.
4.2.2 Machine Translation System

It makes use of all the classes given in section 4.2.1. In addition to that it has a class which generates sentences in English. Its definition is given below.

Class generator : public parser
{
    Protected:
        Char wordgroups[ ][ ];

    Public :
        Pattern_identification ( );
        Generate_English_sentences ( );
        Display_sentences ( );
}

4.2.3 Natural language Interface System

Class morphologicalanalyser
{
    protected:
        char query [ ];
        char querywords [ ][ ];
        char wordcategory [ ][ ];
        int wordpointer[ ];

    public :  
        getquery ( );
        separatewords ( );
        getwordcategory ( );
}

104
Class `localwordgrouper` : public morphological analyser

```cpp
{
  Protected:
  char conditionalwordgroups[ ][ ];
  char outputwordgroups[ ];

  Public:
  Wordgrouping ();
  DeepLevelEllipsisHandler();
}
```

Class `parser` : public `localwordgrouper`

```cpp
{
  protected:

  char Vroot[ ];
  char kkQweryWord [ ];
  char kkAttribute [ ];
  char kkEntity[ ];
  struct karmakarakam
  {
    char krValue [ ];
    char krAttribute [ ];
    char krRelation :
    char krQuantifier [ ];
  }kr[ ];

  public:
  get_properties_of_words();
```
fill_slots();
AnaphoricReferenceHandler();
SurfaceLevelEllipsisHandler();

Class SQL_generator : public parser
{

Protected:
    char SQL_statement[];

Public:
    ProcessVerbs();
    ProcessKanthaKarakamList();
    ProcessKarmaKarakamList();
    JoinTables();
    FormSQLStatement();

};

Class Response_generator: public SQL_generator
{

private:
    int error;

public:
    displayResults();
    displayErrorMessages();
    CheckException();
    CheckTemporalEvent();

};
Class propernoun
{
    private:
        char root[];
    struct form
        {
            char suffix[];
            char vibakthi[];
        }ST;
        int gender, number, person;
    char dbfieldnames[][ ];

    public:
        add_alues_to_lexicon();
        search_value();

};

Class common_noun_1
{
    private:
        char root[];
    struct form
        {
            char suffix[];
            char vibakthi[];
        }ST;
        int gender, number, person;
    char databasefildnames[][ ];
public:

    add_values_to_lexicon ( );
    search_value ( );

};

Class common_noun_2
{
private:

    char root [ ];

    struct form
    {
        char suffix [ ];
        char vibakthi [ ];
    }ST
    int gender, number, person;
    char databasetablenames[ ][ ];

public:

    add_values_to_lexicon ( );
    search_value ( );

};

Class Verbs
{
private:

    char root [ ];
    char tablesInvolved[ ];

public:

    add_values_to_lexicon ( );
    search_value ( );

}
Class qweryWord
{
    private:
        char root [ ];
        char databasefieldnames[ ][ ];
    public:
        add_values_to_lexicon ( );
        search_value ( );
};

Class relationWords
{
    private:
        char root [ ];
        char operator:
    public:
        add_values_to_lexicon ( );
        search_value ( );
}

Class quantifier:
{
    private:
        char root [ ];
    public:
        add_values_to_lexicon ( );
        search_value ( );
};
Chapter 5
Performance Evaluation of the Model

5.1 Introduction

There are two basic types of performance evaluation. Black box evaluation and glass box evaluation. Black box evaluation measures system performance on a given task in terms of well defined input/output pairs and glass box evaluation examines the internal working of the system. Black box evaluation focuses on the accuracy of the output, user-friendliness, modularity, portability and maintainability. Black box evaluation is done without knowing anything about the inner workings of the system. The NLP systems could be judged on the basis of the following aspects

Coverage and Habitability

Coverage is a characterization of the linguistic competence of a system. Habitability measures how quickly and comfortably a user can recognize and adapt to system's limitations. The coverage of the NLP system could be measured along dimensions like lexical coverage, syntactic coverage and semantic coverage [61]. Lexical coverage refers to the size of the vocabulary, internal structure of the vocabulary and the easiness with which the vocabulary could be extended. Syntactic coverage refers to the range of syntactic phenomena the system can deal with. It includes complex verb forms, relative clauses, various question forms, passives, comparatives, subordinate clauses, ellipsis etc. Semantic coverage refers to the extent to which the system understands the domain. The critical issues regarding coverage are whether the system has enough coverage to let users meet a
reasonable proportion of their needs, whether the user can quickly find an appropriate way of expressing a request and whether the user can easily learn to avoid the system's blind spot.

A system's habitability is reduced if the user is led to believe that the system has capabilities that are beyond it and there is no clear indication of the boundaries. This can happen if the language the system presents to the user is not the language that the user can present to the system. The difficulties in achieving habitability with a semantic grammar are based on the fact that without great care such grammars can give users misleading clues as to coverage.

Inference
This is the process of drawing logical conclusions based on the data in the database or general knowledge of the subject domain. Retrieving only data that is explicitly stored in a database is usually insufficient to meet a normal user's needs. The system should have the capability to infer new information from that already existing in the database.

Anaphora
Pronouns are special case of the linguistic phenomena called anaphoric reference. Pronouns usually refer objects explicitly mentioned in previous discourse. Sometimes they can refer to objects mentioned later. Pronouns can also refer to actions. NLP systems should be capable of handling the usage of pronouns. Sometimes pronouns refer to objects in the computer's previous response, not just objects in the user's own language.

Ellipses
In conversation, people often leave out large portion of sentences, assuming that the listener, who shares the context being discussed, can fill in the missing parts. A good NLP system should be capable of handling any kind of ellipses.
Quantification

The use of words like some, every, all and any can complicate NLI system because their interpretation often depends on wide ranging commonsense knowledge or on detailed knowledge of the particular domain. The NLI system should be capable of tackling quantification appropriately.

Presentation of output

This includes formatting reports and tables, interfacing to graphics modules and generating output in the user’s own language.

5.2 Performance of the NLI System for Information Retrieval

The black box testing of the system was carried out with three databases. An election database, an academic database and a library database. A summary of the performance of the system is given in the table 5.1. The system was evaluated by giving about 30 typical queries from each application domain. The queries included simple straight forward queries, multi-relational queries, queries with conjunctions, and quantifiers, elliptical queries and queries with anaphoric references.

The table shows that the accuracy of the system is 100% for simple straight forward queries. The queries of this type manipulates only a single relation for getting the output. For multi-relational queries , excluding queries with anaphoric / elliptical references the accuracy of the system is around 90 %. The accuracy of the anaphoric / elliptical queries is around 85% . Accuracy could be increased by adding more rules to the knowledge base used by the parser module of the system.
<table>
<thead>
<tr>
<th>Database Names</th>
<th>No. of Questions asked</th>
<th>Query type</th>
<th>Percentage of Queries correctly interpreted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Straight forward queries</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Conjunctions</td>
</tr>
<tr>
<td>Election</td>
<td>30</td>
<td>Straight forward Queries</td>
<td>15%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multi Relational Queries</td>
<td>85%</td>
</tr>
<tr>
<td>Academic</td>
<td>30</td>
<td>Straight forward Queries</td>
<td>25%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multi Relational Queries</td>
<td>75%</td>
</tr>
<tr>
<td>Library</td>
<td>30</td>
<td>Straight forward Queries</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multi Relational Queries</td>
<td>80%</td>
</tr>
</tbody>
</table>

Table 5.1. Performance Table
The Lexicon creator program could be used for creating the lexicon for the various application domains. Since the content of the lexicon is very much dependent on the structure of the database, a person who is thorough about the database schema is responsible for the development of it. No change in the meaning representation format is required. But the tables and some rules associated with the SQL generator are to be rewritten since the rules in this module are domain dependent. For handling modifiers, which refer to special procedures and verbs which refer to the relationships between the basic entities, new rules are to be added to the SQL generator module. Hence a person having at least the proficiency of a programmer is required for customizing this system for any database. The system could be scaled to larger databases.

Glass box evaluation of the system focuses on the performance of the important functional modules of the system. In this case the performance of individual modules like morphological analyser, parser, semantic interpreter, response generator, SQL generator etc were evaluated separately. A black box evaluation of a particular component's performance could be considered as a form of glass box evaluation. For example the evaluation of a parser with a set of specified input/output pairs would be a black box evaluation of the parser. Since it is an evaluation of a component that cannot by itself perform an application, and since it will give information about the component's coverage that is independent of the coverage of the system in which it might be embedded, this can be considered as providing glass box information for the overall system [62]. The testing of the individual modules were carried out when the system was under development.

The table 5.2 gives a comparison of the various size aspects of the three databases. The important fact derived from the table is that the size of the
lexicon is significantly small when compared to the size of the database. An important issue in NLIs to DBMS is the relative size of the lexicon as compared to the size of the database itself [63,64]. As mentioned chapter 3, the lexicon stores all words that are to be understood by the system. While attribute values are stored repeatedly in the database, they are stored only once in the lexicon. For example, in the election database, the party name is stored for each candidate in the database, while party names are stored only once in the lexicon. Also numeric values and code values are not stored in the lexicon. These two factors make the size of the lexicon significantly small when compared to the size of the database.

Some of the typical queries processed by the systems are given next. First the text in Malayalam is given. Then the Roman notation of it is given. Then query in English language, which is equivalent to it, is given next. Finally the SQL statement and result of query processing are given.

1. പാല സഭയിൽ എത്ര ഉപായകർ പങ്കെടുന്നതാണോ?
   pAIA niyojk mNtlwwi2l ewR szWanaRWik2L m2lsriccu?
   (How many candidates contested from Pala constituency?)

   SELECT COUNT(candidate.candname)
   FROM candidate, contest, cons
   WHERE candidate.candcode = contest.candcode AND
     contest.conscode = constituency.conscode AND
     constituency.consname = 'pAIA'

   candidate.candname

8
<table>
<thead>
<tr>
<th>Database Names</th>
<th>No. of Tables</th>
<th>Table Names</th>
<th>Rows Per Table</th>
<th>No. of Attributes per table</th>
<th>Ratio of Size of Database to Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Election</td>
<td>5</td>
<td>Party</td>
<td>4</td>
<td>2</td>
<td>5 : 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Constituency</td>
<td>100</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Candidate</td>
<td>400</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contest</td>
<td>400</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Win</td>
<td>100</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Student</td>
<td>100</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quarter</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Courses</td>
<td>30</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Department</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Teacher</td>
<td>17</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Entrolls</td>
<td>800</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>offers</td>
<td>30</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Academic</td>
<td>7</td>
<td>Books</td>
<td>1500</td>
<td>5</td>
<td>7 : 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Borrower</td>
<td>100</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Circulation</td>
<td>400</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Publisher</td>
<td>10</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Topic</td>
<td>10</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Library</td>
<td>5</td>
<td>Books</td>
<td>1500</td>
<td>5</td>
<td>3 : 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Borrower</td>
<td>100</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Circulation</td>
<td>400</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Publisher</td>
<td>10</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Topic</td>
<td>10</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2 Database size comparison table
2. თავისი წლის ასაკი ღმერთთანავითარი იქნა?  
woOmsz enn szWnaRWiyute vyz ewR?  
(What is the age of the candidate Thomas?)

SELECT candidate.age  
FROM candidate  
WHERE candidate.candname = 'woOms'

Candidate.age  
56

3. 70 წლის ასაკში არ გაუდინენ მომხმარებლებმა თავად?  
70 vySi2l kutuwluLL szWanaRWiK2l ArellAM?  
(List the candidates with age greater than 70.)

SELECT candidate.candname  
FROM candidate  
WHERE candidate.age > 70

Candidate.candname  
1. ke. KruNAkrV  
2. ti. PqAVsisz  
3. vrcgIsz  
4. eIl.seviyrc  
5. seviyrcaRkkIl  
6. eM.pi.pHiOsz  
7 eM.pi.mANi

4. სახელმწიფოს მფლობელს სამეგრელოში თავად სტალ?  
KOQgrsz pA2Rtti2l wORRv2R ArellAM?  
(List the candidates who have lost the contest in Congress party.)
SELECT candidate.candname
FROM candidate, win, party
WHERE candidate.candcode <> win.candcode AND
    candidate.partyname = party.partyname AND
    party.partyname = 'kOQgrsz'

candidate.candname
1. pi.ke.pwRosz
2. eM.pi.mANi

5. വിഖ്യാതിയായ നിരക്കുള്ള കരിഡ് എന്ത് കിട്ടി? (What is the total number of votes won by BJP party?)

SELECT SUM (contest.vote)
FROM contest, party, candidate
WHERE contest.candcode = candidate.candcode AND
    candidate.partyname = party.partyname AND
    party.partyname = 'bi.je.pi'

contest.vote
10,11156

6. കോട്ടയം സിസ്റ്റമിലെ മിക്ക നാമാംശങ്ങൾക്കു തെക്കിയത് എന്ന്? (Who are the Congress candidates who have won from kottayam district?)

SELECT candidate.candname
FROM win, candidate, constituency, party
WHERE candidate.candcode = win.candcode AND
    win.conscode = constituency.conscode AND
    candidate.partyname = party.partyname AND
    party.partyname = 'kOQgrszkA2R ArellaM'
SELECT constituency.consname
FROM contest, candidate, constituency, party
WHERE candidate.candcode = contest.candcode AND
  contest.conscode = constituency.conscode AND
  candidate.partyname = party.partyname AND
  party.partyname = 'kOqgrsz'

constituency. Consname

1. pAlA
2. wOtupuV
3. krinAgppLLi

SELECT candidate.candname
FROM win, candidate, party
WHERE candidate.candcode = win.candcode AND
  candidate.partyname = party.partyname AND
  party.partyname = 'kOqgrsz'


candidate.candname

1. e.joQsz
2. sumM

9. பலா லீன் மறையுத்திய பழக்கம் பெற்றவர் என்று?  
   pAlAy2l jyiccwz Arz?
   *(Who has won from Pala?)*

   SELECT candidate.candname 
   FROM candidate, win, constituency 
   WHERE win.candcode = candidate.candcode AND 
   Win.conscode = constituency.conscode AND 
   Constituency.consname = 'pAlA'

Candidate.candname

1. sjJv2n

10. பிரவியத்தை சான்றா மறையுத்திய விளக்கச் சொற்றொக்கிய கையெழுத்து?  
    pRoPs2R wOmsz pTippicc ViYyf2L Ev?
    *(Give the names of courses offered by Prof. Thomas.)*

   SELECT course.coursename 
   FROM teacher, offers, course 
   WHERE teacher.teachid = offer.teachid AND 
   offers.courseid = course.courseid AND 
   teacher.teachname = 'wOmsz' AND 
   teacher.designation = 'pRoPs2R'
11. What are the names of all students of the Electronics department?

```
SELECT student.name
FROM student, department
WHERE student.deptid = department.deptid AND department.deptname = 'Electronics'
```

12. How many students are there in the Electronics department?

```
SELECT COUNT(student.studname)
FROM student, department
WHERE student.deptid = department.deptid AND department.deptname = 'Electronics'
```
13. dERRAbEsz kOVzi2nte AvREjz mA2Rkkz ewR
(what is the average mark of database course?)

SELECT AVG(entrolls.mark)
FROM entrolls, course
WHERE entrolls.courseid = course.courseid AND
course.coursename = 'dERRAbEsz'

entrolls.mark
56.5

14. 100l kutuw21 vixyA2RwWikL uLL dipARRum2neRz EwellAM
(Which departments have more than 100 students?)

SELECT department.deptname ,COUNT(student.studname)
FROM department, student
WHERE student.deptid = department.deptid
GROUP BY department.deptname
HAVING COUNT(student.studname) > 100

Department.deptname
1. mawwmaRRikzs

15. kMpuUt2RgrAPikzs2neR ewR puswkff2L untz?
How many books are there for computer graphics?

SELECT COUNT (book.accno)
FROM book, subcode
WHERE book.scode = subcode.scode AND
subcode.string = 'kMpuUt2RgrAPiksz'
14. க.அச எவ்வும் புத்துக்காட்சியை எதிர்ப்புணரார்?

k.AS ewR puswkff2L etuwwittuntz?

(How many books did K.Asha take?)

```
SELECT COUNT (circulation.Idno)
FROM circulation, borrower
WHERE circulation.idno = borrower.idno AND
Borrower.name = 'k.As'
```

5.3 Performance of the Machine Translation System

In the system for machine translation, the lexicon creator object is used for building the lexicon. The vocabulary is spread over 4 dictionaries. About 2500 root words are stored in the dictionaries. Declarative sentences in the active voice are analyzed. Both simple and complex sentences are considered. Since in a general context anaphoric references are difficult to consider, sentences are treated isolated. Each sentence is analyzed independently, with the result that a previous sentence does not affect the interpretation of the following sentence. Any ambiguity in this regard is transferred from source language to destination.
Several sentences were translated using this system. Some sample sentences are given in table 6.3.a and table 6.3.b. The sentences are coded using the Roman notation given in appendix – 4. In the tables “M” denotes the Roman notation of the sentence, “E” denotes the English Translation and “Pattern” denotes the pattern to which the Malayalam sentence is mapped. Sentences in pure Malayalam language is also given with in brackets.

The story about the Lion and the Rat translated by the system from Malayalam to English is given next. The text in Malayalam is given in table 6.4 and the text in Roman notation is given in 6.5 and the English translation obtained is given in table 6.6.

The sentences were selected in such a manner that they could be mapped to one of the 15 patterns selected in chapter 3. All the sentences generated correct translation. The analyzer's ability was tested further by letting it handle more types of sentences. During this extension stage, it was found that more verb forms should be included, more semantic tags were required for word sense disambiguation, more patterns should be added to the generator and more rules regarding karaka sharing should be included. The performance of the system increases by incorporating more knowledge to the system.
1. M. kutti av2neR PuszwkM vAyiccu  
   E. The child read his book.  
   Pattern. (Subject+Verb+Object)

2. M kARRz atikkunnu  
   E. The wind is blowing.  
   Pattern. (Subject+Verb)

3. M rAmu rAjuvinz pe2nsi2l kotuwwu  
   E. Ramu gave Raju a pencil  
   Pattern. (Subject+Verb+In-direct object+direct object)

4. M. rAmu pAwrff2L vqwweyAyi  
   E. Ramu washed the plates clean  
   Pattern. (Subject+Verb+Object+Object complement)

5. M av2n bAgz wiryunnu  
   E. He is searching for the bag  
   Pattern. (Subject+Verb+preposition+Prepositional Object)

6. M rAju eyuwwz aykkuvA2n mRnnu  
   E. Raju forgot to post the letter  
   Pattern. (Subject+Verb+to-Infinite)

Table 6.3a Some Sample Sentences Translated
E. He told me to go there
Pattern. (Subject+Verb+Indirect object+to-infinite)

8. M. av2n pAlM ktkkunnWz fA2n kNtu
E. I saw him crossing the bridge.
Pattern. (Subject+Verb+Present Participle+clause)

9. M. suXyute pe2nsi2l etuwwittz rAmu ptM vrccu
E. Raman drew a figure having taken Sudh’s pencil
Pattern. (Subject+Verb+Object+Perfect Participle+clause)

10. M. e2neR pErz viLiccwz FA2n kettu.
E. I heard my name called.
Pattern. (Subject+Verb+Object+Past participle+clause)

11. M. kutti pTikkukyANennz amm pRffu
E. Mother told that the child was
Pattern. (Subject+Verb+that-clause)

E. The cat went out when the door was opened.
Pattern. (Subject+Verb+when-clause)

Table 6.3b Some Sample Sentences Translated
<table>
<thead>
<tr>
<th>Table 6.4 Malayalam text of the story “The Lion and the Rat”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. തുണിക്കറി ഏഴ് പ്രണമിക്ക എന്ന് തുടർന്ന് ദോഷക്കാരിമാണ്.</td>
</tr>
<tr>
<td>2. എന്ന് പറഞ്ഞു കഠിഞ്ഞ.</td>
</tr>
<tr>
<td>3. എന്ന് പറഞ്ഞു പ്രണമിക്കുന്ന ലോകത്തിൽ കാട്ടിക്കുമ്പോൾ നോക്കി.</td>
</tr>
<tr>
<td>4. പ്രണമിക്കുന്ന് വെളുത്തു മാലു.</td>
</tr>
<tr>
<td>5. പ്രണമിക്കുന്ന നീണ്ടുകാണിക്കാൻ പറഞ്ഞു.</td>
</tr>
<tr>
<td>6. എന്ന് പറഞ്ഞു തകർക്കാന് പറഞ്ഞു.</td>
</tr>
<tr>
<td>7. എന്ന് പറഞ്ഞ പ്രണമിക്കുന്ന പസി തക്കാലോടെപ്പോള് പറഞ്ഞു.</td>
</tr>
<tr>
<td>8. പ്രണമിക്കുന്ന പ്രണമിക്കുന്ന.</td>
</tr>
<tr>
<td>9. എന്ന് പറഞ്ഞ പോൾ ഈ പ്രണമിക്കുന്ന അപ്‌ലിക്ക് നീണ്ടു.</td>
</tr>
<tr>
<td>10. എന്ന് പറഞ്ഞ് പ്രണമിക്കുന്ന അപ്‌ലിക്ക്.</td>
</tr>
<tr>
<td>11. എന്ന് പറഞ്ഞ് പ്രണമിക്കുന്ന അപ്‌ലിക്ക്.</td>
</tr>
<tr>
<td>12. എന്ന് പറഞ്ഞു പ്രണമിക്കുന്ന ലോകത്തിൽ കാട്ടിക്കുമ്പോൾ നോക്കി.</td>
</tr>
<tr>
<td>13. പ്രണമിക്കുന്ന പ്രണമിക്കുന്ന നീണ്ടു മാല.</td>
</tr>
<tr>
<td>14. പ്രണമിക്കുന്ന പ്രണമിക്കുന്ന പ്രണമിക്കുന്ന.</td>
</tr>
<tr>
<td>15. പ്രണമിക്കുന്ന പ്രണമിക്കുന്ന പ്രണമിക്കുന്ന.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

Table 6.5  Story in Roman Notation
1. Once upon a time a lion was sleeping in a forest.
2. A rat saw this.
3. That rat started to play near the lion.
4. The lion grew angry.
5. The lion said that it will eat the rat.
6. The rat said not to kill it.
7. The rat told that it will help the lion one day.
8. The lion set the rat free.
9. This lion fell into a trap one day.
10. It roared loudly.
11. The rat came quickly having heard it.
12. The rat cut open the trap with its teeth.
13. The lion came out of the trap.
14. The lion thanked the rat.
15. The lion went its way.

Table 6.6 English Translation of the Story
The results show that the method comprising the usage of verb boundaries as phrase markers, the karaka based approach of extracting meaning from phrases and the frame structure for meaning representation is an ideal approach for sentence comprehension in Dravidian languages. The object-oriented methodology used, facilitates portability and scaling up of this model.