ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

IMPLEMENTATION OF DFA PARSER FOR MANUFACTURING QUERY LANGUAGE TOKENS

Mr. Girish R. Naik*, Dr. V.A.Raikar, Dr. Poornima G. Naik
* Associate Professor, Production Department, KIT's College of Engineering, Kolhapur,India
Principal Govt. Engineering College, Karwar, India
Assistant Professor, Department of Computer Studies, Chh Shahu Institute of Business Education and Research, Kolhapur, India

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. Recently, we have developed a tool 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. In this paper we present a deterministic finite automata (DFA) parser developed by us for parsing MQL tokens. The state table and state diagrams are developed for different tokens of MQL identified by us. State information is stored in a persistent database management system as a measure towards improving efficiency and extensibility. Currently, MQL consists of only few commands and more commands will be added to MQL in near future.

KEYWORDS : Finite Automata, Manufacturing, Parser, State Table, State Graph, State Transition

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 •

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

- 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

In this paper we present a deterministic finite automata (DFA) parser developed by us for parsing MQL tokens. In our previous work we have presented DFA parser for parsing MQL sentences constructing a parse tree based on the grammar specified [2]. 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) [3].

DETERMINISTIC FINITE AUTOMATA
Deterministic Finite Automata (DFA) can be seen as a special kind of finite state machine, which is in a sense an abstract model of a machine with a primitive internal memory. It is a finite state machine that accepts/rejects finite strings of symbols and only produces a unique computation (or run) of the automaton for each input string. 'Deterministic' refers to the uniqueness of the computation.

A deterministic finite automaton M is a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, consisting of A finite set I of input symbols.

1. A finite set of states (Q)
2. A finite set of input symbols called the alphabet, Σ
3. A transition or next state function δ, $\delta : Q \times \Sigma \rightarrow Q$
4. A subset F of Q of accept or final states, $(F \subseteq Q)$
5. An initial or start state $(q_0 \in Q)$.

Let $w = a_1a_2 \ldots a_n$ be a string over the alphabet Σ. The automaton M accepts the string w if a sequence of states, $r_0, r_1, \ldots, r_n$, exists in Q with the following conditions:

$r_0 = q_0$
$r_{i+1} = \delta(r_i, a_{i+1})$, for $i = 0, \ldots, n-1$ where, $r_n \in F$.

In words, the first condition says that the machine starts in the start state q_0. The second condition says that given each character of string w, the machine will transition from state to state according to the transition function δ. The last condition says that the machine accepts w if the last input of w causes the machine to halt in one of the accepting states. Otherwise, it is said that the automaton rejects the string. The set of strings M accepts is the language recognized by M and this language is denoted by L(M).

LITERATURE SURVEY

There exists a vast amount of literature on manufacturing process monitoring using both crisp and fuzzy logic approach [4,12] which focus mainly on software selection, technology selection and system project selection. Chenhui Shao et.al [13] 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 [14]. 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 [15] has developed a Fuzzy Multi Criteria Model for Maintenance Policy which is used for the optimized decision making.

In our current work each state is identified to be present in one of the set of initial, intermediate or final states. A function is constructed which accepts input token and current state as arguments and defines a unique transition to the next state which may be one of the intermediate or final states with the label Accept = 'N' or Accept='Y'. The input token is broken into the set of alphabets and the next state for each is evaluated using the state transition function given by
$f(a, q_c) = q_n$

where, a is the input alphabet, q_c and q_n are the current and the next states, respectively.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Recursively the function is invoked till all the alphabets of the given token are exhausted and the final state yielded on recursive function calls is examined. If it is labeled with the tag Accept='Y', the state is in one of the available final states. Hence the token is accepted, otherwise the token is rejected.

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

$\sum$ = {a, c, i, l, m, o, p, s, t, x}

and a set of words over an alphabet is given

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}

STATE GRAPH AND STATE TABLE MQL.

DFA is a set S of states that are connected by function f. A transition is an event of going from one state to another. DFAs are represented in two formats, Table and Graph. Graph representation of DFA for MQL alphabets is given by Figure 1 a) and the corresponding table representation is given by Figure 1 b

)

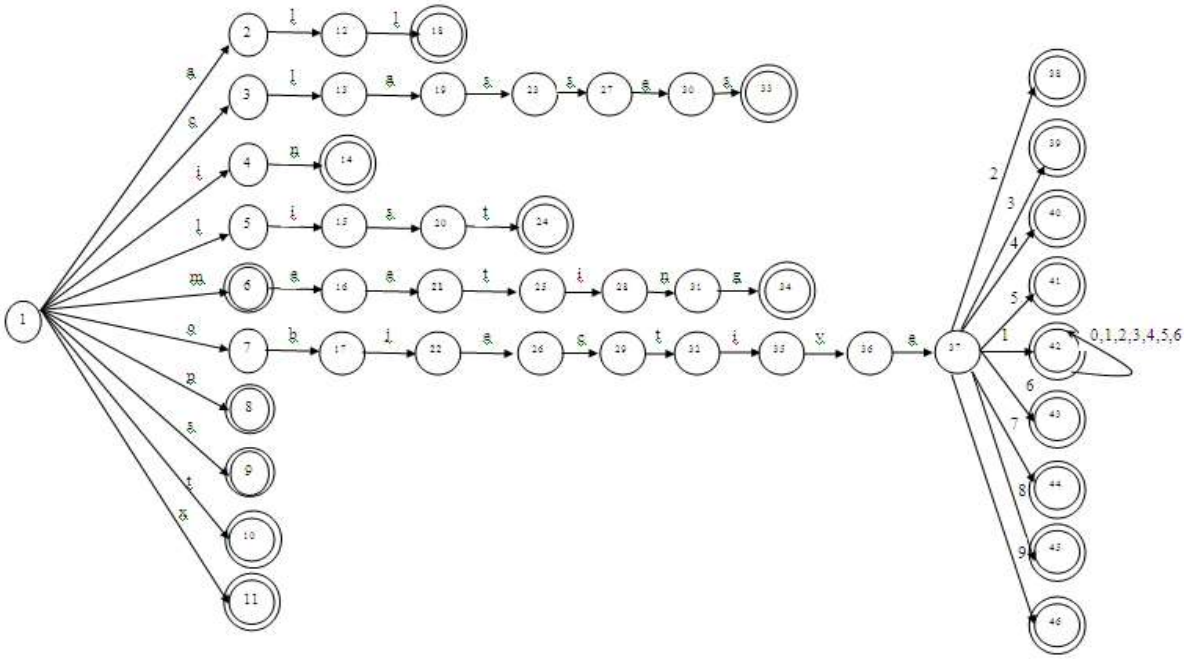


Figure 1 a) State Graph for MQL Alphabets.

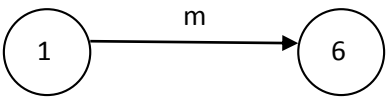
Sr.No.	Initial State	Input	Next State	Accept?	Sr.No.	Initial State	Input	Next State	Accept?

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

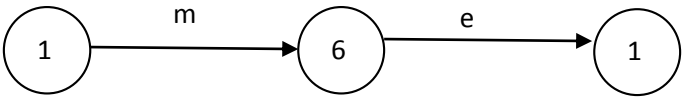
1	1	A	2	N	20	16	E	21	N
2	1	C	3	N	21	17	J	22	N
3	1	I	4	N	22	19	S	23	N
4	1	L	5	N	23	20	T	24	Y
5	1	M	6	Y	24	21	T	25	N
6	1	O	7	N	25	22	E	26	N
7	1	P	8	Y	26	23	S	27	N
8	1	S	9	Y	27	25	I	28	N
9	1	T	10	Y	28	26	C	29	N
10	1	X	11	Y	29	27	E	30	N
11	2	L	12	N	30	28	N	31	N
12	3	l	13	N	31	29	T	32	N
13	4	n	14	Y	32	30	S	33	Y
14	5	i	15	N	33	31	G	34	Y
15	6	e	16	N	34	32	I	35	N
16	7	b	17	N	35	35	V	36	N
17	12	l	18	Y	36	36	E	37	N
18	13	a	19	N	37	37	1	42	Y
19	15	s	20	N	38	42	0,1,2,3,4,5,6	42	Y

Figure 1 b) State Table for MQL Alphabets.

Figure 1 illustrates a deterministic finite automaton using a state diagram for the MQL alphabets. There are 38 unique states and 10 alphabets are identified to be in an initial state. The automata takes the finite sequence of alphabets as input and makes a transition to the next state on consuming the input. For example, if the automata is currently in the state 1, and if the input 'a' is given to the automata, it makes A transition to the next state 2. If the current state of the automata is 1 and if the input 'c' is given to it, it makes a transition to the state 3 and so on. If the final state is identified by Accept ='Y', the given token is accepted, otherwise the token is rejected. For example, in order to check whether the token 'meeting' is accepted or not, we start with the initial state 1, on consuming alphabet 'm' the automata will make a transition to state 6 as seen from Table and the same is depicted in the following state transition.



Now, state 6 becomes the current state or automaton. On consuming the next alphabet 'e' it will make a transition to the next state 16 as shown below:



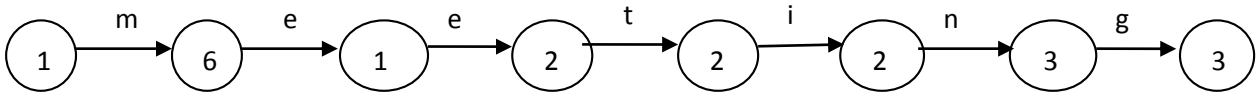
With 16 as the current state, the automaton will successively consume the alphabets 'e', 't', 'i', 'n' and 'g' making transitions to next states successively, identified by states 21, 25, 28, 31, and 34, respectively. The same states are highlighted in Figure 2.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

	State		State			State		State	
1	1	a	2	N	20	16	e	21	N
2	1	c	3	N	21	17	j	22	N
3	1	i	4	N	22	19	s	23	N
4	1	l	5	N	23	20	t	24	Y
5	1	m	6	Y	24	21	t	25	N
6	1	o	7	N	25	22	e	26	N
7	1	p	8	Y	26	23	s	27	N
8	1	s	9	Y	27	25	i	28	N
9	1	t	10	Y	28	26	c	29	N
10	1	x	11	Y	29	27	e	30	N
11	2	l	12	N	30	28	n	31	N
12	3	l	13	N	31	29	t	32	N
13	4	n	14	Y	32	30	s	33	Y
14	5	i	15	N	33	31	g	34	Y
15	6	e	16	N	34	32	i	35	N
16	7	b	17	N	35	35	v	36	N
17	12	l	18	Y	36	36	e	37	N
18	13	a	19	N	37	37	1	42	Y
19	15	s	20	N	38	42	0,1,2,3,4,5,6	42	Y

Figure 2. Parsing of token 'meeting'.

The entire transition is depicted in the following figure.



For the final state, the Accept column has the value 'Y'. Hence the final state is accepted and as such the token 'meeting' is accepted. On the contrary, the token 'meet' has a final state 25 for which Accept column has a value 'N'. Hence, the token 'meet' is not an acceptable string of the language.

PROPOSED ALGORITHM

The algorithm for parsing of MQL tokens is given below and the corresponding flow chart is depicted in Figure 3.

Step 1 : Read token and current state.
Step 2 : Split token into alphabets.
Step 3 : Read the next alphabet, current state and determine next sate.
Step 4 : Assign next state to current state.
Step 5 : Repeat step 3 and 4 till all the alphabets are exhausted.
Step 6 : Assign current state to final state.
Step 7 : If final state is acceptable, print "String is Acceptable" otherwise print "String is Rejected"
Step 8 : Stop }

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

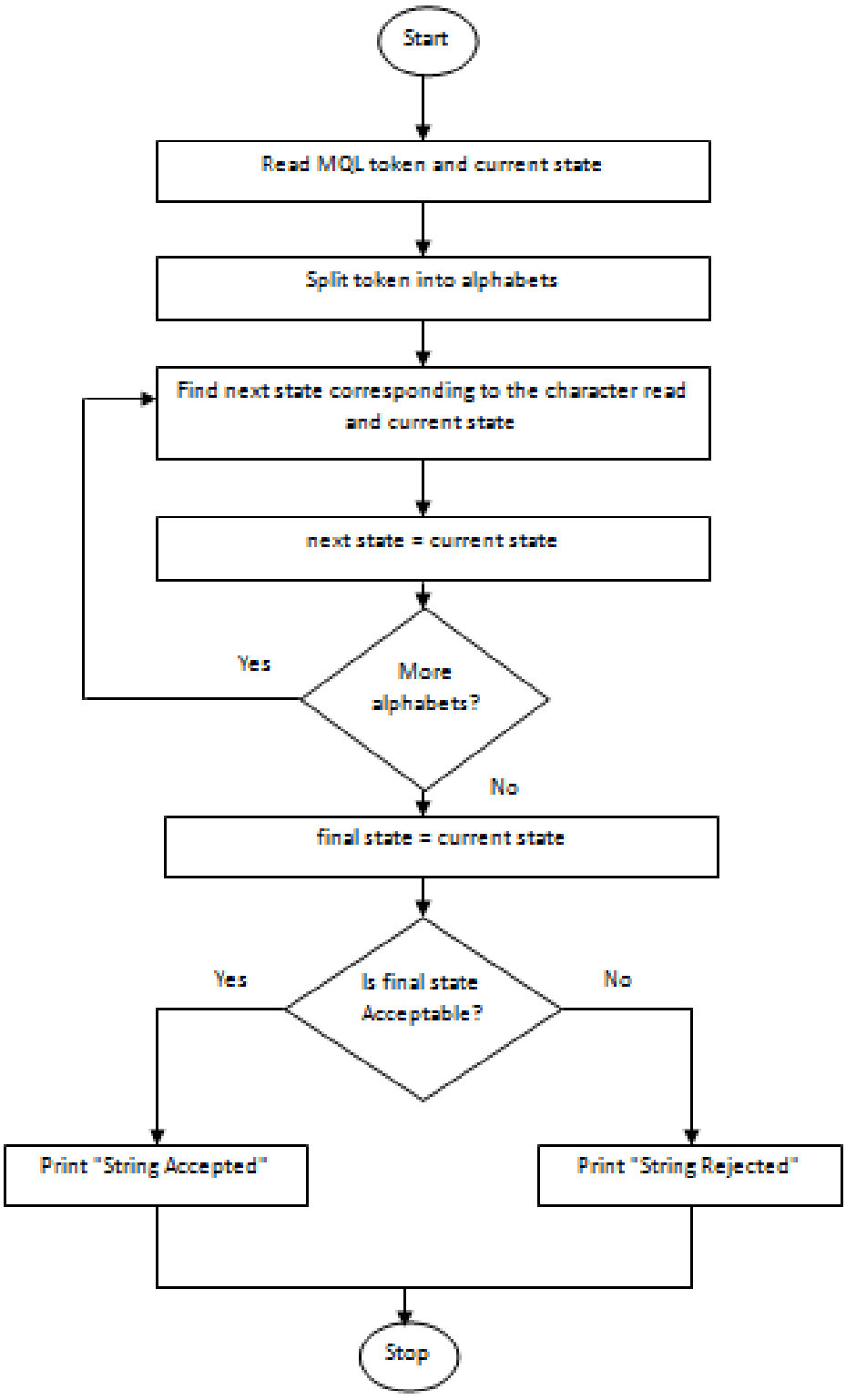


Figure 3. Flow chart for parsing of MQL tokens.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

PSEUDO CODE

Pseudo code for parsing MQL tokens in C++ notation is depicted below:

```
/* Structure for storing the state information in memory */

struct StateTransition
{
        char c;
        int currentState;
        int finalState;
        char accept;
}



/*
Function to parse given MQL token
Return Value : true, if the token is parsed successfully, otherwise false
*/

function boolean parseToken(char[] token, int currentState)
{
         /* Determine length of the token */
          length = strlen(token);
        count=0;
        while (count <= length)
        {
           nextState= findNextState(token[count]);
           currentState=nextState;
           if (currentState == 0)
              return false;
           count++;
        }
        finalState=currentState;
        accept = isAcceptable(finalState);
        if (accept == 'Y')
          return true;
       else
          return false;
}

/*
Function to determine next state for the given character and current state.
Return Value :    integer representing the next state, if the character c is found in the current state,
                  otherwise 0
*/

function int findNextState(char c, int currentState)
{
         /* Search 38 states for character c and current state  */
        /* st is an array of StateTransition structure of size 38 storing information
                  of each state in memory */
         for (i=1;i<=38;i++)
         {
            If (st[i].currentItem == currentState && st[i].c == c)
               Return st[i].finalState;
         }
        return 0;
}

/*
Function to determine whether the final state is acceptable.
Return Value : true, if the final state is acceptable, otherwise false.
*/

function boolean  isAcceptable(int finalState)
{
         /* st is an array of StateTransition structure of size 38 storing information
                  of each state in memory */
         for(i=1;i<=38;i++)
         {
            if (st[i].finalState == finalState)
                  return st[i].accept;
         }
         return false;
}
```

RESULTS AND ANALYSIS

The results presented above are implemented in Java with MS-Access as back end for storing state information. The structure of the state transition table is shown in Figure 4. Figure 5 depicts some sample states. The Graphical User Interface (GUI) is developed in Java Swing to accept a token from the end user. Figure 6 depicts the class diagram for implementation of MQL token parser in Java. Figure 7a) - 7g) show parsing of various MQL tokens with the corresponding state transition diagram to the final state.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

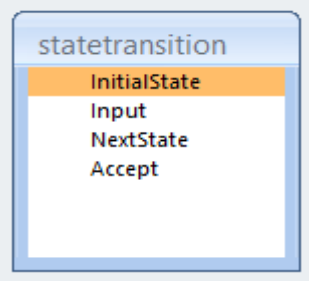


Figure 4. Structure of StateTransition Table

InitialState	Input	NextState	Accept
1	a	2	☐
1	c	3	☐
1	i	4	☐
1	l	5	☐
1	m	6	☑
1	o	7	☐
1	p	8	☑
1	s	9	☑
1	t	10	☑
1	x	11	☑
2	l	12	☐
3	l	13	☐
4	n	14	☑
5	l	15	☐
6	e	16	☐
7	b	17	☐
12	l	18	☑
13	a	19	☐
15	s	20	☐
16	e	21	☐
17	j	22	☐
19	s	23	☐
20	t	24	☑

Figure 5. Sample States

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

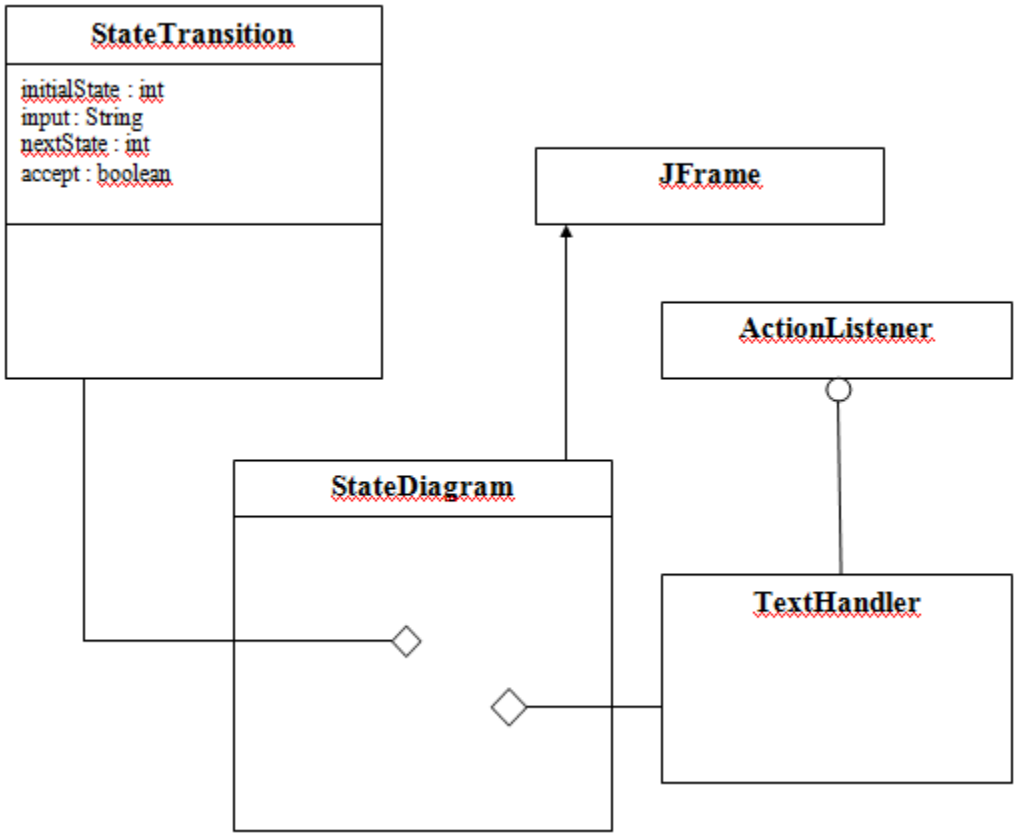
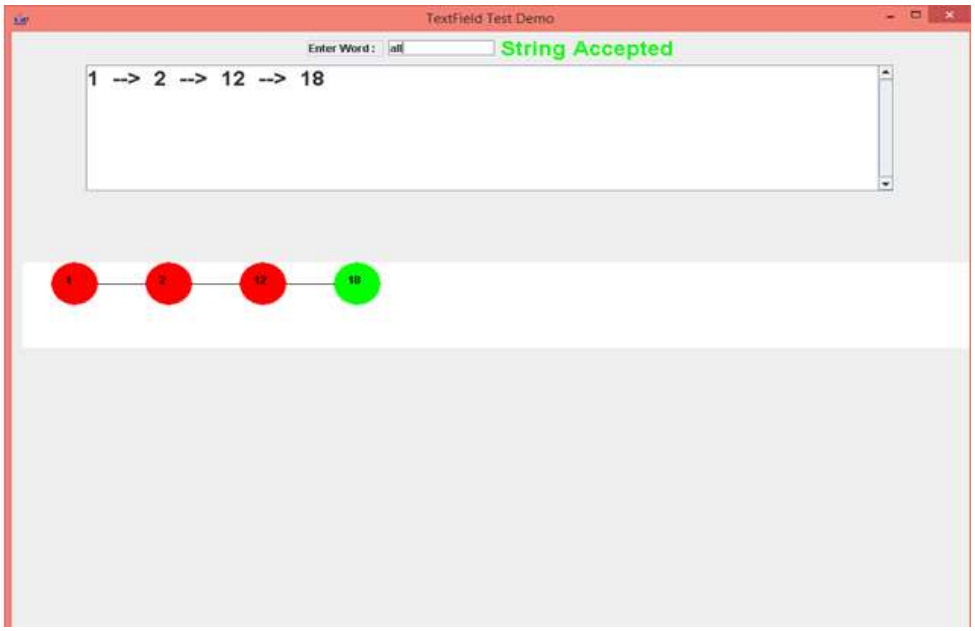
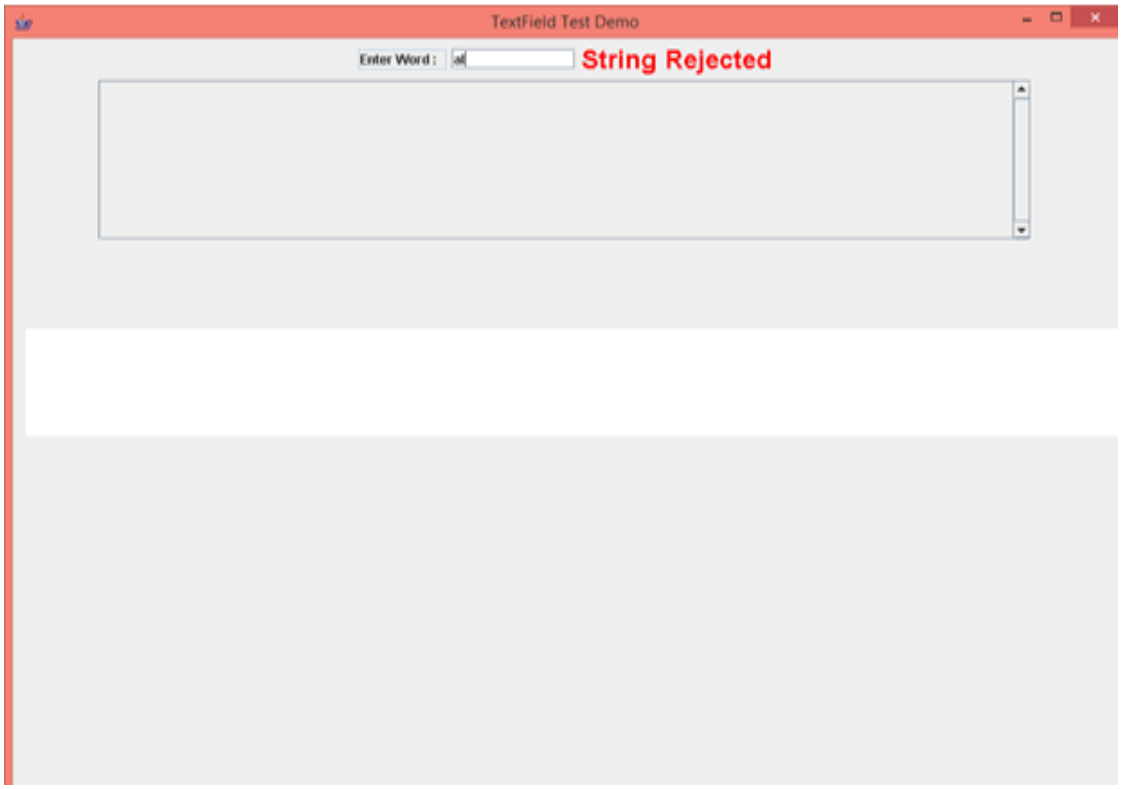
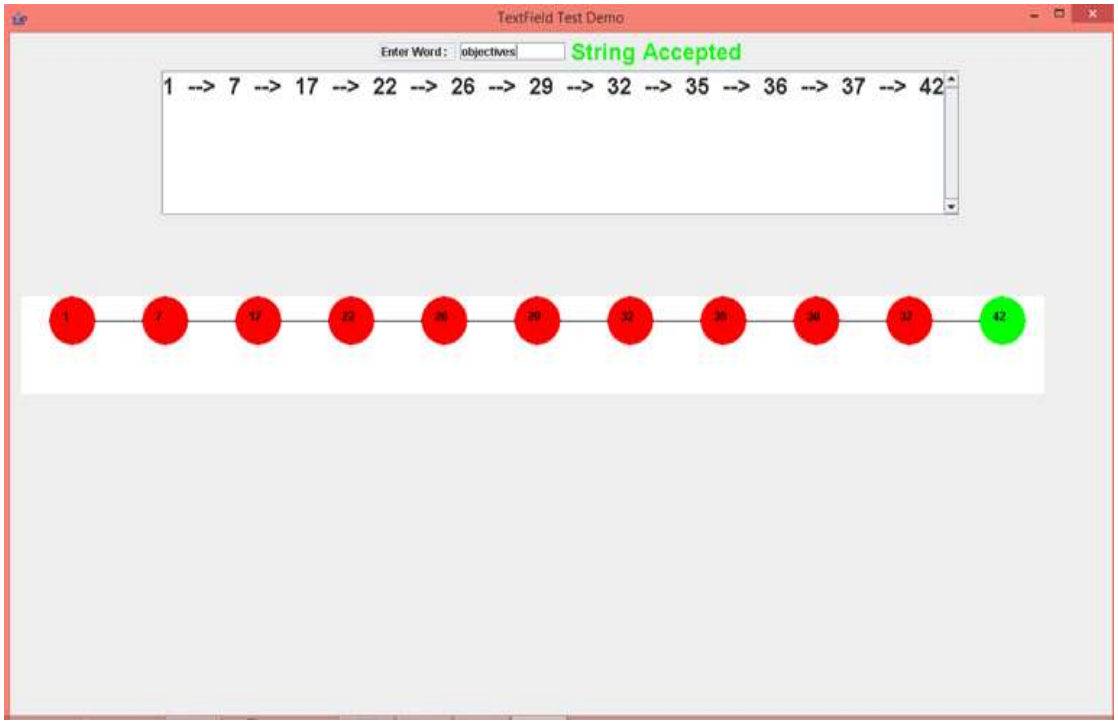


Figure 6. Class diagram for implementation of MQL token parser in Jav

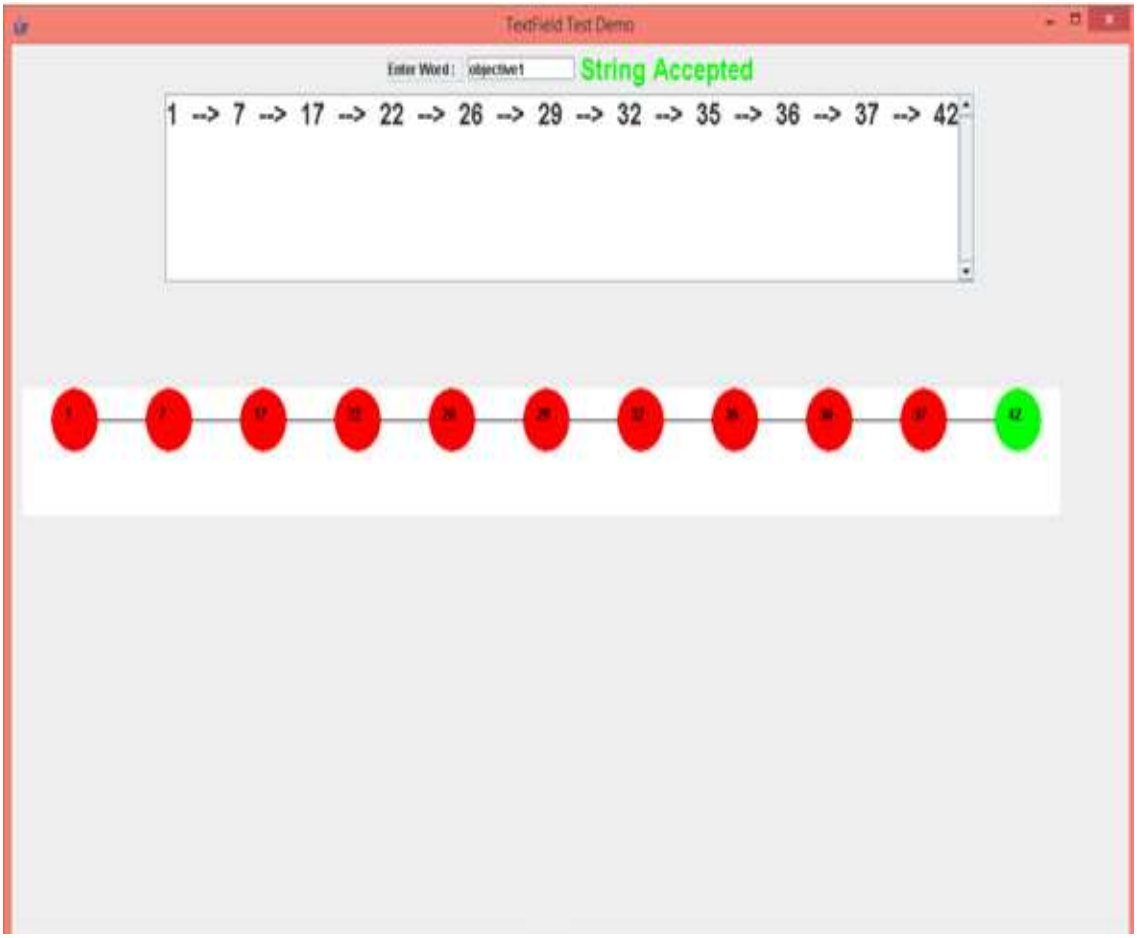


ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114





ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114



ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

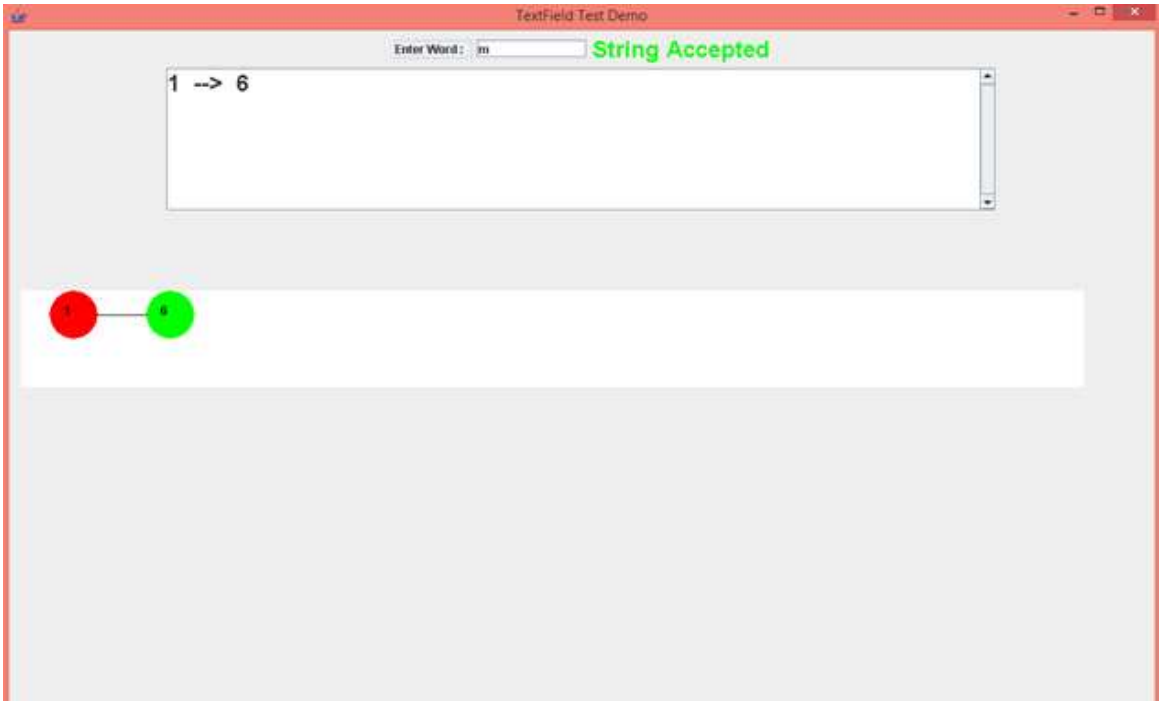


Figure 7a) – 7g) Parsing of MQL Tokens

CONCLUSION AND FUTURE WORK

In this paper we have presented a deterministic finite automata (DFA) parser developed by us for parsing MQL tokens. To implement MQL, we have designed our own language by defining a finite set of symbols, words and language rules, MQL grammar. The state table and state diagrams are developed for different tokens of MQL identified by us. State information is stored in a persistent database management system as a measure towards improving efficiency and extensibility. Currently, MQL consists of only few commands and more commands will be added to MQL in near future.

REFERENCES

1. Gideon Halevi, Handbook of Production Management Methods, Butterworth Heinemann publications, ISBN 0 7506 5088 5.
2. Girish R. Naik, V.A.Raikar and Poornima G. Naik, Multi Objective Criteria for Selection of Manufacturing Method, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 7, July 2014. ISSN:2277 128X.
3. Girish R. Naik, V.A.Raikar and Poornima G. Naik, Multi Objective Criteria for Selection of Manufacturing Method using NLP Parser, International Journal on Recent and Innovation Trends in Computing and Communication, Vol 2, Issue 11, Nov 2014, p.no 3484 – 3493, ISSN: 2321-8169.
4. 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.
5. 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.
6. 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-

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

erational Research, Vol. 137, No. 1, pp. 34–144, 2002.
7. 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.
8. 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.
9. R. V. Rao, "Decision making in the manufacturing envi-ronment using graph theory and fuzzy multiple attribute decision making methods," Springer-Verlag, London, 2007.
10. 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.
11. Dhananjay R. Kalbandeand G.T.Thampi, Multi-attribute and Multi-criteria Decision MakingModel for technology selection using fuzzy logic, International Journal of Computing Science and Communication Technologies, VOL. 2, NO. 1, July 2009. (ISSN 0974-3375)
12. Journal of Micromechanics and Microengineering, Xuan F Zha and H Du, Manufacturing process and materialselection in concurrent collaborativedesign of MEMS devices, 13, 509–522, 2003.
13. Chenhui Shaoa, , Kamran Paynabarb, Tae HyungKima, Jionghua (Judy) Jinc, S. Jack Hua, J. Patrick Spicerd, HuiWangd, 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
14. 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
15. Mohammad Akhshabi, A New Fuzzy Multi Criteria Model for Maintenance Policy, Middle-East Journal of Scientific Research 10 (1): 33-38, 2011