Web shop user error detection based on rule based expert system

M.Špundak
Vipnet usluge d.o.o.
Vrtni put 1, 10000 Zagreb, Croatia
Phone: (385) 01-4692023 Fax: (385) 01-4691259 E-mail: m.spundak@vipnet.hr

N.Bogunović and K.Fertalj
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
E-mail: nikola.bogunovic@fer.hr, kresimir.fertalj@fer.hr

Abstract - Rule based systems (RBS) have been recognized as probably the best solution for knowledge based expert systems. This article tries to provide the overview of the architecture and basic characteristics of the RBS, focusing on both their weaknesses and strengths. Based on a theory, rule based expert system for web shop error detection has been proposed. The RBS builds on the available application knowledge base and focuses on a problem of detecting the possible error in the shortest possible timeframe. The formalization of the whole process has a potential to significantly reduce the time required to detect the possible error.

I. INTRODUCTION

Rule based systems (RBS) that describe world by conditions and consequences have become popular during 1980's. Since then, they have imposed themselves as the best solution for the problem of reliable and sustainable engineering paradigm [1]. Such systems are based on set of rules which describe what should be done or concluded in given situations (called IF-THEN rules), set of facts and control part of the application that interprets facts and executes rules based on these facts [2].

IF-THEN form of rules is comprised of two parts: the factual left side and action right side. If the left side conditions are fulfilled or facts are true, the rule can be executed. By firing the rule, the right side of the rule adds new facts or executes certain actions.

Furthermore, RBS can be extended with fuzzy logic.

Very often, expert systems are built based on the rules, representing a system that comprise expert knowledge for specific knowledge area and is used for solving complex problems [3].

II. RULE BASED SYSTEMS

There are two types of RBS, forward and backward chaining systems. The main differences are starting point and the final goal of the system. Forward chaining system starts from the initial facts and based on them fires the rules and adds new facts or executes actions. Backward chaining system starts from the hypothesis that needs to be proven and tries to find rules that will lead to proving the hypothesis [2].

Furthermore, forward chaining system is used in the situations when there are more facts than consequences (left side is prevailing) or when there is substantial amount of known facts [1].

It could be said that forward chaining system is led by facts while backward chaining system is led by goal [2].

RBS are very flexible technique for developing expert systems, which allows basic control as well as decision making. Adding new rules to the system or changing existing ones is mostly simple. It is also generally accepted thinking that RBS are easy to understand and to implement, until they are of reasonable size [4].

A. Rule based system architecture

As mentioned earlier, typical RBS will comprehend three parts [2, 3]: inference engine or interpreter (control part of the application), rule base or knowledge base and working memory (set of facts). Interpreter is comprised of pattern matcher that defines which rules can be fired, agenda that determines the order of rules and execution engine that fires the rules.

The expert system is typically extended with rule base editor (enables development and modification of rules), explanation editor (enables explanations for some actions) and user interface [3].

B. Backward chaining systems

The backward chaining systems are mainly used for proving the known hypothesis. When trying to prove the hypothesis, the right side of the rule represents a state or conclusion and not the action. Also, the rules do not add new facts to the fact base, as all the facts are known in this case.

The system tries to find a conclusion on the right side that match the final hypothesis. When such rule has been found, all the facts on the left side become new hypotheses to be proven. The procedure is repeated until starting facts appear as hypotheses, by which the starting hypothesis has been proved.

The potential problem that may appear is the situation when several rules can be used for proving specific fact. If a programming language with search function is not used, then one solution could be to use the agenda. In that case, every part of the agenda can be used as potential path in solution finding, in a way that every part is searched until the solution is found. Schedule of the agenda determines solution finding [2].

C. Forward chaining systems

With the forward chaining systems, the facts are in the working memory and not on the stack as it is the case with backward chaining systems. The rules represent possible actions which will be executed if the conditions for executing them have been satisfied. As already stated, actions can be related to executing actions or to adding or deleting facts from the working memory.

Interpreter, as a control part of the system, checks rule base in the working memory in cycles and separates rules with satisfied left side, which can be fired. The next step is, based on predefined principles, to select one and to fire it what will add new facts to the working memory and the cycle will start from the beginning. Cycle is repeated until there are no rules to be fired or until wanted end state is reached.

The order of firing the rules is very important because it can significantly change the flow of the conclusions and consequently the end result. That was the reason for developing many different conflict resolution strategies [1, 2]:

- The same rule can not be fired more than once on the same fact base.
- Rules should be fired on newer facts in working memory rather than on older ones which ensures conclusion flow.
- Rules that relate to more focused facts should be fired before the ones that relate to more general facts.
- First available rule: the first available rules should be fired, according to the given agenda.
- Randomness: rules are fired in a random order, which is an advantage only in applications where uncertainty is welcomed (e.g. games).
- The most determined rule: rule with the most conditions is fired.
- The least used rule: by this strategy usage of all rules is maximized.
- “The best rule”: every rule has a weighting factor.

D. Rule based system development and maintenance

But, together with mentioned strategies, it is very important how the rules are set up. It should be carefully defined, with clear determination, when they can be fired what helps to control and to understand the RBS and its possible future extension and maintenance.

The help is provided from the parts of the system which are specifically in charge of controlling firing the rules, together with additional grouping of the rules or grouping the system by states [2].

In general, RBS can be applied successfully to problems that represent well known principles, which are suitable for modelling through a set of rules, such as orders, instructions and similar. It is necessary to have in mind that fact especially in cases when it is tried to model knowledge that is hard to express through rules [5].

There are typically 4 steps for iterative development of a RBS in specific knowledge area, where the first three steps represent system modelling while the last step represent its execution, starting from the higher levels to dissemination on lower levels [5]:

1. Create and edit a task model. Task model defines application structure and dynamic and this step includes task description, information flow between tasks and control flow. Tasks are disseminated on simpler tasks until they could be solved with simple rule set, and each task on the lowest level is comprised of action part (left side) which activates procedure for task execution (right side).
2. Add control knowledge. Logical parts of the task which express conditions for their start and finish.
3. Add object-knowledge connection. Describes conditions and relations within application domain so tasks within knowledge area are connected.
4. Execute a task model.

By using previously described procedure, it is possible to develop a RBS, but it is also possible to use a more practical approach [3]:

1. Knowledge collection. It is necessary to collect knowledge for rule set development from experts, through questionnaires, interviews, books, etc.
2. Data structuring. The correct data needs to be selected and displayed in a structured way.
3. Writing rules.
4. Interface creation. Data can be obtained from databases or by input through user interface or through interface with other systems.
5. Testing. By testing, it is possible to check results of the specific part or the whole system and it should be automatic.

As already stated, RBS are very sensitive to changes. Even a different order of executing the rules could lead to different results.

It is known fact that majority of existing expert system are results of research conducted on academic institutions, so there was no need to think about future maintenance and changes because all changes were done by authors.

Nevertheless, increase in usage of expert systems, especially usage in complex problems, stressed the importance of the maintenance. To successfully maintain such system, it is necessary to understand two basic parts of the expert system, its design and knowledge base used for system creation.

One of the possible solutions is to group rules (knowledge base) and definitions and to formally specify data flow between those groups. Rules are grouped by their mutual influence.

Such methodology defines several steps in development of RBS, which are recommended for usage in the early phase of system development [6]:

1. Extract control variables
2. Divide rules in groups
3. Identify local facts (all rules that change the fact are within the same group) and non-local facts (the facts that are shared between groups)
4. Describe non-local facts
5. If possible, separate control rules
6. Other issues

System syntax should be adjusted in order to apply this methodology.

If the system is already in use, algorithm to divide rules in groups should be used, according to definition of a measure of distance between the rules (their relation). Algorithm is based on an assumption that facts between two rules could be connected in three distinct ways and defines weight for each way:

- Left side – right side (input- output) (1.0).
- Shared right side (outputs) (0.75).
- Shared left side (inputs) (0.5).

The measure of relation between the rules is the sum of relations between the shared facts among these rules. To avoid grouping around single rules, measure for relation when two rules do not share a fact is introduced with weight 0.25.

Downside of this algorithm is that it never checks if there are better solutions, while the positive side is that by adding additional rule it is possible to favour smaller groups [6].

E. Disadvantages of rule based systems

The main disadvantage of the RBS that is often stated is that majority of developed systems failed to prove its value in production [7]. It is connected with the fact that many programmers have problems with programming RBS [7].

From the software engineering perspective, the main disadvantages of the RBS are [7]:

1. Impossibility to maintain. It is often stated that non existing border line between development and maintenance of RBS is actually its advantage, but it is not realized in practical applications. Reality is that different persons are responsible for development and for maintenance and if clear border line does not exist, it is a problem. Furthermore, RBS do not provide enough control in complex applications.
2. Impossibility to test. Usually, testing tries to conclude which data is correct based on input data and application control flow what is extremely difficult for RBS.
3. Unreliability. If it is not possible to test application, it is not possible to determine existence or non-existence of errors what significantly reduce system reliability. So, in the best case, RBS can be used only for consultancy, but than it is questioned when such consultancy is really needed.

But, all these disadvantages of the RBS could be stated as disadvantages of software engineering in general.

Very frequent problem with the large production systems is the large amount of the working memory, which can scale significantly by keeping information about the facts and control information [10].

Development of such systems required move to secondary memory environment, where databases became natural solution to the problem. Therefore, search and rule execution control algorithms applied on working memory have to be adjusted and optimized for application in new situations and accent is mostly on control algorithms [10].

If the problem is approached in a structured way, basic disadvantages could be mostly controlled, so RBS represents one of the simplest and widely used techniques for development of expert systems [1, 4, 5].

F. Rete algorithm

Rete networks [8] are used only for forward chaining RBS and their advantage is in reducing complexity of the RBS from exponential to linear, so significantly speeding up entire system, but using greater amount of memory. So it is question which is more important, speed or memory usage.

Rete algorithm can be also applied for error correction in RBS, for which the time stamp is added because it is important to know when certain rule has been fired [8].

Rete networks are comprised out of three types of memory nodes [8]:

1. Alpha node – for every condition on a left side, one instance for facts.
2. Beta node – instance for 2 or more consistently satisfied conditions.
3. Production node – instance satisfied for all conditions.

Two alpha nodes which represent rules conditions are connected into beta node which is connected with another alpha node into new beta node and so on for all conditions on the left side of the rule. At the end, everything is connected into production node.

Basic rules that should be applied for effectively usage of Rete algorithm are [9]:

1. The most specified samples in the rule should be placed at the beginning of the rule, but control variables should precede.
2. Samples with fewer facts should be placed at the beginning to reduce number of partial matches.
3. Samples that often vary (they are often placed and removed from the working memory) should be placed at the end of the sample list.

III. WEB SHOP APPLICATION

By using previously described rules and methods, it is possible to start collecting and structuring needed knowledge in order to write the rules with the defined conflict resolution strategies.

We will use CLIPS, a tool for expert system creation, as a help for rule creation and as a syntax reference. CLIPS fully implements RBS architecture and has a modified syntax based on Lisp [9]. Even though CLIPS implements advanced functionalities [9], only basic functionalities will be used for creating rules and facts. Final results are also tested through simulation in CLIPS.

A. Knowledge collection and structuring

The purpose of the web application is to enable web shopping. The shopping process has been defined with sequential diagram of the user process through all the systems of the web application. During the user interaction, errors may appear, on specific points in the user process. When error appears, user will receive specific information with description and instructions what to do.

The main problem is when the user wants to contact application provider with question or complaint. In that case, it is needed to determine the reason why error actually appeared in order to successfully solve the problem.

Someone could notice that it is possible to try to solve the problem sequentially considering the limited number of possible causes, but the question is is it possible to optimize the order of the questions to reduce the time from the initial question until the problem solution to a minimum. Such requirement is the consequence of the limited resources available for solving the user questions and complaints.

The start of creation of the system for finding the possible cause of the problem is the collection of the information that is displayed to the user. That information will be the ones the user will provide through the question, and are provided in the Table 1.

Table 1 Set of facts available to user

Fact	Description
unavailable-address	Browser displays error that web application address is unavailable
http-gateway-timeout	Browser displays error which is generated on the provider side.
page-is-not-available	The page tried to be displayed is unavailable.
general-error	General error of the web application
password-incorrect	Message for the incorrect password
username-unknown	Message for the incorrect username
mobile-number-incorrect	Message for the incorrect telephone number
payment-invalid-data	Message for the incorrect payment details
payment-missing-data	Message for the incomplete payment details
payment-verification-failed	Message for the unsuccessful payment verification
quantity-error	Message for the unavailable quantities of the products in the shopping basket
order-creation-error	Message for the unsuccessful order creation

Information can be obtained from the user with the set of YES/NO questions that will add correct facts to the set of facts. In that way, errors in data entry can be avoided.

Also, it could be mentioned that at the points where errors could appear, it is possible to have several reasons. Possile reasons are listed in the Table 2.

Table 2 Possible error reasons

Reason
User browser
Web application server is not available
Proxy server is not available
User has entered incorrect login information
Unsuccessful payment verification
User has entered incorrect payment information
Payment server is not available
User has insufficient amount for payment
Order creation system is not available
Network problem in web application
Requested amount is not available

B. Rule base

Rules have been structured in 2 groups. First group in Table 3 represent rules based on user interaction, i.e. rules based on which could be concluded what happened to the user. Information from the Table 1 is coded into variables based on user's answers.

Table 3 Rules for user interaction

```
;;**********************
;;* questions          *
;;**********************

(defrule initial-question ""
  ?for-remove <- (initial-fact)
  (not (shopping-state ?))
  =>
  (if (yes-or-no-p "Did you finish the shopping successfully
(yes/no)? ")
    then
          (assert (shopping-state ok))
          (retract ?for-remove)
    else (assert (shopping-state notok))))

(defrule error-1-a
  (shopping-state notok)
  (process login-not-ok)
  =>
  (if (yes-or-no-p "Did the browser message appear (yes/no)? ")
    then
      (assert (error unavailable-address))))

(defrule error-1-b
  ?for-remove <- (initial-fact)
  (process ?)
  (shopping-state notok)
  =>
  (if (yes-or-no-p "Did message '504 HTTP GATEWAY
TIMEOUT' appear (yes/no)? ")
    then
      (assert (error http-gateway-timeout))
           (retract ?for-remove)))

(defrule error-1-c
  ?for-remove <- (initial-fact)
  (process ?)
  (shopping-state notok)
  =>
  (if (yes-or-no-p "Did message application not available appear
(yes/no)? ")
    then
      (assert (error page-not-available))
           (retract ?for-remove)))

(defrule error-1-d
  ?for-remove <- (initial-fact)
  (process ?)
  (shopping-state notok)
  =>
  (if (yes-or-no-p "Did  'General error'
appear (yes/no)? ")
    then
      (assert (error general-error))
           (retract ?for-remove)))

(defrule error-2-a
  (process login-not-ok)
  (shopping-state notok)
  =>
  (if (yes-or-no-p "Did message 'Password not correct' appear
(yes/no)? ")
    then
```

```
        (assert (error password-incorrect))))

(defrule error-2-b
  (process login-not-ok)
  (shopping-state notok)
  =>
  (if (yes-or-no-p "Did message  'Username not recognized'
appear (yes/no)? ")
     then
       (assert (error username-unknown))))

(defrule error-2-c
  ?for-remove <- (initial-fact)
  (process ?)
  (shopping-state notok)
  =>
  (if (yes-or-no-p "Did message 'Telephone number not vaild'
appear (yes/no)? ")
     then
       (assert (error mobile-number-incorrect))
              (retract ?for-remove)))

(defrule error-4-a
  (process login-ok)
  (process payment-not-ok)
  (shopping-state notok)
  =>
  (if (yes-or-no-p "Did message about wrong payment data
appear (yes/no)? ")
     then
       (assert (error payment-invalid-data))))

(defrule error-4-b
  (process login-ok)
  (process payment-not-ok)
  (shopping-state notok)
  =>
  (if (yes-or-no-p "Did message about missing payment data
appear (yes/no)? ")
     then
       (assert (error payment-missing-data))))

(defrule error-5
  (not (solution ok))
  (process payment-ok|payment-not-ok)
  (process login-ok)
  (shopping-state notok)
  =>
  (if (yes-or-no-p "Did message about insuficient quantity appear
(yes/no)? ")
     then
       (assert (error quantity-error))))

(defrule error-6
  (process login-ok)
  (process payment-not-ok)
  (shopping-state notok)
  =>
  (if (yes-or-no-p "Did message about unsucceful payment appear
(yes/no)? ")
     then
       (assert (error payment-verification-failed))))

(defrule error-7
  (not (solution ok))
  (process payment-ok|payment-not-ok)
  (process login-ok)
  (shopping-state notok)
  =>
  (if (yes-or-no-p "Did order creation error appear (yes/no)? ")
     then
       (assert (error order-creation-error))))


(defrule error-login
  ?for-remove <- (initial-fact)
  (shopping-state notok)
  =>
  (retract ?for-remove)
  (if (yes-or-no-p "Did you manage to succesfully login (yes/no)?
")
     then
       (assert (process login-ok))
            else
              (assert (process login-not-ok))))

(defrule error-payment
  (process login-ok)
  (shopping-state notok)
  =>
  (if (yes-or-no-p "Did you manage to complete payment
(yes/no)? ")
     then
       (assert (process payment-ok))
            else
              (assert (process payment-not-ok))))
```

The second part of the rules shown in Table 4 is comprised of possible solutions, according to Table 2. Solutions could also be an integral part of the rules in the first part, together with the questions, but for easier maintenace possible solutions were separated from the questions. This opens possibility for changing procedure in case of specific error, which is often the case in the process lifecycle.

Table 4 Rules for asserting solution

```
;;**********************
;;* solutions              *
;;**********************

(defrule solution-1
  (error unavailable-address)
  =>
  (assert (solution ok))
  (printout t "User browser is not set up" crlf))

(defrule solution-2
  (error http-gateway-timeout|general-error|page-not-available)
  =>
  (assert (solution ok))
  (printout t "Web application server in not available (internal
error)" crlf))

(defrule solution-3
  (error http-gateway-timeout)
  =>
  (assert (solution ok))
  (printout t "Proxy server is not available" crlf))
```

```
(defrule solution-4
  (process login-not-ok)
  (error password-incorrect|username-unknown|mobile-number-
incorrect)
  =>
  (printout t "Wrong user information at login" crlf))

(defrule solution-5
  (error http-gateway-timeout)
  =>
  (printout t "Unsuccesful payment verification " crlf))

(defrule solution-6
  (error payment-invalid-data|payment-missing-data)
  =>
  (printout t "User entered wrong payment information" crlf))

(defrule solution-7
  (process payment-not-ok)
  (error payment-verification-failed)
  (error general-error)
  =>
  (printout t "Payment server is not available" crlf))

(defrule solution-8
  (process payment-not-ok)
  (error payment-verification-failed)
  =>
  (printout t "User does not have enough amount for payment"
crlf))

(defrule solution-9
  (error error order-creation-error)
  =>
  (printout t "Order creation system is not available" crlf))

(defrule solution-10
  (error general-error)
  =>
  (assert (solution ok))
  (printout t "Web application network error" crlf))

(defrule solution-11
  (error quantity-error)
  =>
  (printout t "Requested amount is not available" crlf))
```

IV. CONCLUSION

The usage of the rule based expert system in error detection within the web shop application has been thought as a natural choice. The form of asking questions and concluding based on the answers is the usual form of expert communication with the end user, especially in a remote form of the communication.

Formalizing this process in a rule based expert system has a potential to ease this communication in several ways. It significantly reduces the time needed to detect error by focusing on a predefined order of questions. It also reduces the need for experts by gathering expert knowledge in a system.

To fully exploit the advantages of this application, further work could focus on implementation of rule based expert system as an end user application which could provide guidance to the end user without human interaction at the application provider side.

REFERENCES

[1] AI Depot. [Online]. Available: http://ai-depot.com/

[2] A. Cawsey, “Rule-Based Systems”. [Online]. Available: http://www.zemris.fer.hr/predmeti/krep/Rules.pdf

[3] E. Friedman-Hill, *Jess in Action: Java Rule-Based Systems*. Greenwich, CT, USA: Manning Publications Co., 2003

[4] The Rule-Based System, AIGameDev.com. [Online]. Available: http://aigamedev.com/

[5] M. Vestli, I. Nordbo, A. Solvberg, “Modelling control in rule-based systems”, *IEEE Software*, vol. 11, pp. 77-81, Mar 1994

[6] R.J.K. Jacob, J.N. Froscher, “A Software Methodology for Rule-Based Systems”, *IEEE Transactions on Knowledge and Data Engineering*, vol. 2, pp. 173-189, Jun 1990

[7] Xiaofeng Li, “What's so bad about rule-based programming?”, *IEEE Software*, vol. 8, pp. 103-105, Sep 1991

[8] S.M.Tuttle, C.F. Eick, “Historical Rete Networks for Debugging Rule Based Systems,” in *Proceedings of the 1991 IEEE International Conference on Tools for AI*, IEEE, 1991, pp. 450-457

[9] J.C. Giarratano, *Clips User's Guide*, Version 6.10, 1998

[10] J. Tan, J. Srisvastava, “Efficient Rule Matching in Large Scale Rule Based Systems,” in *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, IEEE, 1992, pp. 391-402