Modelling a Rule Based System for Medical Underwriting in an Insurance Industry

Arnold Lephoto and Okuthe P. Kogeda

***Abstract* — insurance companies rely on medical underwriting to assess the risk of prospective clients. However, insurance companies often experience slow responses, inefficiencies and unreliable decisions from the current underwriting systems. In this paper, we present the architecture and implementation of a rule based Medical Underwriting System (MUP). The proposed architecture is multi-tier based system using Rete algorithm to implement the medical underwriting rule engine. The specific design and implementation is done using Java platform. The results show that MUP outperformed the current underwriting systems. The MUP memory usage is minimal and stable. The MUP obtained a faster response time. Errors generated by MUP were fewer than the current underwriting systems, with a ratio of 3:9.**

***Index Terms*— Insurance industry, medical underwriting, rule-based systems, Multi-tier architecture, and Rete algorithm.**

I. Introduction

Current ways of performing medical underwriting are highly ineffective, very slow and error-prone. This leads to most companies losing clients to competitors, and a lot of disputes arise during the insurance claims process. It is vital for insurance companies to achieve 99% accuracy in medical underwriting. This in turn helps in managing the allocation of appropriate life cover to customers with various medical conditions.

In this paper, we propose Rete algorithm for effective execution of medical underwriting business rules. The proposed system is based on Java Enterprise Edition (JEE) platform, which is aimed at implementing the proposed Medical Underwriting Prototype (MUP) architecture. The proposed MUP architecture is multi-tier based system. MySQL server was used to store and retrieve client's medical information. We used Drools expert system and rete algorithm to implement our system [22]. Advanced testing tools such as SoapUI [25], JMeter[26] and JUnit[27] were used to test and evaluate the proposed system.

The remainder of this paper is organized as follows: In Section II, we present an overview of medical underwriting and life insurance. In Section III, we present an overview of rule-based systems. In Section IV, we provide various techniques for medical underwriting and review previous work relating to this study. In Section V, we present system requirements and the proposed system architecture. In Section VI, we present the system implementation. In Section VII, we present testing, evaluation and discuss results. We conclude in Section VIII.

Manuscript received June 30, 2014; revised August 10, 2014. This work was supported in part by Tshwane University of Technology grant. Arnold Lephoto is with Tshwane University of Technology, department of Computer Science, Private Bag X680, Pretoria 0001, South Africa (arnoldlephoto@gmail.com).

Okuthe P. Kogeda, is with Department of Computer Science, ICT faculty, Tshwane University of Technology, Private Bag X680, Pretoria 0001, South Africa (kogedapo@tut.ac.za).

II. Overview of Medical Underwriting

A. Definition

For a long time, insurance companies used medical underwriting as a risk assessment and management tool. Medical underwriting has always been seen as a critical part of the insurance business that affects the future of the company and its bottom line.

Wang [4] defines medical underwriting as "the insurers' effort to distinguish among different risk types and to assign premium rates that reflect differences in risk levels among insurance applicants". A client with high medical risk pays more for insurance while a client with low risk pays less [4]. Underwriting is also a process in which a business can protect itself, and be sustainable in the long run.

Yan & Bonissone [3] defined medical underwriting as a "complex decision making task traditionally performed by highly-trained individuals. Given an application, the underwriter compares the information provided by the applicant with policy guidelines and standards used by the insurance company." Underwriting is mainly performed by highly skilled professionals with a huge amount of business knowledge. This puts underwriters under a lot of pressure to perform accurate underwriting. Most life insurance companies are developing procedures aimed at improving the quality of underwriting processes, and to support medical underwriters [10]. Underwriting process requires prospective client's medical information to be collected. The collected medical information needs to be evaluated against the company's underwriting guidelines (i.e., medical underwriting rules). There are two common processes used for collecting medical data [16]. These are:

- Traditional underwriting: a process whereby the underwriter conducts a physical interview with a client face to face interaction.
- Tele-underwriting: "a process whereby the underwriter interviews the client on the telephone. Initial questions are predefined and the underwriter can probe for more details".

Either process must ensure that collected data is reliable and accurate.

B. Importance of Medical Underwriting

Two main reasons why medical underwriting is important are:

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

- Insurance companies use the results of medical underwriting to categorize specific clients. Clients with high-risk profiles are administered differently to those with low risk profiles. This is aimed at helping insurance companies to minimize risks and be sustainable.
- Medical underwriting "prevents the uninsured from waiting until they are sick or in need of medical care before purchasing insurance" [21]. Prospective clients wait till it is too late to acquire life insurance. Medical underwriting process helps in ensuring that clients do not wait until they are sick before applying for life insurance.

C. Challenges of Medical Underwriting

The main challenges of medical underwriting are:

- It indirectly affects insurance sale targets. For instance, a sales agent may sell 200 products a day only to have 65% of them declined, due to underwriting rules. Although underwriting ensures quality and sustainability of the business, it may hamper the growth of the business.
- Even with the introduction of automated underwriting, the process still creates a bottleneck within the sales process. This is due to the fact that there is still human intervention involved (i.e., the underwriter and the client).

III. Rule Based Systems

A Rule Based System (RBS) is a decision support model that has been used to develop intelligent systems. Intelligent systems are best used where human problem solving techniques are needed [1]. RBSs are mainly used where information processing can be expressed in the condition-action form. RBSs are composed of pieces of knowledge known as business rules, which are meant to solve business problems. Various industries use RBS in their business processes. These include industries such as mining, chemicals, manufacturing, gaming, insurance, etc.

A rule-based system is composed of three parts. These are [6]:

- Rule base: is a set of business rules that contains conditions and resulting actions of the business.
- Working memory (WM): is also known as facts. It is a database that actively holds input data.
- Rule engine/Inference engine: is responsible for execution of the business rules within the RBS.

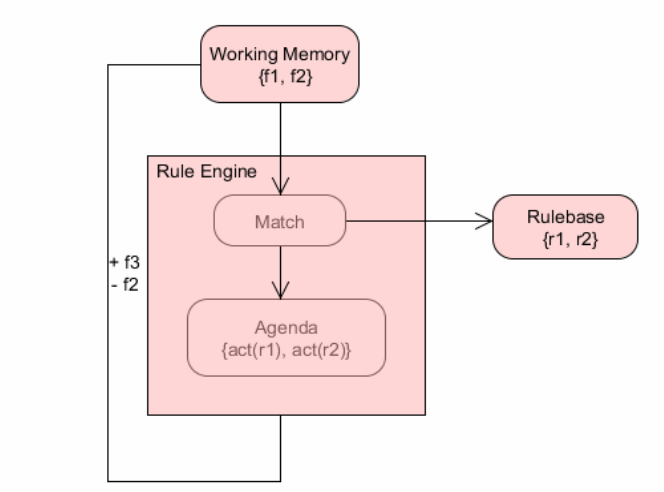

Fig. 1. Execution structure of rule based system [13].

The structure and operation of RBS is illustrated in Fig.1. The WM contains two input facts (i.e., fact1 and fact2) and the rule base contains two business rules (i.e., rule1 and rule2). The rule engine/inference engine executes business rules within the rule base using a pattern matcher and agenda. The results are inserted into the WM [13].

A rule engine is the building block of a RBS, its function is to execute business rules and manipulate the WM. A rule engine can be classified as:

- Forward chaining: this mechanism is data driven. Data gets inserted into the WM; this triggers business rules within the inference engine. This happens until a goal is reached.
- Backward chaining: this mechanism is goal driven, whereby the system searches for rules whose results are mentioned in the data. The system then tests the condition before it uses the rules.

Many industries are relying on RBS to run their core businesses. When applying a RBS a certain criteria needs to be followed, JBoss Drools Team [22] proposes the following criteria:

- When the business problem to be solved is complex for traditional coding paradigm.
- When the business problem has passed the obvious algorithmic solution.
- When the business rules change often.
- When there are non-technical business analysts or knowledge experts.

Andreescu & Mircea [20] defines business rules as "sets of policies and procedures, or definitions that govern the way an organization does business". Business Rules Group [23] defines business rules as "statements that defines or constrains some aspect of the business. This can be a term or a fact, a constraint or a derivation. It is 'atomic' in that it cannot be broken down or decomposed further into more detailed business rules. If reduced further, there would be loss of important information about the business."

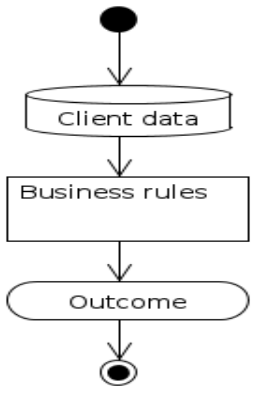

Fig. 2. Medical underwriting rule execution flow

Medical underwriting business rules are jointly developed by insurance business actuaries and business analysts. These professionals analyze the rules and make sure that they are fair to the client and helps sustain the business in the long run. Medical underwriting rules are mainly used to rate clients according to specific categories, for example, low, medium, and high risk category. Medical underwriting rules

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

normally execute based on predefined set of questions presented to the client. Medical underwriting rule execution process is defined as shown in Fig. 2.

Fig. 2 defines medical underwriting process as a flow of client's medical information into enterprise business rule set for execution. After business rules are executed, medical underwriting decision is produced. These components are:

- Client data: medical underwriting requires client's medical information to be available. This can be via Traditional underwriting or Tele-underwriting processes. Business rules: are a variety of business rules in the medical underwriting process. Some of the rules are a simple check that requires a client to respond with yes/no, others are more complex and requires calculation and additional information.
- Outcome: is the results produced by the business rule process. The outcome is used by other business processes within the enterprise.

IV. Techniques of Medical Underwriting and Related Work

Cong [17] defines RETE algorithm as "an efficient algorithm used for matching facts with rule patterns to determine rules that satisfy conditions".

RETE algorithm was invented by Dr. Charles L. Forgy from the University of Carnegie Mellon in 1974. He then presented RETE algorithm in his Thesis in 1979. The research report suggests that, RETE algorithm performs much faster than the naive pattern matching algorithm [14]. RETE algorithm exploits temporal redundancy and structural redundancy to achieve better performance.

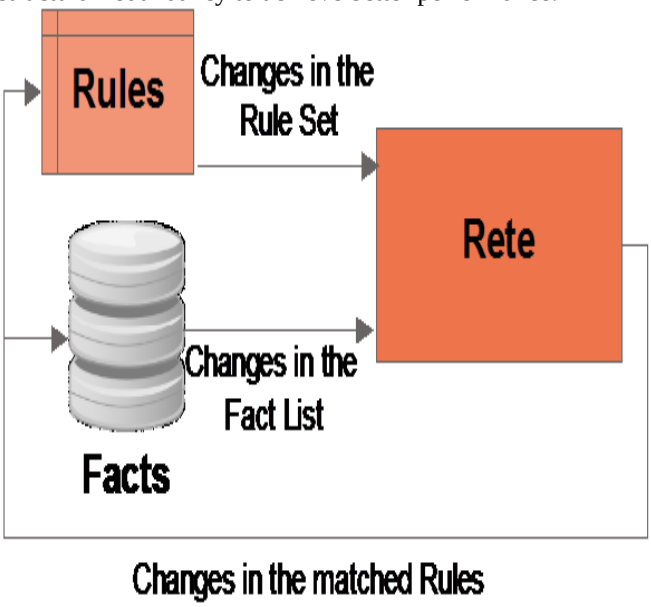

Fig. 3. Rete algorithm input and output [28]

RETE algorithm uses a tree model to execute pattern-matching between rules and facts. The tree contains three types of nodes i.e., Memory node, production and Join [15]. Fig. 3 illustrates how RETE algorithm functions.

The main advantages of Rete algorithm include [28]:

- execution speed and effectiveness. This is achieved within the RETE algorithm's network design. RETE algorithm pools frequently used components so that they are not reinitialized on every computation.
- the ability to share nodes with similar conditions. When two or more rules have a similar condition, RETE uses a single node for all the conditions, unlike creating duplicate nodes. This helps in minimizing duplication of matching effort during execution. This also increases performance.

Several other algorithms have been developed to fill the growing area of rule engines:

- Linear Inference algorithm: works by "ordering rules that can be executed in a single left to right sweep for each inference cycle" [7]. The algorithm requires only one sweep of the rules, in spite of the number of data changed between inference cycles.
- LEAPS algorithm: uses a lazy approach when evaluating rule conditions. It puts all the input data (i.e., facts) on the main stack using the order in which data was asserted in the working memory (WM).
- TREAT algorithm: uses a method called Conflict Set for saving the state in the inference system. Conflict Set is when rules are in a condition-action form, the conditions will be tested against the working memory. All the matches found become candidates for firing [9].
- Linear Inference algorithm uses larger and more-complex data structures. It allows a rule system to reproduce the model of the underlying rules for each inference session. Each change in the fact requires a full cycle of the network, because the system cannot handle multiple changes in the data at the same time. On the other hand each cycle of the network requires access to the memory. This results in a higher memory usage and poor memory access locality. This results to linear inference algorithm not being able to handle highly complex logic effectively [7].
- "LEAPS algorithm is the fastest executable rule sets, often outperforming OPS5 interpreters that use RETE-match or TREAT-match algorithms by far" [19]. However LEAPS algorithm data structures and algorithms are difficult to understand, this is due to the lack of relational database concepts (i.e., relations and select-project-join operators) to help in the critical features of the LEAPS algorithm. This fact makes LEAPS algorithm less attractive to implement in modern rule based systems. LEAPS algorithm is less agile compared to RETE algorithm.

TREAT algorithm is very similar to RETE algorithm. The main performance differences between RETE algorithm and TREAT algorithm is, "RETE algorithm needs fewer comparisons in order to add a new working memory element (WME) to its beta and alpha network than TREAT algorithm" [7]. This makes TREAT algorithm less attractive to use than RETE algorithm.

A lot of research has been conducted in the area of improving medical underwriting. YU-JU et al. [2] presents feed forward neural networks with back-propagation algorithm to build a decision model for various insurance companies. They conducted various experimental researches on this algorithm. Their tests indicate that, neural network requires more variables of data to increase its accuracy. They also discovered that, while neural networks can adapt and learn easily, they have the negative character of a "black box" syndrome, whereby it is difficult to monitor the internal workings of the network. However, the authors failed to

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

indicate the performance of the method they pursued. They also failed to discuss the error rate (i.e., false alarms) generated by neural networks in the study. In this study, we present a clear performance indicator and the rate at which errors are generated.

Horgby et al. [12] presents fuzzy logic method for medical underwriting. The main objective was to show how fuzzy inference system can be used to underwrite a client with diabetes mellitus for a life insurance policy. The findings of the study indicate that fuzzy inference is suitable for helping underwriters cope with the complexity of making suitable decisions. One of the commonly known drawbacks of fuzzy logic is the lack of effective learning capability. However, the study failed in highlighting performance indicators, computational resource usage and error rates. In this study, we use a technique that is easy to train, effective to implement, and performs well under complex conditions.

Bhalla [18] studied a predictive model that can help in enhancing medical underwriting. The predictive analytics by Bhalla [18] used historical and statistical data to analyze and predict the client's risk category. This method allocates a score to applications after predictive analysis. The applications that scores higher are considered to be low risk while those that score lower are considered to be high risk. The results show that the enhancement of the underwriting algorithm to include predictive analysis helps underwriters to focus more on high risk applications. Unfortunately this method is very risky as it does not guarantee the accuracy of the prediction. However this may lead to some high risk applications being overlooked. The authors also failed to provide the performance indicators and failure rate of this predictive model.

V. System Design and Architecture

The main functional requirement of this work was to model a Medical Underwriting Prototype (MUP) that manages client information and perform medical underwriting using a rule based component. We used Unified Modelling Language (UML) to model key activities of MUP. Fig. 4 shows the use case diagram with key activities of MUP application. The proposed architecture is a multi-tier based architecture. Urgaonkar et al. [5] defines a multitier architecture as a framework that provides flexible, modular approach for designing modern internet applications.

Fig. 5 illustrates a proposed MUP multi-tier architecture. The presentation layer handles medical underwriter's interactions with MUP, the business interface layer manages the business rules and business logic, the data access layer manages the access to the database, finally the data store layer stores MUP medical information.

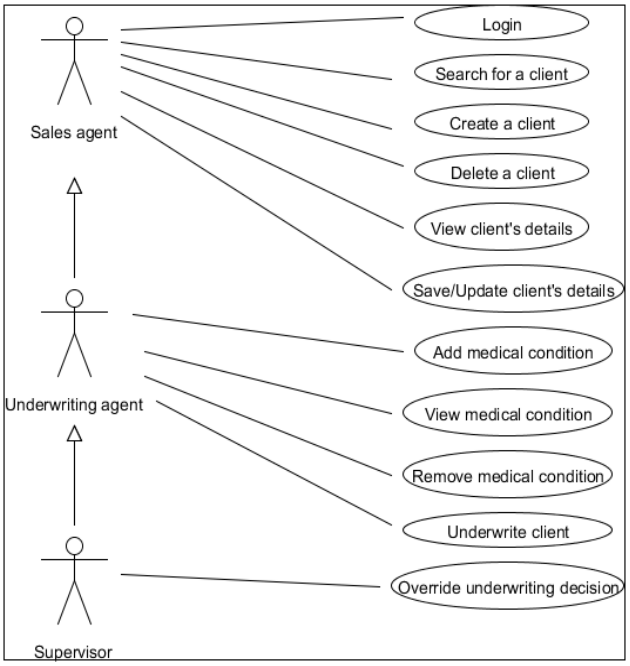

Fig. 4. Use Case diagram

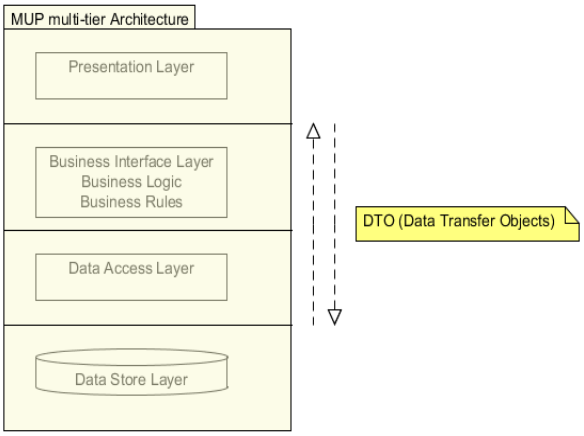

Fig. 5. MUP multi-tier architecture

VI. System Implementation

This work was implemented using JEE platform, JBoss Application Server, Drools Expert for RETE algorithm implementation and MySQL Server for MUP data storage as shown in Fig.7. Fig. 6 shows the MUP Entity Relationship Diagram (ERD) used by MUP.

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

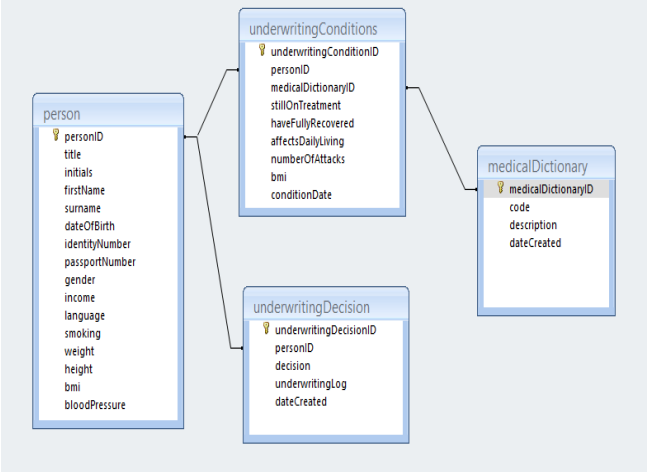

Fig. 6. MUP Entity Relationship Diagram

Drools Expert was acquired by Red Hat as part of their strategic JBoss acquisition in 2006. It is a rule engine implementation of RETE algorithm and it is based on Java platform. The Drools Expert's implementation of RETE is called ReteOO; it is optimized specifically for modern object oriented systems. The reasons we chose Drools Expert are [22]:

- It is a true implementation of RETE algorithm.
- It is based on forward chaining inference mechanism.
- It integrates well with Java based applications.
- It integrates well with Eclipse development environment.
- It is easy to learn, reliable, widely used, and available free of charge.

The core medical underwriting business rules for this work were built within the Drools Expert system using Drools Rule Language (DRL). DRL is a language built specifically for developing business rules targeted at Drools Expert rule engine. The syntax for DRL is easy to learn and to apply. DRL language is capable of expressing highly complex business rules and also supports Java language. Code Snippet 1, illustrates the basic structure of a DRL file.

rule "name"
optional attributes
when
Left Hand Side
then
Right Hand Side
End

Snippet 1. Basic structure of A DRL file

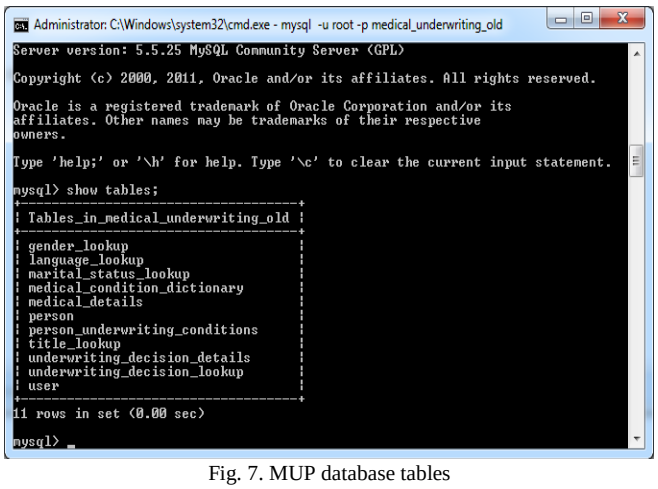

Fig. 7. MUP database tables

In a RETE system, a set of rules sits in production memory, and a set of facts sits in a database called working memory (WM). First a pattern matcher looks at both memories to see which rules have their conditions satisfied by the facts by forward chaining. After that, the matcher generates a list of rules whose conditions have been satisfied, called the Conflict Set. For the given working memory (*WM*), the set of business rules (*R*) and production rule ($[l]p, c => r, a$) the Conflict Set is defined as:

$$CS = \{(l, \{f_1, ..., f_k\}) \mid \exists([l]p, c \Rightarrow r, a) \in Rwhichis(\delta, WM) - fireableon\,WM \quad (1)$$

CS creates a unique element $R@WM_n - conflicset$, where $WM = \{f_1 ... f_k\}$

The Conflict Set can either be empty (i.e., no rules fire-able), unitary (i.e., only one rule can fire), finite (i.e., a finite number of rules is activated) or infinite (i.e., an infinite number of matches are found) [11]. In a situation where infinite or finite match occurs, RETE will decide which rule should be applied. This strategy is called resolution strategy and is defined as:

$$WM_0 \Rightarrow WM_1 \Rightarrow ... \Rightarrow WM_n \quad (2)$$

This returns a unique element $R@WM_n - conflictse\,t$

The transition of medical underwriting data inside the RETE network is defined as:

$$m? \xrightarrow{a} ms \quad (3)$$

Where $m?$ is the input token into the network, $ms!$ is the list of output tokens and a is an optional action to be performed.

VII. Experiment and Results

We used Apache JMeter to test MUP. The test scenarios were based on memory usage, response time and the rate of false alarms generated. We compared the test results of MUP with that of the application currently used in the insurance environment. The current underwriting system uses a traditional conditional statement (i.e., if-then-else) to perform pattern matching and business rules execution. We refer to this implementation as "naive algorithm" because it does not have features of a true rule execution algorithm.

We loaded both systems with 500 rules and the number of medical conditions per client increased from 1 to 5000 in a period of 1 hour. The test scenario shows that MUP application (using RETE algorithm) uses less memory resources and the memory usage is consistent. The current underwriting application memory usage is high and unstable.

We then loaded both rule bases (i.e., MUP application and the current underwriting application) with 500 rules, and 1000 medical conditions per client were created randomly. An Average of 100 requests was sent to the service layers per second. The test was run within a period of 1 hour. The objective was to determine the response time it takes to process the request. Table I shows the results of this test scenario by JMeter where UnderwritingProcessEngine.class is the MUP application and ClientUnderwriter.class is the

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

current underwriting application. This test scenario shows that MUP average response time under extreme usage is 80.32 milliseconds; the current underwriting application average response time under extreme usage is 170.89 milliseconds.

Table I: Response time report for MUP application and current underwriting systems

JAVA REMOTE TEST REPORT				
Time:	60 minutes			
No of threads: No of request:	20 100rps			
Test module	Test point	Avg. response	Errors	Results
ClientUnderwriter.class	underwrite	170.89ms	7	OK
UnderwritingProcessEngine.class	executeUnderwritingRules	80.32ms	2	OK

Finally, both rule bases (i.e., MUP application and the current underwriting application) were loaded with 500 rules. The number of medical conditions per client increased from 1 to 1000. The numbers of prospective clients profiled were 954. The result shows that MUP system (with RETE algorithm) generates fewer errors as compared to the current underwriting system (with naive algorithm). Error generation of RETE algorithm stabilizes as the data increase; this was due to RETE being able to learn from previous errors. The current system error generation increases exponentially as the input data increases. Fig. 7 illustrates this test scenario.

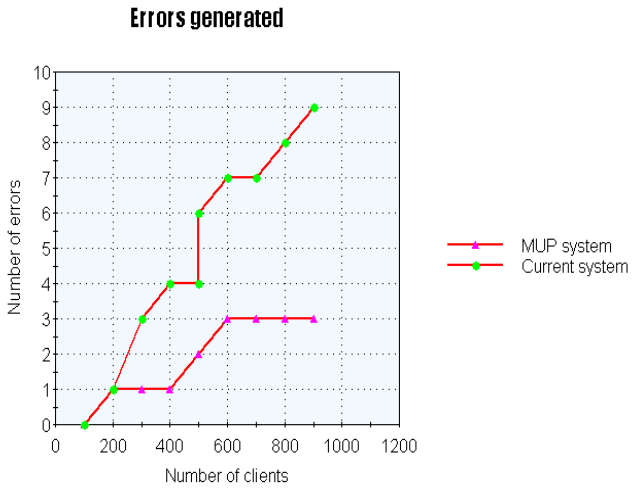

Fig. 7. Errors generated by both RETE algorithm and the naive algorithm

VIII. Conclusion

This paper presented a medical underwriting system that uses RETE algorithm to execute medical underwriting business rules. We presented a brief overview of medical underwriting and rule based systems. We presented system architecture and implementation of MUP. The experiment on the system was conducted, and the results indicated that MUP uses less memory, generates fewer errors and performs faster than the current underwriting system in an insurance environment.

References

[1] G. F. Luger, Artificial Intelligence: Structures and Strategies for Complex Problem Solving, 6. edition, Addison-Wesley, 2009.
[2] L. YU-JU, H. CHIN-SHENG and L. CHE-CHERN, "Determination of Insurance Policy Using Neural Networks and Simplified Models with Factor Analysis Technique," WSEAS TRANSACTIONS on INFORMATION SCIENCE & APPLICATIONS, 2008.
[3] W. Yan and P. P. Bonissone, "Designing a Neural Network Decision System for Automated Insurance Underwriting," IJCNN IEEE, pp. 2106-2113, 2006.
[4] P. Wang, "Effects of Disability-Based Underwriting Prohibitions on the Labor Market," Asia-Pacific Journal of Risk and Insurance: Vol. 4: Iss. 1, Article 4, 2009.
[5] B. Urgaonkar, G. Pacificiy, P. Shenoy, M. Spreitzery and A. Tantawiy, "An Analytical Model for Multitier Internet Services and Its Applications," 2005. [Online]. Available: http://lass.cs.umass.edu/papers/pdf/SIGMETRICS05.pdfAmir. [Accessed 02 02 2013].
[6] C. Seitz, S. Lamparter, T. Schöler and M. Pirker, "Embedded Rule-based Reasoning for Digital Product Memories," Siemens AG, Corporate Technology, Autonomous Systems, 2010.
[7] Oracle, "Linear Inferencing: High-Performance Processing, Oracle White Paper," 2009. [Online]. Available: http://www.oracle.com/us/industries/public-sector/029743.pdf. [Accessed 03 03 2013].
[8] R. K. Nepal, "The Insurance Market in Nepal," Bachelor Thesis, Arcada University, 2012.
[9] D. P. Miranker, "TREAT: A Better Match Algorithm for AI Production Systems," in National Conference On Artificial Intelligence - AAAI, 1987.
[10] R. C. M. Lee, K. P. Mark and D. K. W. Chiu, "Enhancing Workflow Automation in Insurance Underwriting Processes with Web Services and Alerts," in Proceedings of the 40th Hawaii International Conference on System Sciences, 2007.
[11] G. Huet and F. Fages, "Complete sets of unifiers and matchers in equational theories.," Theoretical Computer Science, pp. 43(1):189-200, 1986.
[12] P. Horgby, R. Lohse and Nicola-Alexander, "Fuzzy Underwriting: An Application of Fuzzy Logic to Medical Underwriting," Journal of Actuarial Practice vol 5, No. 1, 1997.
[13] E. Friedman-Hill, "Jess in Action: Rule-Based Systems in Java," Manning, 2003.
[14] C. L. Forgy, "On the Efficient Implementation of Production Systems," PhD Thesis, Carnegie-Mellon University, 1979.
[15] A. G. D. de Oliveira, "Sistema para Validação da Correcção e Completude de Encomendas," Thesis, Universidade Tecnica de Lisboa, 2012.
[16] J. Crumiller, "The Impact of Tele-Underwriting on Mortality Experience," Princeton Consultants Tele-Underwriting Practice, 2006.
[17] C. P. Cong, "An approach to adaptive inference engine for rule-based consultation systems," Doctorate thesis, University of Duisburg-Essen, 2007.
[18] A. Bhalla, "Enhancement in Predictive Model for Insurance Underwriting," International Journal of Computer Science & Engineering Technology (IJCSET), p. Vol. 3 No. 5, 2012.
[19] D. Batory, The LEAPS Algorithms, The University of Texas, 1994.
[20] A. I. Andreescu and M. Mircea, "Perspectives on the Role of Business Rules in Database Design," Database Systems Journal vol. III, no. 1/2012, 2012.
[21] Wikipedia, "Medical Underwriting," 2012. [Online]. Available: http://en.wikipedia.org/wiki/Medical_underwriting. [Accessed 12 10 2012].
[22] JBoss Drools Team, "Drools Expert User Guide," 2011. [Online]. Available: http://docs.jboss.org/drools/release/5.3.0.Final/drools-expert-docs/html. [Accessed 15 01 2013].
[23] Business Rules Group, "Defining Business Rules: What are they Really?, 3rd edition," 2002. [Online]. Available: http://www.BusinessRulesGroup.org. [Accessed 20 11 2012].
[24] Wiktionary, "architecture," 2013. [Online]. Available: http://en.wiktionary.org/wiki/architecture. [Accessed 02 02 2013].
[25] SoapUI - http://www.soapui.org
[26] JMeter - http://jmeter.apache.org
[27] JUnit - http://junit.org.
[28] De Oliveira, A. G., Sistema para Validação da Correcção e Completude de Encomendas. Thesis, Universidade Tecnica de Lisboa, 2012.

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)