Multi Objective Criteria for Selection of Manufacturing Method using NLP Parser

Mr. Girish R. Naik
Associate Professor
Production Department
KIT's College of Engineering
Kolhapur,India
girishnaik2025@gmail.com

Dr. V.A.Raikar
Principal Govt. Engineering College
Karwar, India
Var312@yahoo.com

Dr. Poornima G. Naik
Assistant Professor
Department of Computer Studies
Chh Shahu Institute of Business
Education and Research, Kolhapur, India
luckysankalp@yahoo.co.in

***Abstract*—** Installing a manufacturing method might be very expensive and time consuming project. Organization should examine and decide on how best to make this decision of selecting appropriate process meeting their requirements. In order to improve the manufacturing cycle more than 110 manufacturing processes have been proposed. The objectives aimed at and the functions focused on by these processes vary. The process should be flexible enough to accommodate reasonable changes in design. This poses a great challenge to a manager in selection of effective and economical manufacturing process. Different organizations have different objectives and based on their specific requirement they deploy suitable process conforming to their objective. Today's business scenario is highly competitive, complex and dynamic in nature which demands strategic planning meeting the challenges of changing time. In this paper we have made an attempt to enable the end user a quick selection of appropriate manufacturing method based on multiple objectives. The information pertaining to the method selection is stored in a persistent Relational DataBase Management System (RDBMS) which can be manipulated by the end user as the organizational objectives and the market needs change. The end user instead of querying the database directly will use the natural language, termed as Manufacturing Query Language (MQL) designed by us, which is interfaced with RDBMS using prolog. To implement MQL, we have defined a finite set of symbols, words and language rules, MQL grammar. The parse tree is constructed based on the grammar specified. The NLP query is parsed using NLP parser designed by us and the queries which are successfully parsed are evaluated by mapping them to the corresponding prolog query using Java interface to Prolog (JPL). Prolog rules are stored in three different prolog knowledge bases, mqlgrammar.pl, rules.pl, and methodrules.pl. NLP offers most flexible way to implement grammar which can be readily extended with least efforts and as such offers an efficient way of implementing rules in dynamically changing scenarios. Our current work focuses on a multiple objectives. In real situations multi objective and multi function criteria is required for the proper selection of the manufacturing method. Our future work involves modification of the tool and parser to take account of multiple objectives and functions.

Keywords- *JPL, Manufacturing Method, MQL, Natural Language Processing, NLIDB, Parser Tree*

*****

I. INTRODUCTION

Manufacturing methods are of many different types based upon the technological solution, or software solution or modern management methods to meet the organizational objectives. To assist managers in selecting the best method to achieve certain criteria, two mapping methods are available, one based on the objectives of the method and the other based on the functions that the methods may serve. Based on the maturity of the manufacturing company, a particular manufacturing method may focus on manufacturing hardware, auxiliary software support, production planning and control, next generation production management, processing manufacturing methods, commercial aspects, organization, advanced organizational manufacturing methods, design methods, human factors in manufacturing, environmental manufacturing methods, or cost and quality manufacturing methods. Giden Halevi has presented a review of manufacturing methods and their objectives [1]. The author has listed 110 published manufacturing methods which fall in 5 different classes based on their nature. In this paper we consider the following objectives as proposed by Giden Halevi in selection of a particular manufacturing method.

Meeting delivery dates

- Reduce production costs.
- Rapid response to market demands
- Reduce lead time
- Progress towards zero defects
- Progress towards zero inventory
- Improve management knowledge and information
- Marketing – market share
- Improve and increase team work collaboration
- Improve customer and supplier relationships
- Improve procurement management and control\
- Management strategic planning
- Improve human resources management
- Improve enterprise integration
- Continuous improvement
- Environmental production

The suitability of each method to a specific objective is graded according to the following grades.

a – Excellent for specific dedicated objective
b – Very good
c – Good
d – Fair

This paper focuses on assisting managers to evaluate and select the most appropriate manufacturing method or methods

for their needs, based on multiple objectives. Several alternatives may be proposed, allowing the user to decide which one is more suitable under the circumstances. The user can select the method according to its type. The decision depends on the objectives, the class considered, and on the grading given to each method. The objectives and grades can be manipulated by the end user. The information pertaining to different objectives, classes, and methods along with their grades is stored in a persistent relational database management system (RDBMS) which can be constantly updated by the end user as the organizational objectives change. The end user, instead of querying the database directly will use natural language which is interfaced with RDBMS using prolog. The NLP query is parsed using NLP parser and the queries which are successfully parsed are evaluated by mapping to the corresponding prolog query using JPL interface to prolog.

Introduction to Prolog

Prolog, Programming in Logic, is a special type of declarative type programming in which the various program elements and constructs are expressed in predicate logic. A program consists of mainly a number of declarations representing relevant facts and rules concerning the problem domain. The solution to be discovered is also expressed as a question to be answered or to be more precise the goal to be achieved based on the resolution method suggested by Robinson consisting of matching goals with facts and rules. A prolog program consists of a finite sequence of facts, rules and a query or goal statement. Prolog database or knowledge base consists of facts and rules. Prolog inferencing system mainly consists of three mechanisms viz.,

i) Backtracking
ii) Unification and
iii) Resolution.

Two interesting features of logic programming are non-determinism and backtracking. A non-deterministic program may find a number of solutions, rather than just one, to a given problem. Backtracking mechanism allows exploration of potentially alternative directions for solutions, when some direction currently being investigated, fails to find an appropriate solution.

Introduction to Natural Language Processing

Natural language processing (NLP) is a field that combines computer science technologies such as Artificial Intelligence, Statistical inference and Machine Learning with applied linguistics, concerned with the interactions between computers and natural human languages. Hence NLP is related to the area of human–computer interaction. It allows computer-supported understanding and processing of information expressed in human language for specific tasks, including machine translation, interactive dialog systems, opinion mining, etc. Many challenges in NLP involve natural language understanding by enabling computers to derive meaning from human or natural language input, and others involve natural language generation.

Natural Language Interface to Database

Most of the IT applications continuously store and retrieve information from databases, which requires knowledge of database languages like Structured Query Language (SQL). SQL norms and SQL proprietary extensions have been pursued in almost all the languages for relational database systems. However, for efficient data retrieval the knowledge of the structure of the database and core database concepts such as joining, filtering, query optimization are desirable. This led to the evolution of IDBS. There is an inevitable need for non-expert users to query relational databases in their natural language instead of working with the various SQL intricacies. As a result many intelligent natural language interfaces to databases have been developed, which provides flexible options for the manipulation of the queries. The idea of using Natural Language instead of SQL has prompted the development of new type of processing called Natural language Interface to Database (NLIDB). NLIDB is a step towards the development of intelligent database systems (IDBS) to enhance the users in performing flexible querying in databases. The transformation of a sentence entered in a natural language to the corresponding SQL query before execution against the database is depicted in the following Figure 1 and the corresponding flow of logic for execution of MQL command is depicted in Figure 2.

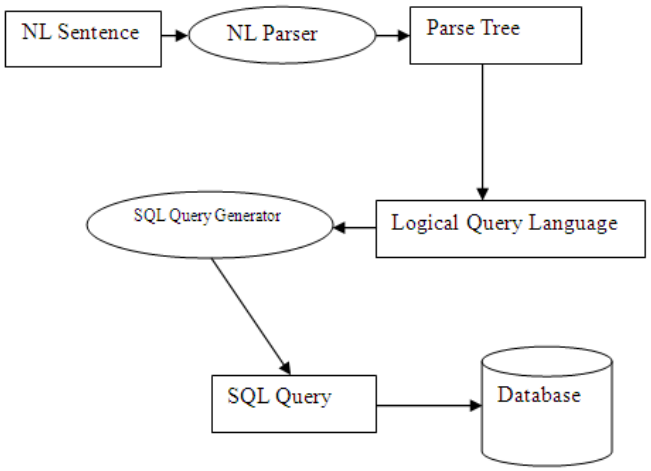


Figure 1. Natural Language Interface to Database

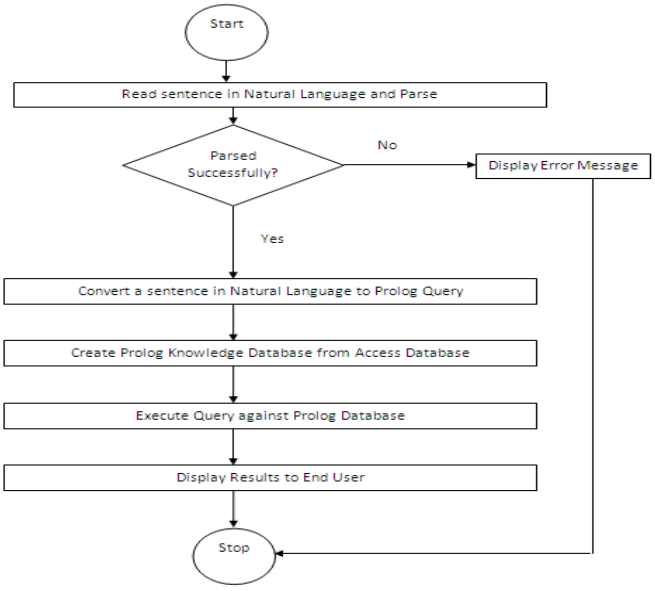
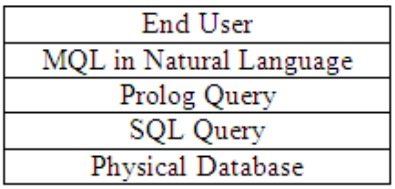


Figure 2. Execution of query in Natural Language Manufacturing Query Language (MQL)

A Manufacturing Query Language is designed which enables the end user to query the database in a human language without worrying about tedious SQL syntax. No formal knowledge of SQL is desirable. It provides a layer on top of SQL to render the query language end user friendly. The architecture is depicted in Figure 3.



II. LITERATURE SURVEY

There exists a vast amount of literature on manufacturing process monitoring using both crisp and fuzzy logic approach [3,9] which focus mainly on software selection, technology selection and system project selection. Chenhui Shao et.al [10] have developed a novel algorithm for parameter tuning and feature selection. Quality monitoring is used for monitoring a quality of a manufacturing process. Multiple criteria decision making method is employed by R. V. Rao, T. S. Rajesh [11]. The authors have presented a decision making framework using a multiple criteria decision making method viz., Preference Ranking Organization Method for Enrichment Evaluations (PROMETHEE) which has been integrated with analytic hierarchy process (AHP) and the fuzzy logic. The framework enables the manager a software selection in manufacturing industries. Mohammad Akhshabi [12] has developed a Fuzzy Multi Criteria Model for Maintenance Policy which is used for the optimized decision making.

The authors have developed different types of parsers for parsing MQL command using text parser, Finite State Automata Parser [13,15] which offer certain limitations in extensibility and are overcome in NLP parser. The following Table 1 depicts the relative comparison between the different approaches used for developing a parser for MQL.

Parser	Adaptability to Changes in query language	Query Optimization
Text Parser	Poor	Poor
Finite State Automata Parser	Moderate	Poor
Natural Language Processing	Excellent	Excellent

Table 1. Comparison between MQL Parsers

III. PROPOSED ALGORITHM

A. *Pseudo Code*

Pseudo code for parsing and evaluating MQL query in C++ notation is depicted below:

```
/* A function for transforming the sentence entered by the
user in Natural Language into a Prolog Query and executing
the query. The query checks the grammar of the sentence by
retrieving the information stored in Prolog knowledgebase
mqlgrammar.pl
   Input      –  MQL query entered by the user in Natural
Language
   Output  -  1 indicates that the query is successfully executed
              0  indicates that the query contains syntactical errors
 */

int function parseQuery(String sentence)
{
  tokens = sentence.split(“ “);
   query = ”[“;
   for (i=0;i<tokens.length;i++)
   {
query=query + tokens[i];
   }
  q1 = new Query(“consult(‘mqlgrammar.pl’)”);
  q2 = new Query(query);
  if (q2.query()== 1)
     return 1;
else
     return 0;
}

/*
   A function for evaluating the query using Natural Language
Interface to Database. MQL query is converted into the
corresponding Prolog query and the query on successful
execution retrieves the contents of Prolog knowledgebase
rules.pl and methodrules.pl
   Input   –  Successfully parsed MQL query
   Output  -  Results of the query stored in a string array.

*/
char[][] function  evaluateQuery(char[] query)
{
   tokens = query.split(“ “);
  l = tokens.length();
 if (I == 3)
 {
      if (tokens[2] == “methods”)
      {
         q3 = new Query(“list_all(X, methods).”);
      }
      else if (tokens[2] == “classes”)
      {
         q3 = new Query(“list_all(X, classes).”);
      }
      else if (tokens[2] == “objectives”)
      {
         q3 = new Query(“list_all(X, objectives).”);
      }
          q4 = new Query(“consult(‘rules.pl’)”);
          q4.query();
      result = q3.allSolutions();
 }
 else if (l ==5 && tokens[3] == “meeting”)
 {
   q4 = new Query(“list_all (X, methods, tokens[4]).”);
   result = q4.allSolutions();
 }
else if (tokens[3] == “meeting” && tokens[l – 2] == “in”)
 {
```

 ISSN: 2321-8169
3484 – 3493

```
        str1="(objective_for_method(X, ";
        str2=", a);objective_for_method(X, ";
        str3=", b))";
        for (i=1; i<=l; i++)
        {
          if (tokens[i] == "and")
          {
          query= query + str1 + tokens[i+1] + str2 + tokens[i+1]
+ str3;
          }
            query = query + ".";
          q5 = new Query("consult('rules.pl')");
          q5.query();
          q6 = new Query(query);
          result=q6.allSolutions();
        }
        else if (l > 5 && tokens[l-2] == "in")
        {
          str5=tokens[l-1];
                    if (str5 == "classm")
                    {
                    query = query + "method_in_class(X, m)";
                    }
              else if (str5 == "classp")
    {
                    query = query + "method_in_class(X, p)";
                    }
                    else if (str5 == "classs")
          {
                    query = query + "method_in_class(X, s)";
          }
          else if (str5 == "classt")
          {
                    query = query + "method_in_class(X, t)";
          }
          else if (str5 == "classx")
          {
                    query = query + "method_in_class(X, x)";
          }
          q7 = new Query("consult('methodrules.pl')");
          q7.query();
          q8 = new Query(query);
          result=q8.allSolutions();
        }
       return result;
    }
```

B. General syntax of 'List' MQL Command

A single MQL command viz., List is implemented at present which has the following syntax.

List All {Methods| Objectives |Classes} [Meeting {Objective1|Objective2|…|Objective16} AND {Objective1|Objective2|…|Objective16}..... [in Class {M|P|S|T|X}]].

List All Methods in Class {M|P|S|T|X}

The following notations are used

{a|b|⋯} → One clause from the group of clauses separated by | must be selected.

[..] → The clause specified is optional

The above semantics generates the following queries.

1. List All Methods
2. List All Objectives
3. List All Classes
4. List All Methods Meeting Objective<n>

where <n> can take any value between 1 and 16.

5. List All Methods Meeting Objective<n> in ClassM
   List All Methods Meeting Objective<n> in ClassP
   List All Methods Meeting Objective<n> in ClassS
   List All Methods Meeting Objective<n> in ClassT
   List All Methods Meeting Objective<n> in ClassX

where, n can take any value between 1 and 16.

6. List All Methods Meeting Objective<n> AND Objective<n> AND Objective<n> .....

   where, n can take any value between 1 and 16.

7. List All Methods Meeting Objective<n> AND Objective<n> AND Objective<n> ..... in ClassM
   List All Methods Meeting Objective<n> AND Objective<n> AND Objective<n> ..... in ClassP
   List All Methods Meeting Objective<n> AND Objective<n> AND Objective<n> ..... in ClassS
   List All Methods Meeting Objective<n> AND Objective<n> AND Objective<n> ..... in ClassT
   List All Methods Meeting Objective<n> AND Objective<n> AND Objective<n> ..... in ClassX

where <n> can take any value between 1 and 16.

Grammar for MQL.

To implement MQL, we have constructed a language by defining the rules which specify how to test a string of alphabet letters to verify. A finite set of symbols used in the language is given by

Σ = {a, c, i, l, m, o, p, s, t, x}

and a set of words over an alphabet is given by

L={list, all, methods, objectives, classes, meeting, objective1, objective2, objective3, objective4, objective5, objective6, objective7, objective8, objective9, objective10, objective11, objective12, objective13, objective14, objective15, objective16, in, classm, classp, classs, classx, classt}

Syntax and Semantics of a Natural Language

Languages are defined by their legal sentences. Sentences are sequences of symbols. The legal sentences are specified by a grammar.

Our first approximation of natural language is a context-free grammar. A context-free grammar is a set of rewrite rules, with non-terminal symbols transforming into a sequence of terminal and non-terminal symbols. A sentence of the language is a sequence of terminal symbols generated by such rewriting rules. For example, the grammar rule

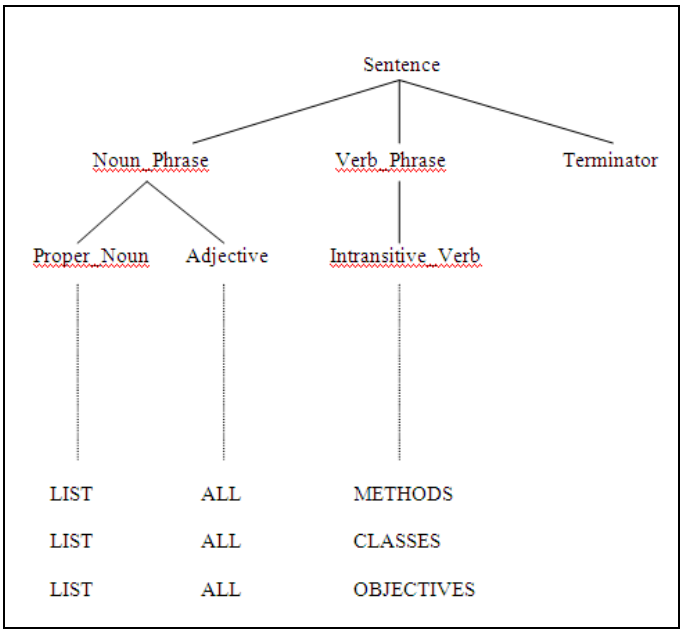
sentence→noun_phrase, verb_phrase

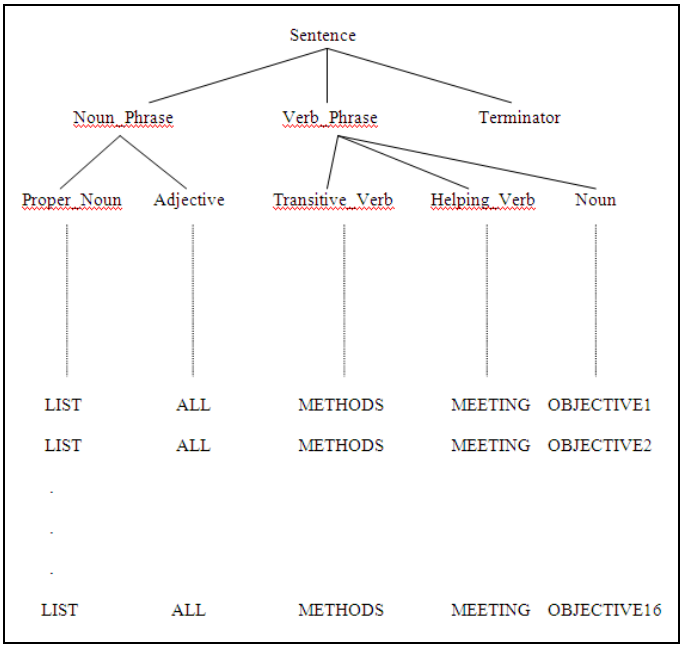
means that a non-terminal symbol sentence can be a noun_phrase followed by a verb_phrase. The symbol "→" means "can be rewritten as." The complete set of grammar for MQL is depicted below.

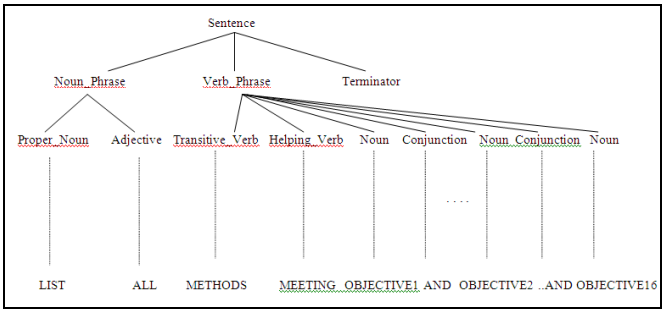
ISSN: 2321-8169
3484 – 3493

```
sentence --> noun_phrase, verb_phrase, terminator.

noun_phrase --> proper_noun, adjective.
noun_phrase --> determiner, noun.

verb_phrase --> intransitive_verb.
verb_phrase --> intransitive_verb, preposition, determiner.
verb_phrase --> transitive_verb, helping_verb, noun.
verb_phrase --> transitive_verb, helping_verb, noun, conjunction, noun.
verb_phrase --> transitive_verb, helping_verb, noun, conjunction, noun, conjunction, noun.
verb_phrase --> transitive_verb, helping_verb, noun, conjunction, noun, conjunction, noun, conjunction, noun.
verb_phrase --> transitive_verb, helping_verb, noun, conjunction, noun, conjunction, noun, conjunction, noun, conjunction, noun.
proper_noun --> [list].
adjective --> [all].

intransitive_verb --> [methods].
intransitive_verb --> [classes].
intransitive_verb --> [objectives].

transitive_verb --> [methods].

helping_verb --> [meeting].
noun --> [objective1].
noun --> [objective2].
noun --> [objective3].
noun --> [objective4].
noun --> [objective5].
noun --> [objective6].
noun --> [objective7].
noun --> [objective8].
noun --> [objective9].
noun --> [objective10].
noun --> [objective11].
noun --> [objective12].
noun --> [objective13].
noun --> [objective14].
noun --> [objective15].
noun --> [objective16].

conjunction --> [and].
conjunction --> [or].

preposition --> [in].
determiner --> [classm].
determiner --> [classp].
determiner --> [classs].
determiner --> [classt].
determiner --> [classx].

terminator --> ['.'].
terminator --> [].
```

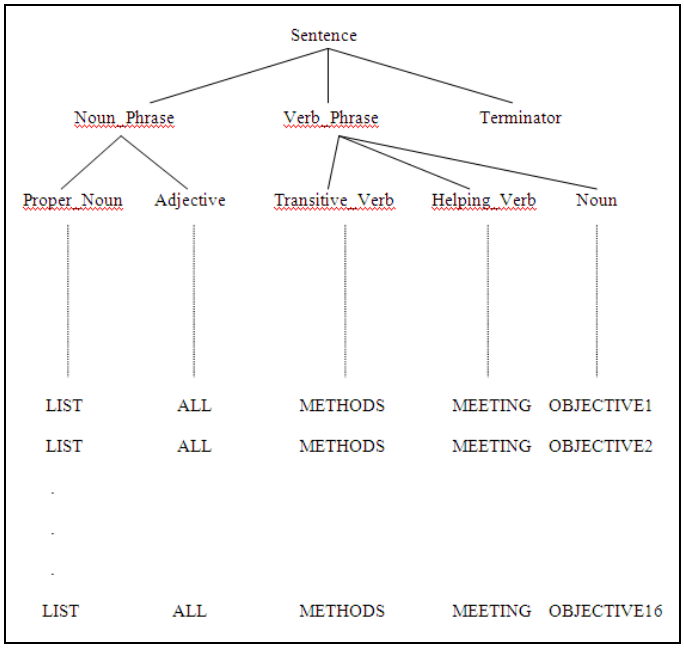
The corresponding parse tree is shown in Figures 4 (a) - 4 (e).







 ISSN: 2321-8169
3484 – 3493



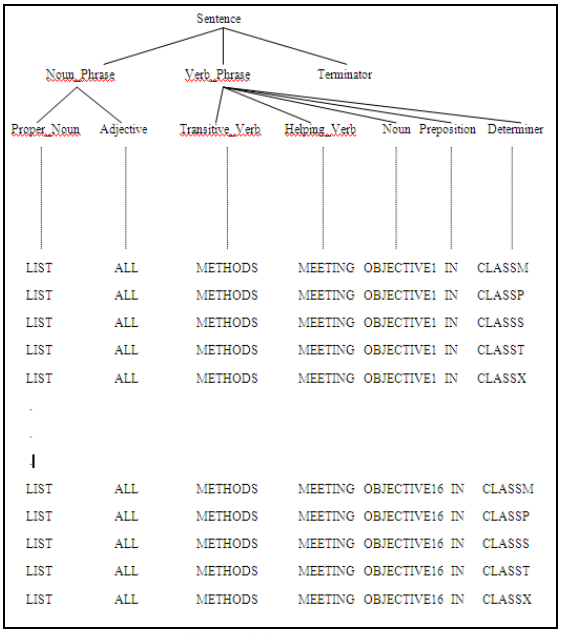


Figure 4(a) - 4 (e) MQL Parse Trees

Evaluation of MQL Commands

The prolog rules for retrieving all method names, class names, objective names, methods in a particular class and methods meeting one or more objectives are constructed in the following format by retrieving the corresponding database information.

objective_for_method(<Method Proposed>, <Objective>, <Grade of the Objective>).

For the current domain and the problem under consideration 110 method name rules, 5 class name rules and 16 objective name rules are generated and stored in a prolog database rules.pl.

Data Cleaning

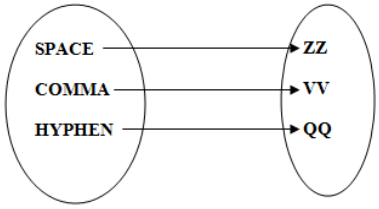


Figure (5) Data Cleaning by Character Set Mapping

The format of the sample prolog facts are depicted below:

list_all(activityqqbasedzzcosting, methods).

list_all(agentqqdrivenzzapproach, methods).

list_all(technologicalzzsolutionsvvzzrequireszzhardwarezzresources, classes).

list_all(softwarezzsolutionvvzzrequireszzcomputer, classes).

list_all(meetingzzdeliveryzzdateszzqqzzproductionzzplanningzzandzzcontrol, objectives).

list_all(reducezzproductionzzcosts, objectives).

list_all(agentqqdrivenzzapproach, methods, classm).

list_all(agilezzmanufacturing, methods, classm).

list_all(bioniczzmanufacturingzzsystem, methods, classp).

list_all(commonqqsensezzmanufacturing, methods, classp).

list_all(activityqqbasedzzcosting, methods, classs).

list_all(benchmarking, methods, classs).

list_all(flexiblezzmanufacturingzzsystem, methods, classt).

list_all(manufacturingzzexecutionzzsystem, methods, classt).

list_all(knowledgezzmanagement, methods, classx).

list_all(mobilezzagentzzsystem, methods, classx).

The whole set of MQL query to Prolog query mapping is depicted in the following Table 2:

ISSN: 2321-8169
3484 – 3493

MQL Query	Equivalent Prolog Query
list all methods	list_all(X, methods).
list all classes	list_all(X, classes).
list all objectives	list_all(X, objectives).
list all methods in classm	list_all(X, methods, classm).
list all methods in classp	list_all(X, methods, classp).
list all methods in classs	list_all(X, methods, classs).
list all methods in classt	list_all(X, methods, classt).
list all methods in classx	list_all(X, methods, classx).
list all methods meeting objective1	(objective_for_method(X, objective1, a);objective_for_method(X, objective1, b)).
list all methods meeting objective1 and objective2	objective_for_method(X, objective1, a);objective_for_method(X, objective1, b)),(objective_for_method(X, objective2, a);objective_for_method(X, objective2, b)).
list all methods meeting objective1 and objective2 and objective3	(objective_for_method(X, objective1, a);objective_for_method(X, objective1, b)),(objective_for_method(X, objective2, a);objective_for_method(X, objective2,b)),(objective_for_method(X, objective3, a);objective_for_method(X, objective3, b)).
list all methods meeting objective1 in classm	((objective_for_method(X, objective1, a);objective_for_method(X, objective1, b)), method_in_class(X, m)).
list all methods meeting objective1 in classp	((objective_for_method(X, objective1, a);objective_for_method(X, objective1, b)), method_in_class(X, p)).
list all methods meeting objective1 and objective2 in classp	((objective_for_method(X, objective1, a);objective_for_method(X, objective1, b)), method_in_class(X, p)), ((objective_for_method(X, objective2, a);objective_for_method(X, objective2, b)),

Table 2. Generation of Equivalent Prolog Query from MQL Query

IV. RESULTS AND ANALYSIS

The results presented above are implemented in Java with MS-Access as backend for storing method and objective details. The structure of the database is shown in the following Figure 6.

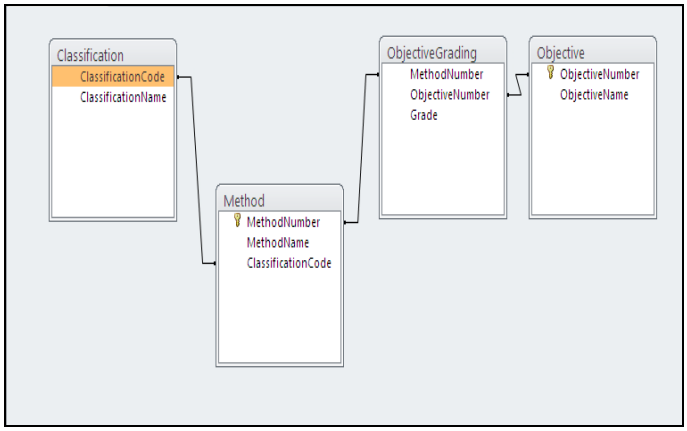


Figure 6. Structure of MQL Database

A prolog rule relating the method proposed for the given organizational objective along with the grade is constructed in the following format by retrieving the database information.

objective_for_method(<Method Proposed>, <Objective>, <Grade of the Objective>).

For the current domain and the problem under consideration 680 method rules and 110 class rules are generated and stored in a prolog database methodrules.pl. The format of the prolog facts are depicted below:

```
objective_for_method(method1, objective14, c).
objective_for_method(method1, objective2, c).
objective_for_method(method1, objective11, d).
objective_for_method(method1, objective7, c).
method_in_class(method1, s).
method_in_class(method2, m).
method_in_class(method3, m).
method_in_class(method4, x).
method_in_class(method5, p).
```

The format of the query for selection of manufacturing methods conforming to objective1 is as follows.

If (objective1.grade=a OR objective1.grade=b) then select method.

Where, objective1.grade refers to the grade of objective1.

The equivalent prolog query is:

Selected_methods:=objective_for_method(X,objective1,a); objective_for_method(X,objective1,b).

The format of the query for selection of manufacturing method conforming to objective2 and belonging to class M or S is as follows:

If ((objective1.grade=a OR objective1.grade=b) AND class=M) OR ((objective1.grade=a OR objective1.grade=b) AND class=S) THEN select method.

The equivalent prolog query is:

Selected_methods:=((objective_for_method(X,objective1,a); objective_for_method(X,objective1,b)),method_in_class(X,m));((objective_for_method(X,objective1,a); objective_for_method(X,objective1,b)),method_in_class(X,s)).

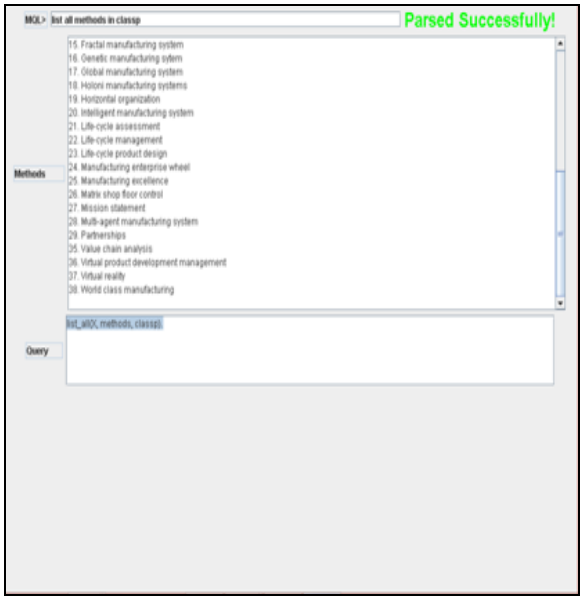
jpl.jar file contains the necessary java classes for interfacing with prolog. The structure of java program for executing prolog query is shown below:

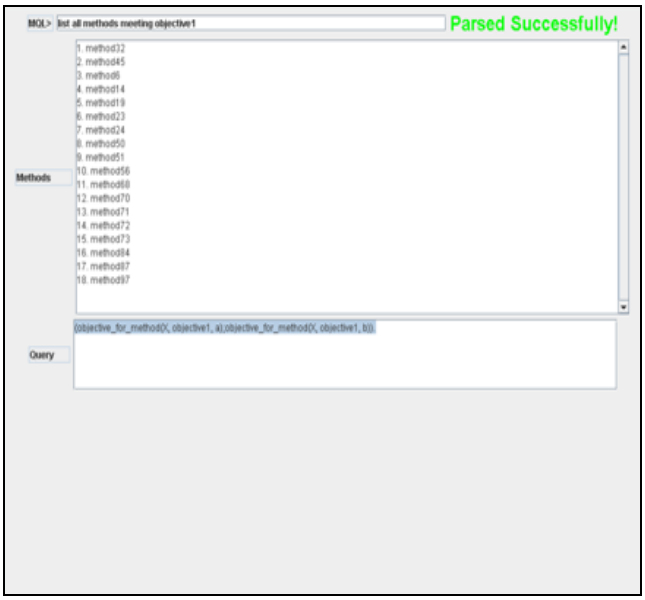
```
String t1 = "consult('methodrules.pl')";
Query q1 = new Query(t1);
FileOutputStreamfos=new FileOutputStream("c:\\methods.txt");
FileOutputStream fos1=new FileOutputStream("c:\\query.txt");
byte[] arr=new byte[20];
byte[] q=new byte[100];
String str;
System.out.println( t1 + " " + (q1.hasSolution() ? "succeeded" : "failed") );
String str1 = "((objective_for_method(X," + args[0] + ", a);objective_for_method(X, " + args[0] + ", b)),method_in_class(X,";String str2 = "))";
String t2="";
for (int i=1;i<args.length-1;i++)
{
        t2=t2+str1+args[i]+str2+";";
}
t2=t2+str1+args[args.length-1]+str2;
Query q2 = new Query(t2);
System.out.println( "first solution of " + t2 +
": X = " + q2.oneSolution().get("X"));
```

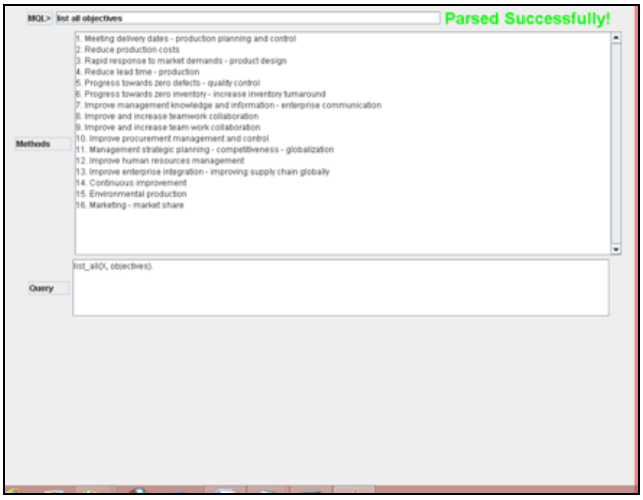
 ISSN: 2321-8169
3484 – 3493

```
q=t2.getBytes();
fos1.write(q);
//-----------------------------------------------------
java.util.Hashtable[] ss4 = q2.allSolutions();
System.out.println( "all solutions of " + t2);
for ( int i=0 ; i<ss4.length ; i++ ) {
System.out.println( "X = " + ss4[i].get("X"));
str=ss4[i].get("X").toString()+"\r\n";
arr=str.getBytes();
fos.write(arr);
}
fos.close();
fos1.close();
```

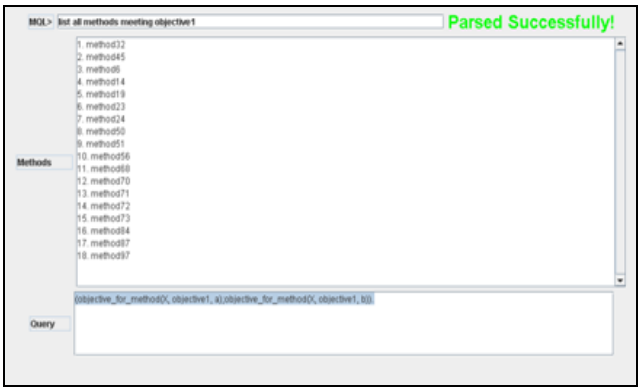
The result presented above is implemented in Java using SWI prolog and Java interface to prolog. The user interface is implemented in JFC swing and is presented in Figures 7 (a) – (j)

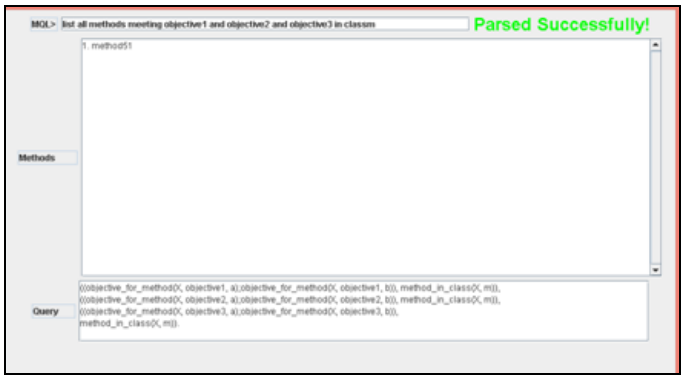






ISSN: 2321-8169
3484 – 3493





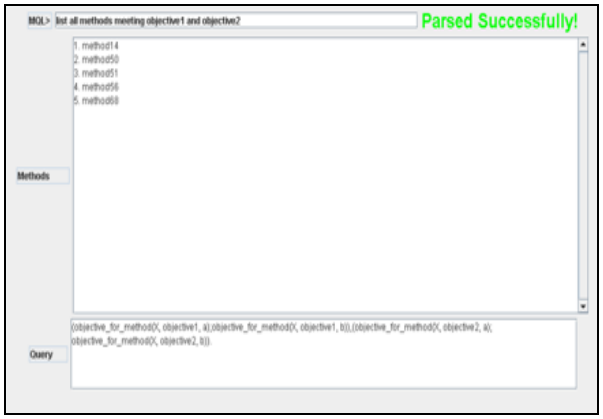


Figure 7 (a)-(j) User Interface for parsing and execution of MQL Commands.

V. CONCLUSION *AND FUTURE WORK*

This paper presents the design of an NLP parser for parsing human like query in the manufacturing domain which assists the manager in selection of a manufacturing method based on multiple objectives. A manufacturing query language (MQL) is designed to assist the manager to query a database in conventional language. A general syntax and a parse tree of a query language are presented. To implement MQL, we have designed our own language by defining a finite set of symbols, words and language rules, MQL grammar. . The NLP query is parsed using NLP parser designed by us and the queries which are successfully parsed are evaluated by mapping them to the corresponding prolog query using Java interface to Prolog (JPL). Prolog rules are stored in three different prolog with the view to incorporate distributed file system and distributed processing in future as data set grows. NLP offers most flexible way to implement grammar which can be readily extended with least efforts and as such offers an efficient way of implementing rules in dynamically changing scenarios. In this context, it clearly out scores other similar implementations of parsers such as text parsers, finite automata parsers etc. Our future work focuses on modification of the parser and the query language to incorporate multiple objectives and functions.

REFERENCES

[1] Gideon Halevi, Handbook of Production Management Methods, Butterworth Heinemann publications, ISBN 0 7506 5088 5.

[2] L. Mikhailov and M. G. Singh, "Fuzzy analytic network process and its application to the development of decision support systems," IEEE Transactions on Systems, Man, and Cybernetics, Part C. Applications and Reviews, Vol. 33, No. 1, pp. 33–41, 2003.

[3] R. Santhanam and G. J. Kyparisis, "A multiple criteria decision model for information system project selection," Computers & Operations Research, Vol. 22, No. 8, pp. 807–818, 1995.

[4] V. S. Lai, K. W. Bo, and W. Cheung, "Group decision making in a multiple criteria environment: A case using the AHP in software selection," European Journal of Op-erational Research, Vol. 137, No. 1, pp. 34–144, 2002. C. C. Wei, C. F. Chien, and M. J. J. Wang, "An AHP- based approach to ERP system selection," International Journal of Production Economics, Vol. 96, No. 1, pp. 47– 62, 2005.

[5] J. P. Brans, B. Mareschal, and P. Vincke, "PROMETHEE: A new family of outranking methods in multicriteria analysis," Operational Research, Vol. 3, pp. 477–490. 1984.

[6] R. V. Rao, "Decision making in the manufacturing envi-ronment using graph theory and fuzzy multiple attribute decision making methods," Springer-Verlag, London, 2007.

[7] R. Santhanam and G. J. Kyparisis, "A multiple criteria decision model for information system project selection," Computers & Operations Research, Vol. 22, No. 8, pp. 807–818, 1995.

[8] Dhananjay R. Kalbande and G.T.Thampi, Multi-attribute and Multi-criteria Decision Making Model for technology selection using fuzzy logic, International Journal of Computing Science and Communication

Technologies, VOL. 2, NO. 1, July 2009. (ISSN 0974-3375)

[9] Journal of Micromechanics and Microengineering, Xuan F Zha and H Du, Manufacturing process and material selection in concurrent collaborativedesign of MEMS devices, 13, 509–522, 2003.

[10] Chenhui Shaoa, , Kamran Paynabarb, Tae Hyung Kima, Jionghua (Judy) Jinc, S. Jack Hua, J. Patrick Spicerd, Hui Wangd, Jeffrey A. Abelld, Feature selection for manufacturing process monitoring using cross-validation, Journal of Manufacturing Systems, Volume 32, Issue 4, October 2013, Pages 550–555

[11] R. V. RAO, T. S. RAJESH, Software Selection in Manufacturing Industries Using a Fuzzy Multiple Criteria Decision Making Method, PROMETHEE, Intelligent Information Management, 2009, 1, 159-165, December 2009

[12] Mohammad Akhshabi, A New Fuzzy Multi Criteria Model for Maintenance Policy, Middle-East Journal of Scientific Research 10 (1): 33-38, 2011

[13] Mr. Girish R. Naik, Dr. V.A.Raikar, Dr. Poornima G. Naik, Single Objective Criteria for Selection of Manufacturing Method, International Journal of Computer Science and Engineering (IJCSE) ISSN(P): 2278-9960; ISSN(E): 2278-9979 Vol. 3, Issue 2, Mar 2014, 35-46.

[14] Mr. Girish R. Naik, Dr. V.A.Raikar, Dr. Poornima G. Naik, Single Objective Single Function Criteria for Selection of Manufacturing Method, International Journal of Emerging Technology and Advanced Engineering, (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 2, February 2014, 182-190.

[15] Mr. Girish R. Naik, Dr. V.A.Raikar, Dr. Poornima G. Naik, Multi Objective Criteria for Selection of Manufacturing Method, International Journal of Advanced Research in Computer Science and Software Engineering ISSN: 2277 128X, Volume 4, Issue 7, July 2014, 989-1002.