Enhanced Design of a Rule Based Engine Implemented using Structured Query Language

Mohammad J. Sawar, Umair Abdullah, Aftab Ahmed

***Abstract*—Rule Based Systems belong to a well established branch of Artificial Intelligence. So far thousands of rule based systems and their related systems have been built and successfully used. Recently a Rule Based Engine has been successfully designed and developed using Structured Query Language, and applied in Medical Claim processing domain. The rule engine has been integrated with medical billing software to identify billing errors in medical claims at real-time. Performance of the engine has been good, giving promising results. To further improve the efficiency of the system and to utilize power of rule based systems' techniques, enhancements in the existing rule based engine are being proposed in this research paper. Besides explaining the design of new rule based engine, this paper also reviews the design of current engine, which is already in operation, and the overall architecture of the whole system. Enhanced rule engine being proposed here can be implemented in any domain which involves large number of knowledge oriented checks, and in which frequent modification or updating of these checks is required. This research work proposes a new frame work of rule based systems related to relational database environment (i.e. Structured Query Language), and can have great impact on business world where large amount of data is stored in relational format.**

***Index Terms*—Artificial Intelligence and its Applications, Knowledge Engineering, Rule Based Systems.**

I. Introduction

Incorporation of production rule systems in relational database is not a new idea; it is regularly used in active databases [11]. Similarly deductive and inductive databases also involve rules for learning purpose [4], [5]. However, one feature is common in all of these studies i.e. they focus on programming of rules at database server level. Similarly research conducted by Hanson and Widon [2] gives us an overview of production rule systems developed in relational database environment, but again it describes "programming of production rules" at database server level with production rule programming language. In order to be flexible, efficient, and robust a production rule should not be programmed rather it should be inserted like a record. In other words rules should be part of data not part of program or source code. That is why no success stories have been revealed so far about the implementation of a rule based system at application layer level without the need of rule programming. Such systems have not found widespread adoption outside of academia. In active databases production rules are like triggers with some additional conditions and action parts [2], [11]. One cannot realize the need of an advance form of database trigger to be called as production rule in database. To understand the concept of a rule based system we need to explain the difference between an "if" statement in a programming language, a "trigger" in a database and a "rule" in a production rule system.

At conceptual level all of these are the same i.e. having a condition portion and a then-part which is executed when condition part returns true. One possible explanation is that an "if" statement of a programming language and a "trigger" in a database are part of code, while a "rule" is part of data, separate from code. Changing a portion of code requires programming skills and a development environment. While changing data does not require programming skills and development environment, rather it needs some editor or interface for editing of rules (rule editor, knowledge editor). This separation of knowledge from code gives us fast flexible and powerful ways for implementation of knowledge oriented checks without the need of programming skills i.e. a domain user can develop rules by himself/herself by using a rule editor.

Conceptual relationship between a rule and a SQL query is well-known, and well-understood (i.e. a rule can be considered as "where" clause of a SQL query). But relationship between a Rule Inference Engine and SQL is neither well-known, nor well-understood.

Artificial Intelligence systems, which originally started with machine learning techniques, such as inductive learning and explanation based learning [6], [7], [13], [15], are now moving towards integration of data mining modules to take further advantage of huge volumes of business data stored in almost every organization. Simple architecture of classical expert system with and without a data mining module has been

Manuscript received March 18, 2010. This work has been supported by Higher Education Commission of Pakistan under "5000 – indigenous PhD Fellowships Scheme".

Mohammad J. Sawar is Dean of Faculty of Engineering & IT, Northern University, Nowshera, Pakistan. (phone: +923218507321; e-mail: sawar@northern.edu.pk). He is also acting as Director at Barani Institute of Information Technology, Rawalpindi, Pakistan. (e-mail: sawar@biit.edu.pk)

Aftab Ahmed is Dean and Director at Foundation University of Engineering and Management Sciences, Foundation University Islamabad, Pakistan. (e-mail: aftab_ff@hotmail.com).

Umair Abdullah is PhD scholar at FUIEMS, Foundation University, Islamabad. Pakistan. (e-mail: umair_pitafi@yahoo.com).

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

described by Nada Lavra'c [10]. Many researchers around the world are working on integration of data mining and AI systems (rule based, intelligent, knowledge based, expert etc.) Main hypothesis behind all these research activities is that an AI system fed with knowledge from domain experts, using data mining plus machine learning techniques can acquire refined and large volumes of knowledge as compared to AI systems which use domain experts and machine learning only [1]. System developed by Ortega [12] is an example of intelligent system which uses data mining techniques, thus following the hypothesis stated above.

Medical billing is a process employed by healthcare providers (such as Surgeons, Physicians, Practicing Nurses, etc.) or their billing companies, to submit and follow-up medical claims [17]. The medical claims are generally submitted to Insurance Companies for reimbursement of services rendered by the providers. Claim itself is a complex beast, composed of codes for procedures (treatments/tests), diagnoses (diseases) and their relevant modifiers along with patient data. The relevance of procedures to diagnosis is very critical; similarly various modifiers change the overall meaning of a diagnosis code, etc. Generally medical practices employ experienced coders to achieve their billing goals.

When the billing is carried out by the healthcare providers locally it is termed as in-house billing, which invariably tends to be a very tedious task. To overcome this problem, the providers generally sign up with medical billing companies which are geared up for bulk claim submissions and follow-ups. Follow-up normally occurs when a claim is rejected due to inconsistency in patient data, services rendered or improper relation between the diagnoses and the procedures.

There are a number of medical billing companies providing claim submission and follow-up services plus other IT based healthcare solutions (such as Electronic Medical Records, Automated Patient Calling for data verification, etc.). One major challenge for these billing companies is to handle claim rejection issue and to minimize the costs related to claim follow-ups. Some companies have developed claim scrubbing software, which allows them to check the claim for inconsistencies in patient data and improper use of procedures for specific diagnoses, before sending the claim forward [8], [9], [14], [16], [17]. Claim scrubbing reduces the claim turn-over time and at the same time increases the chance of claim acceptance.

This paper presents the research and development work being carried out at a medical billing company. A rule based system was developed using Structured Query Language [16]. The rule engine has been moved to production by integrating it in medical billing software. Now rules are being developed and added to the system by knowledge engineers (operations staff) of the company. These rules perform medical billing related data consistency checks and modify claims where required.

Second section of this paper briefly outlines the overall architecture of the whole rule based system [1]. Simplified version of rule based engine design taken from [16] has been presented in the third section. Fourth section describes the characteristics of current Rule Based Engine (RBE), highlighting strong and weak points. Section V is the detailed explanation of proposed enhanced rule engine design being implemented using Structured Query Language. Last section contains the final conclusion along with future research directions.

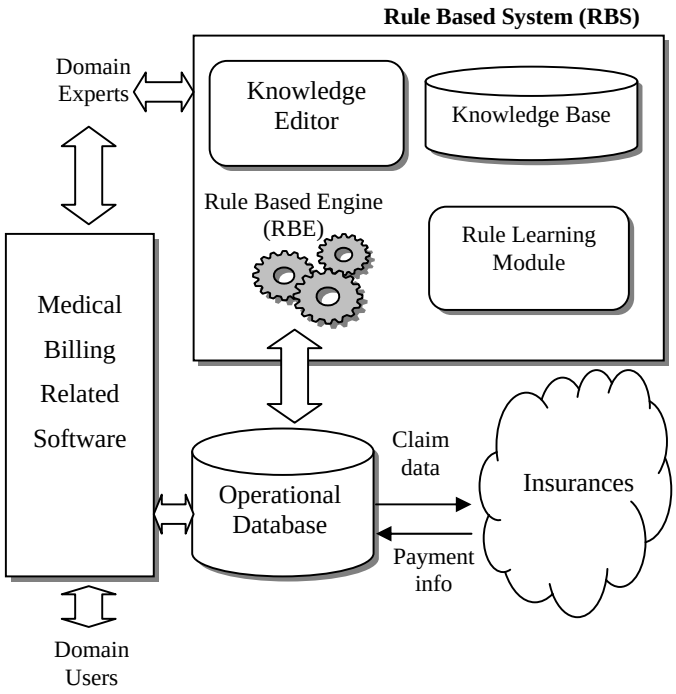

Figure 1. Simplified Architechture of Rule Based System for Medical Billing.

II. Over all architechture of Rule Based system

Architecture of a data mining driven learning apprentice system for medical billing is given in [1]. A simplified version of Rule Based System (RBS) is shown Fig. 1. Billing software is the main entity with which most of the users interact. Billing executives use it for inserting, updating, modifying claims' data. Domain experts use it for pulling various types of reports, importing data and following up claims. Billing software triggers Rule Based Engine (RBE) to perform billing compliance related checks on a claim saved at that time. Thus RBE applies its production rules on the claim (which is being saved) and informs the user about faults found in the claim. RBE also performs scrubbing activity i.e. modify record(s), which is defined as "then" part of the production rules. Main database contains claim related data, input from billing software, web sites, HL7 (i.e. Health Level 7) files, data imported from database of new practices and data sent by synchronization server.

Electronic Data Interchange (EDI) module gets data from the database and converts it into text files in specific format (like 837 format) which is acceptable by the insurance companies. Submission module sends these files to the insurances via internet as electronic submission or paper submission. This process is outside the domain of our learning apprentice system (represented by the box). The team which is developing and managing the RBS communicates with domain experts (operations staff) through email for knowledge engineering process. Domain experts use knowledge editor to add new knowledge (as production rules) to the knowledge

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

base or verify any new rules discovered by production rule mining module.

III. Design of existing Rule Based Engine

Design of Rule Based Engine (RBE) has been discussed in detail in [16]. A simplified version of existing RBE design is shown in Fig. 2. It has been implemented in structured query language for claim scrubbing process. Main components of RBE are "meta rules", "rules" and "logical variables", stored in the database in their respective tables.

First phase (represented by dotted line box in Fig. 2) is the selection of applicable rules. In the initial stage only those rules are selected which have priority 25 or 75. Then all the Meta rules are executed one by one. When a Meta rule returns true then its related rules are selected. At the end of this phase all applicable rules have been selected (either due to priority or due to true condition of their Meta rule).

In the second phase, rule engine applies all these selected rules individually on a given claim and identifies the inconsistencies and errors. Each rule is like a check with some action part associated to it, implemented as a "where" clause of a SQL query [16]. The "select" portion of the rule query is attached automatically within RBE. For example the check of "missing date of birth" will be implemented as **"***where '<DOB>' = ''* **"**. The token <DOB> is a logical variable and will be replaced by its value. RBE will get its value by executing the query stored in Logical variables table. For example, if <DOB> is blank then rule query after replacing the value of logical variable will become:

select @rowcnt = count(*) where '' = ''

Now "where" clause is true so @rowcnt SQL variable will get value 1 indicating the error of "patient date of birth missing". Suppose date of birth of the patient is not missing i.e. it is "08/20/2009" then rule query after replacing value of logical variable <DOB> will become:

select @rowcnt = count(*) where '08/20/2009' = ''

In this case @rowcnt SQL variable will get 0 value (as condition is false in "where" clause) thus indicating that error of "patient date of birth missing" has not occurred.

The claim scrubbing behavior of RBE is briefly demonstrated above, where it can identify the inconsistencies in the patient data (such as the missing or invalid information). Each production rule in the RBE's knowledge base has its own scrubbing behavior. The critical rules in general stop RBE from submitting a faulty claim, while non critical rules simply generate a warning message [17]. Some rules exclude faulty procedure code instead of blocking the whole claim. Furthermore some rules apply modifiers with the procedure code, but these are according to the instructions given by providers. Rule application log is maintained by the system. This log is used for pulling "daily claim error report" and "RBS blocked claim reports" and any other analysis related to the performance of RBS [16].

IV. Characteristics of Existing Rule Based Engine

This section describes characteristics of existing rule based engine, which is currently in production. Current RBE has some advantages. First of all it is implemented in structured query language (SQL), in the form of stored procedures and functions. It eliminates the need of transformation of data from operational environment (i.e. relational database) to some knowledge base environment. Meta rules have been used to find out potential errors without executing/ evaluating the whole condition. Meta rules increase the efficiency of the system. Concept of "must try" priorities (which are 25 and 75 for lower and higher must try rules respectively) has been used. These numbers are intuitive, assuming the priority of a rule between 1 to 100. Rules which must be tried should be defined with priority 25 or 75. Rules with priorities between these two, will be tried after the execution of priority 75 and before trying priority 25 rules.

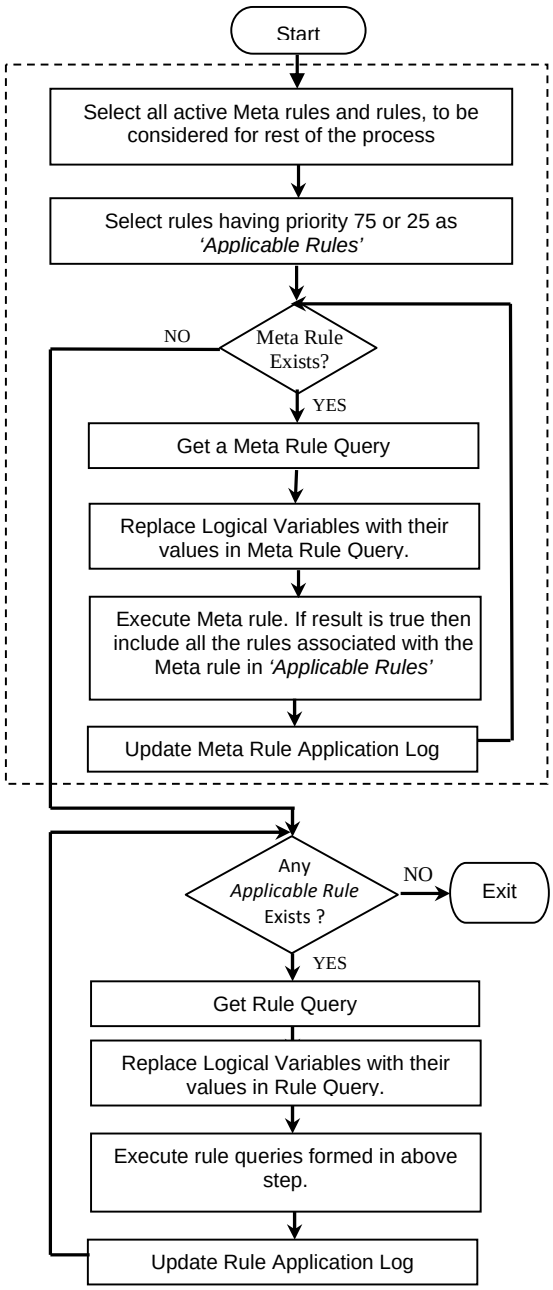

Figure 2. Simplified Design of Current Rule Based Engine.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Current design has following weaknesses causing efficiency problems when number of rules and claims are in thousands.

1. No separate working memory. Whole production database serves the purpose of working memory.
2. Rules are not interacting with each other and work independently.
3. Original tables of operation database are accessed again and again for getting values of logical variables. If a logical variable is used again in another rule, its value will be acquired again from original operational tables.
4. There is only one level of hierarchy below meta rules. A rule cannot associate with more than one meta rules, causing bottlenecks in various situations.
5. Rule engine performs linear one pass evaluation of RBS rules, that is rules are selected and tried. If condition of a rule is true then its then-part is executed. While traditional rule engines perform multiple passes of rules and continues to try rules till some specific condition is met or exists in working memory.
6. Although priority numbers have been associated with every rule but it is of no use when only priorities 25 and 75 are currently being used.
7. When a critical rule identifies some error, it simply blocks the claim i.e. it no longer remains "being billed" to insurance. This stops remaining rules from executing on the given claim. Since all the rules should be applied/tried when claim is being billed. This limitation stops RBE from identifying all the errors present in the claim.
8. Mutually exclusive rules are not catered for. If a rule returns true other rules which are mutually exclusive are also tried, understandably consuming more CPU time without generating any results.
9. Works in batch format, where the claims from multiple practices and users are processed sequentially, thus reducing the overall efficiency of RBE.

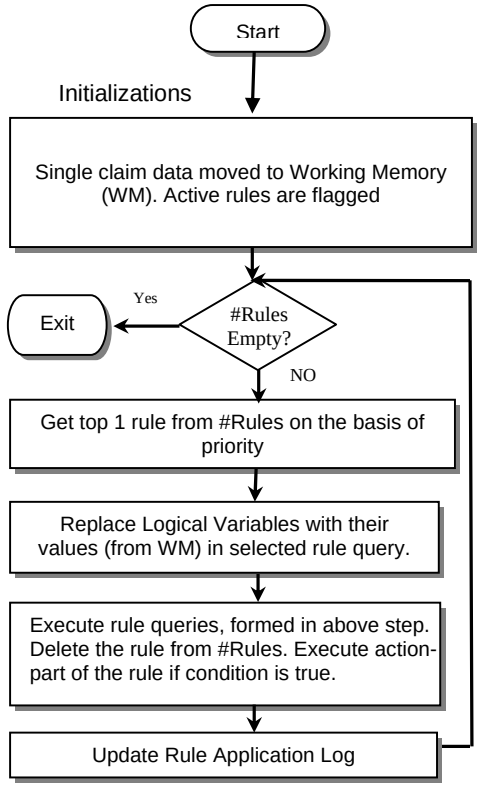

Figure 3. Proposed Enhanced Design of Rule Engine.

Improved RBE design described in the following section is also being implemented using Structured Query Language (SQL). It is expected to overcome all the weaknesses mentioned above, and will be more thorough and efficient as compared to existing engine.

V. Proposed Enhanced Design Of Rule Based Engine

Proposed enhanced design is according to the true nature of Rule Based Systems [3]. Proposed design of enhanced engine has been shown in Fig. 3. It is simple yet expected to be more powerful and efficient than the existing engine design. Starting point is same i.e. like existing engine, enhanced engine will also take one claim at a time as input and process that claim. It will check the claim for potential errors and carry out modifications where possible.

First of all before starting the main loop, RBE fetches all the data which is related to the given claim into working memory. As the design is proposed to be developed in SQL, the working memory is represented by temporary tables. When fetching the values of logical variables, engine will not access original data tables rather it will use these temporary tables (working memory).

Second major modification proposed is the selection of rules for applying on the claim. Instead of using must execute priority and meta rules, in enhanced RBE, rules will be activated and deactivated by other rules, starting from "rStart" which is the designated starting rule of the engine. It will activate other rules, which in turn can activate more rules.

A temporary table #Rules will be used for activating and deactivating rules. A rule is activated (by other rule) by simply adding its name to #Rules table and a rule is deactivated (by some other rule) by simply deleting its name from #Rules temporary table.

This kind of mechanism will allow multi-level rule hierarchy i.e. one condition will be checked in one rule and if it is true remaining condition will be checked in the following rule (which will be activated by this rule) and so on.

Furthermore main loop of rule engine will be multi-pass. It will continue trying rules till the #Rules table is empty. Note that a rule will be automatically deleted from #Rules table immediately after its execution. But it can be re-activated again by some other rule. Similarly mutually exclusive rules will also be handled in the same fashion.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Third proposed major change is the defining of #Variable_values temporary table for storing values of logical variables. This will allow us to overcome problem of showing only one error at one time. It will also reduce RBE effort of fetching value(s) of the same logical variable again and again. Result of a rule will also be stored in #Variable_values temporary tables, so that it could be used in other rules thus allowing multi-level rule hierarchy and step by step checking of faulty conditions or data inconstancies.

A. Working Memory (WM)

The expected difference in problem solving power and efficiency of existing RBE and enhanced proposed RBE design is due to the introduction of #Rules, #variable_values and claim related temporary tables. These tables serve the purpose of working memory of the rule engine. These tables will be populated when the RBE is activated on a single claim and will be flushed at the end. Working memory (temporary tables) will be re-filled again when RBE is applied on other claims.

B. Concurrency

Contrary to the existing RBE, the proposed RBE will operate concurrently, where many users will be able to process their claims simultaneously. Each user will have its own set of temporary tables (working memory), for scrubbing the claim.

VI. Conclusion

Proposed enhanced design is according to the true nature of Rule Based Systems. Existing engine is linear with only two levels of hierarchy i.e. Meta rules and rules. Only Meta rules can activate normal rules. While in proposed enhanced RBE design normal rule can activate and also deactivate one or more rules, thus allowing multi-levels of rule hierarchy.

The concepts of working memory and concurrency in Rule Based System have been handled very elegantly by using temporary tables, which allow multiple users to execute multiple instances of the RBR in the same SQL environment.

Although both (existing and proposed) engine designs are being used in medical billing domain, but these engines can be applied (without modification of rule engine SQL code) to virtually any real life domain which requires extensive application of knowledge oriented data consistency checks.

Acknowledgment

Many thanks to USA based medical billing company for providing excellent research environment. Name of the company has not been disclosed due to the "Cooperate Identity Disclosure Policy" of the company.

References

[1] A. Ahmed, U. Abdullah, M. J. Sawar, *"Software Architecture of a Learning Apprentice System in Medical Billing"* The 2010 International Conference of Computational Intelligence and Intelligent Systems, at World Congress on Engineering, London, U.K., 30 June - 2 July 2010. to be published.

[2] E. N. Hanson, J. Widom, *"An overview of production rules in database systems."* Knowl. Eng. Rev. 8, 2, 121-143. 1993.

[3] F. Hayes-Roth, *"Rule-based systems,"* Communications of the ACM, vol. 28, pp. 921-932, September 1985.

[4] Herve Gallaire , Jack Minker , Jean-Marie Nicolas, *"Logic and Databases: A Deductive Approach,"* ACM Computing Surveys (CSUR), v.16 n.2, p.153-185, June 1984.

[5] Luc De Raedt, *"A perspective on inductive databases,"* ACM SIGKDD Explorations Newsletter, v.4 n.2, p.69-77, December 2002.

[6] M. J. Sawar and R. C. Thomas *"Learning Apprentice System for Turbine Modeling"* in Proceedings of IEA/AIE-90, Charleston, U.S.A. July 1990.

[7] M. J. Sawar, T. G. Brennan, A. J. Cole and J. Stewart *"An Expert System for PostOperative Care (POEMS)"*, Proceedings of MEDINFO-92, Geneva, Switzerland, Sep. 1992.

[8] Mays II, Carl *"Don't Leave Money on the Table - Use a Claim Scrubber."* 30 Sep. 2008.

[9] "MediSoft The Total Revenue Cycle Solution For Physicians" SLC Software Services, www.slcsoftware.com

[10] N. Lavra'c, *"Data Mining and Decision Support: A note on the issues of their integration and their relation to Expert Systems"* PKDD'01 workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning: Positions, Developments and Future Directions, p 1-8. 2001.

[11] Norman W. Paton , Oscar Díaz, *"Active database systems,"* ACM Computing Surveys (CSUR), v.31 n.1, p.63-103, March 1999.

[12] P. A. Ortega, C. J. Figueroa, G. A. Ruz, *"A medical claim fraud/abuse detection system based on data mining: A case study in Chile"* , DMIN'06, The 2006 International Conference on Data Mining, Las Vegas, Nevada, USA, June 26-29, 2006.

[13] Pat Langley and Herbert A. Simon, *"Applications of Machine Learning and Rule Induction,"* Communications of the ACM November 1995, Vol. 38, No. 11. Pages 55-64.

[14] T. Hazen, *"Scrubbing Reimbursement Rates Clean"* Alpha II Claim Staker solution (online) www.alphaII.com

[15] T. M. Mitchell, Sridhar Mahadevan, and Louis I. Steinberg. *"LEAP : A Learning Apprentice for VLSI Design,"* in Proceedings of Ninth IJCAI. pp. 573-580. 1985.

[16] U. Abdullah, M. J. Sawar, A. Ahmed, *"Design of a rule based system using Structured Query Language"*, in Proceedings of 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC09), Chengdu, China. pp 223-228, 2009.

[17] U. Abdullah, M. J. Sawar, A. Ahmed, *"Comparative Study of Medical Claim Scrubber And A Rule Based System"* in Proceedings of IEEE 2009 International Conference on Information Engineering and Computer Science (ICIECS 09), Wuhan, China. pp 1-4, 2009.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)