International Journal of Computer and Information Technology (ISSN: 2279 – 0764)
Volume 06 – Issue 06, November 2017

An Improved Fuzzy Rule-Based System for the Diagnosis of Software Rots Metrics

Adedokun Ajibike Mariam
Department of Computer Science
University of Port Harcourt, Nigeria

Onyejegbu Laeticia Nneka
Department of Computer Science
University of Port Harcourt, Nigeria
Email: nneka2k [AT] yahoo.com

Abstract--- **Software rot has affected the productivity of various organizations due to the fact that our modern computer facilities are not immuned to Software and Architectural rot. Numerous algorithms are made available in order to diagnose related Software rot issues. Some of these algorithm application for diagnosing Software rot issues generate problems such as slow service delivery, which results from non-application of Open-Source fuzzy logic tools, and limited storage capacity for datasets due to non-usage of Relational Database Management System (RDBMS). However, this study addresses the mentioned problems by reducing the complexity involved in diagnosing Software rot metrics through the adoption of Structured System Analysis and Design Methodology (SSADM), interfaced with FuzzyTech (FT) algorithm, and also adopts MySQL Database for adequate storage of datasets. Furthermore, the Proposed System is divided into two phases. The first phase involves the recognition of Software rot symptoms; the second phase diagnoses and represent the outputs in Triangular Form Membership Function format through the application of FuzzyTech(FT) open-source Software and C# programming language. FuzzyTech(FT) was used as a platform to improve the fuzzy logic system. In addition, this study mainly solves the problem of slow service delivery in diagnosing Software rot issues, and limited storage capacity for datasets used. Our result also shows that an essential feature of the fuzzy system is that a numerical value does not have to be fuzzified using only a membership function. In other words, a value can belong to multiple sets at the same time. For instance, Software Symptom value can be considered as negative, zero and positive at the same time with different degree of membership.**

Keywords--- Fuzzy Logic, FuzzyTech, Membership Function, Metrics, Open-Source, Software rot, Triangular Form.

I. INTRODUCTION

As stated in Lehman's first law [1] to maintain the usefulness or suitability of software in real-world applications, the software must have the ability to evolve with changes in requirements.
Unfortunately, most of the software systems used in companiesare susceptible to software rot and ageing. As such existing software systems may become less maintainable as time progresses. Software rot in software products is a common problem associated with software systems.
Software Rot is a situation where the software system acquires flaws or bugs and other issues which makes it incompatible with the present situation because of factors such as changes in technology, disagreement amongst stake-holders. These changes may force the company to abandon the software application. A detailed study of the existing work in this area reveals that the cost or difficulty in terms of money, skill and effort involved in the maintenance of a software increases with increase in time. That is as the software ages, it becomes more difficult to effect changes. Obviously, it is more viable to replace the system or application with a new or redesigned system than to keep on maintaining the existing system. The design decision considered during development system may interfere with requirements that are to be introducedas a result of system evolution. Software Rot is the aftereffect of repeated unrestrained maintenance reduces the system's quality. The consequence of uncontrolled maintenance overtime which degrades quality of system. In that case it becomes mandatory to replace the existing one old system. [1]
Notwithstanding, complete systemreplacement is dangerous because it comes with great effect on technology, skill and organizational financial state. Often, replacement involves, retraining of user(s) and operators and the new system may not have the core functionalities present in the old system. These factors may place unbearable financial responsibility on the organization. Also, developing a new software usually involve writing of programs with lines of code running into several millions. These codes are grouped into several classes. To be sure that the new software will not also move into software Rot, there is need for a machine learning algorithm which can take care of voluminous dataset for the finding and discovering of pattern such that the system will have a learning ability.
Machine Learning algorithms consist of methods and procedures which enables learning in machines such as computer. To handle the challenges of software rots, it is better to adopt the evolutionary maintenance approach instead of complete replacement of the existing system. This way the impact of software rot could be minimized. [2]

II. IIREVIEW OF RELATED WORK

Software rot is also known as "code rot", "bit rot", "software erosion", "software decay" or "software entropy" describes the perceived "rot" which is either a slow deterioration of software performance over long period of time or its diminishing responsiveness that will eventually lead to software becoming faulty, unusable, or otherwise called "legacy" and in need of "upgrade".

[3] made a distinction between architectural rot and architecture drift. The former is caused by violation of the architecture while the late is caused by architectural insensitivity. Architectural rot, according to them is the result of 'violations of the system's architecture.

They emphasized on the importance of the application of an object oriented fuzzy logic system in tackling Software Architecture rot and Architectural drift.

[4] investigated the aging process of software systems. He observed that aging is caused by wrong decisions at design phase. He confirmed that rot is a result of drift in architecture. He suggested that more considerations be placed on change especially during the design phase of a software and that coding be done after the system has been properly designed.

[5] worked on the properties of architectural rot. They showed ways which architectural quality could be properly accessed

[6] proposed refactoring as a method of countering rot in system. Refactoring is an improvement strategy that involves changing the existing source code. There are different methods of refactoring which can be use to resolve violations. According to [7] some of the refactoring methods are more difficult to use than others.

[8] proposed a new approach for system designs. In this approach design related decisions are used to describe the architectural structure of the system. They observed that to solve the problems leading to software crisis, there is need for more efficient design and development methods and tools.

[9] are of the view that efficient design and development methods only delay the occurrence of rot and not a perfect solution to the problem of rot. They suggested that dealing with the causes of rot will yield more positive results. [10] carried out a comprehensive evaluation of evolutionary design approaches. They observed that decisions made early in the design process may hinder the incorporation of other requirements late in the evolution of the system. To solve this problem, in this work many of such decisions were reversed. This reversal seems helpful but the consequence of such a radical change on functional systems could be unbearable.

Furthermore, fundamental algorithms which can categorize software into one of four classes which are High- rot, low-rot and Moderate-rot have been developed. This work however uses the supervised machine learning technique to analyze traffic in network

III. ANALYSIS OF EXISTING SYSTEM

The influence of Software Architecture Rot has increased tremendously in Software Systems. However, the methodology of the existing system for diagnosing Software Architecture Rot is known as the Architectural Pattern Decision System (APDS). During the process of Software Architecture Design, a lot of architectural patterns are being implemented by the architect. [3] Architectural Pattern Decision System (APDS) is a unique system that aids in the diagnosis of Software Architecture Rot. This is because the major reason for the existence of Architectural Rot is the absence of useful design decisions This decisions are often made during the design of the systems architecture and are part of the resulting architecture. Several types of Architectural Pattern Decision System (APDS) include: Archium, AREL, ArchDesigner and AQUA. Most of these decision system examples use different methodology to implement design decisions.

i) **How the Architectural Pattern Decision System (APDS) works:**

The Architectural Pattern Decision System (APDS) detects and diagnose Software Architecture Rot by gathering the most necessary and useful design decisions relating to the system's architecture.

Furthermore, most developers make use of most used architectural models in their design.

Hence, the Architectural Pattern Decision System (APDS) captures what and how the architectural patterns are used, and then generate a set of corresponding output [3].

In Software Architecture Design of the existing system, the architects usually adopt some architectural patterns.

ii) Process Diagram of an Architecture Pattern Decision System (APDS)

From figure1, horizontally, analysis of system requirements, architectural design, implementation and deployment make up the stages of the software life cycle.. Vertically, the segments; manual and tool support represent the manual activities and the finished events in the system. Furthermore, in the first phase, the architects will be able to arrange the entire Architectural Significant Requirements (ASR) of the System. [11]

The Proposed System for the diagnosis of Software Rot Metrics is known as FuzzyTech System. The FuzzyTech System is a unique and efficient way for developing tools for fuzzy logic and neural fuzzy solutions. Furthermore, it comprises of an Open-Source FuzzyTech Software that enables adequate fuzzy logic designs. The Methodology of this study further covers the analysis and development of the Proposed System.

The Methodology used for the Proposed System is Structured System Analysis and Design Methodology (SSADM). This Methodology uses the systemsanalysis and design strategy, the algorithm used is fuzzyTech algorithm.

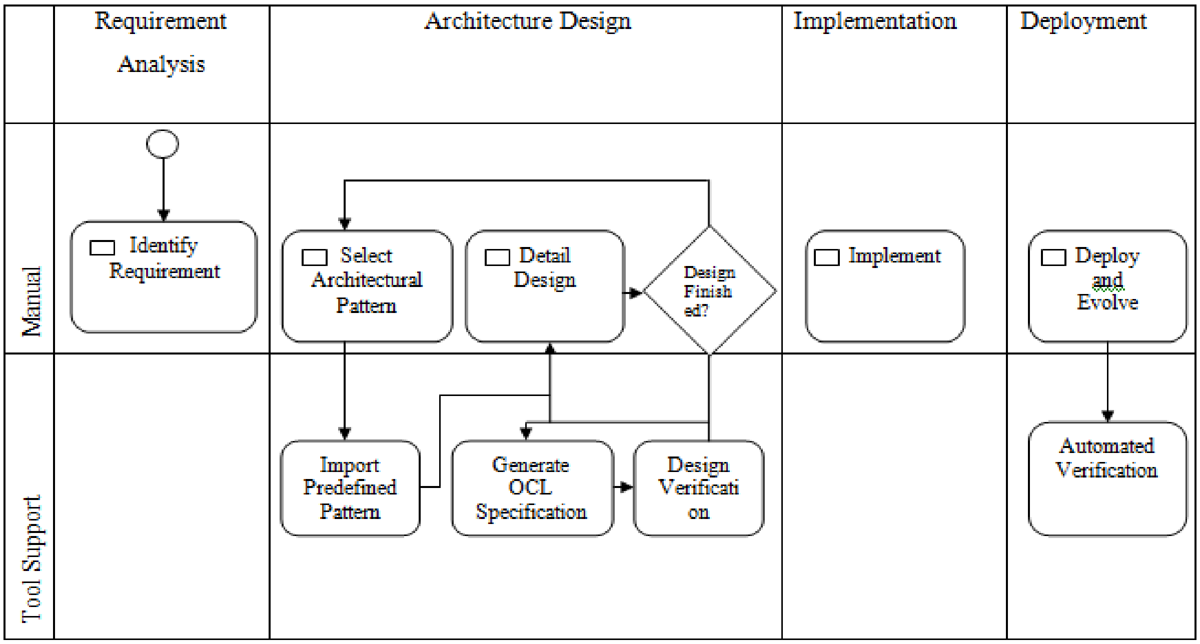

Fig. 1: Process Diagram of an Architecture Pattern Decision System (APDS)
(**Source:** Perry et al, 2011)

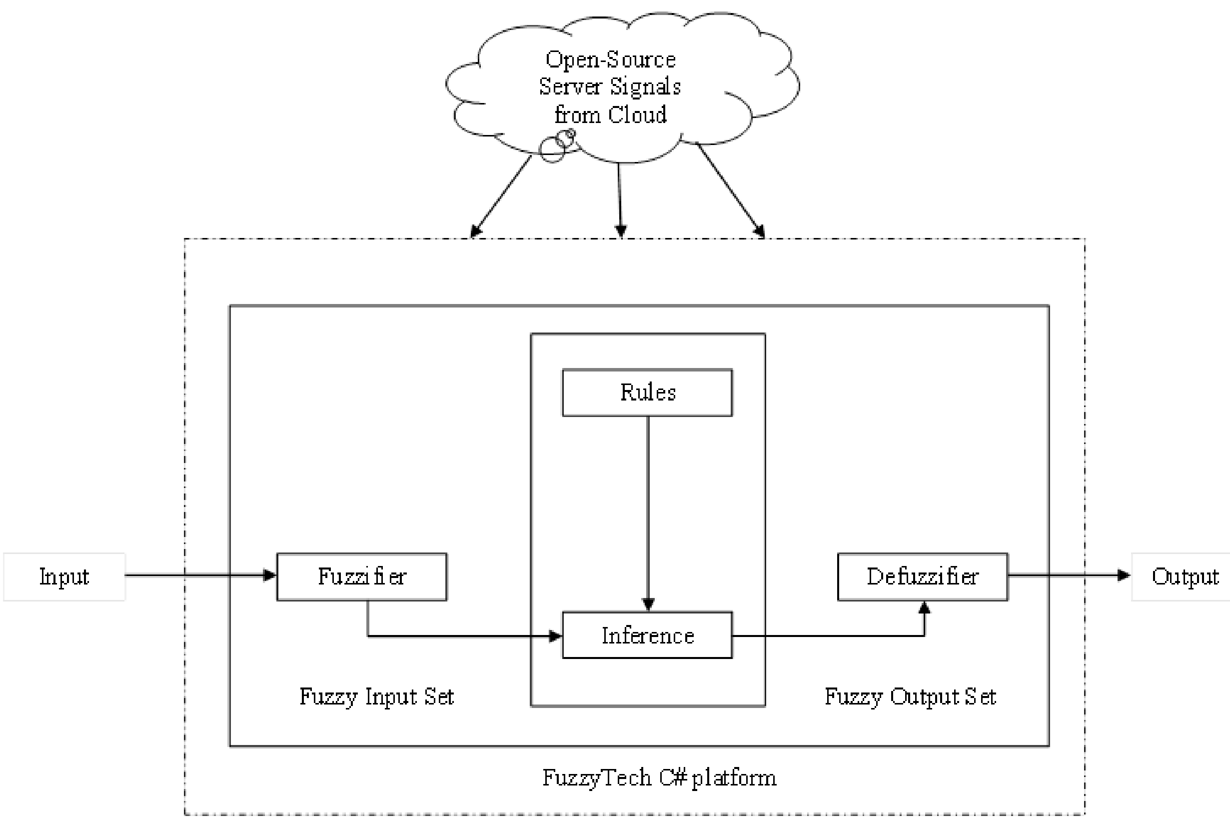

Fig. 2: Sample Design of a FuzzyTech System (Proposed System)

Table. 1: Basic Software Rot Diagnosis for MariamLaeticia Fuzzy Tech Systems.

SYMPTOM	METRICS	DIAGNOSIS	POSSIBLE SOLUTION
Missing Documentation	0.8 unit	Missing knowledge	Re-document through reverse engineering.
Dead Code	0.75 unit	Dead Code	Remove Dead Code
Missing Functionalities	0.8 unit	Missing Knowledge	Create, Split or modify programs to support those functionalities.
Useless Data	1.0 unit	Anomalous Data	Remove programs that create obsolete data.
Poor Lexicon	0.6 unit	Missing Knowledge	Rename and Refactor
Source less Programs	0.8 unit	Pollution	Rewrite Source code by means of reverse engineering
Pathological Files	0.55 unit	Coupling Issues	Re-factor by means of reverse engineering.

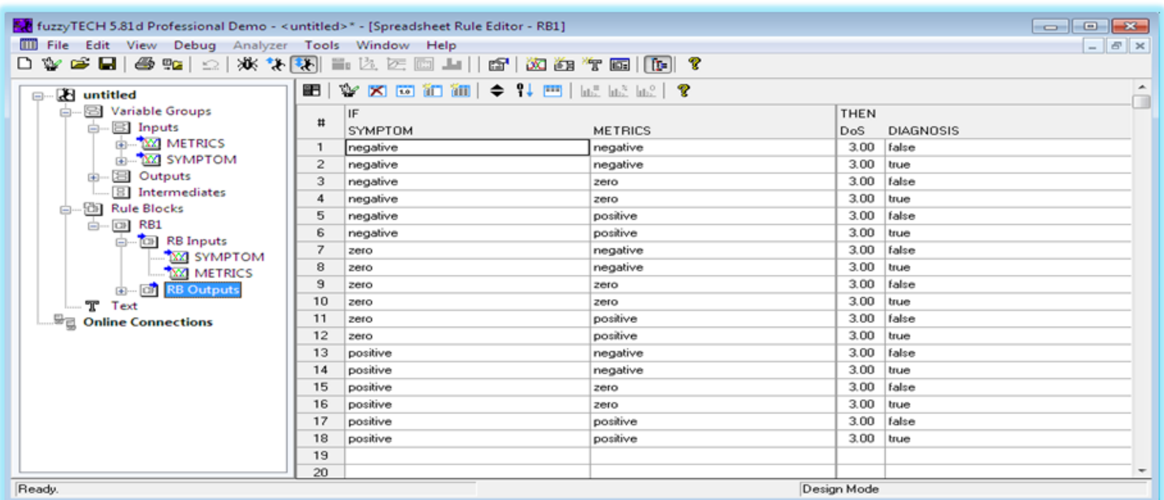

Fig.3: FuzzyTech 5.8: Proposed Sample Fuzzy Rules Software Rot Diagnosis System

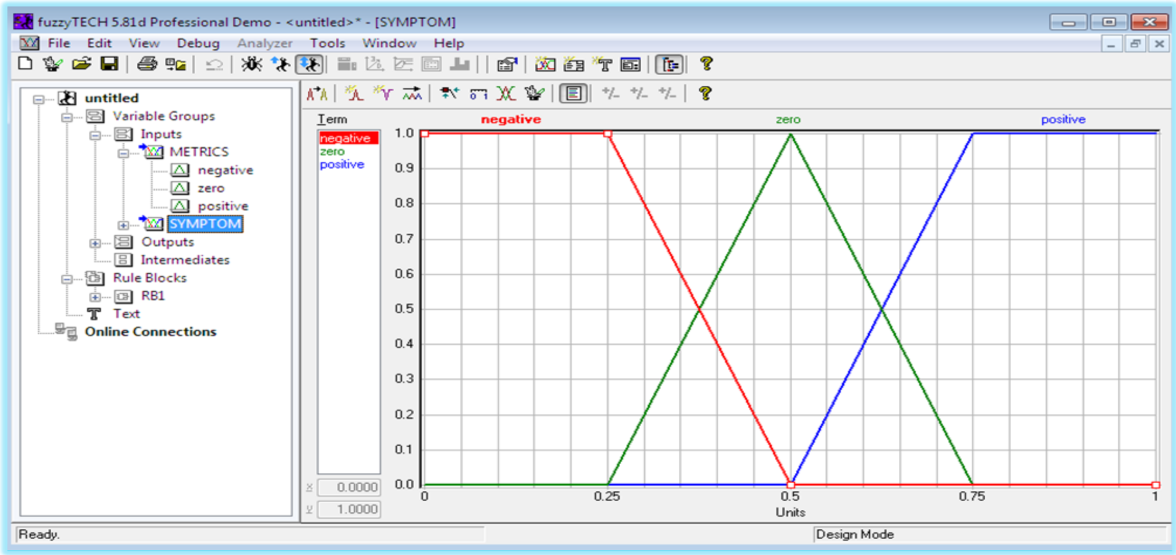

Fig.4: FuzzyTech 5.8: Triangular form Membership Function of SYMPTOM = {negative, zero, positive}

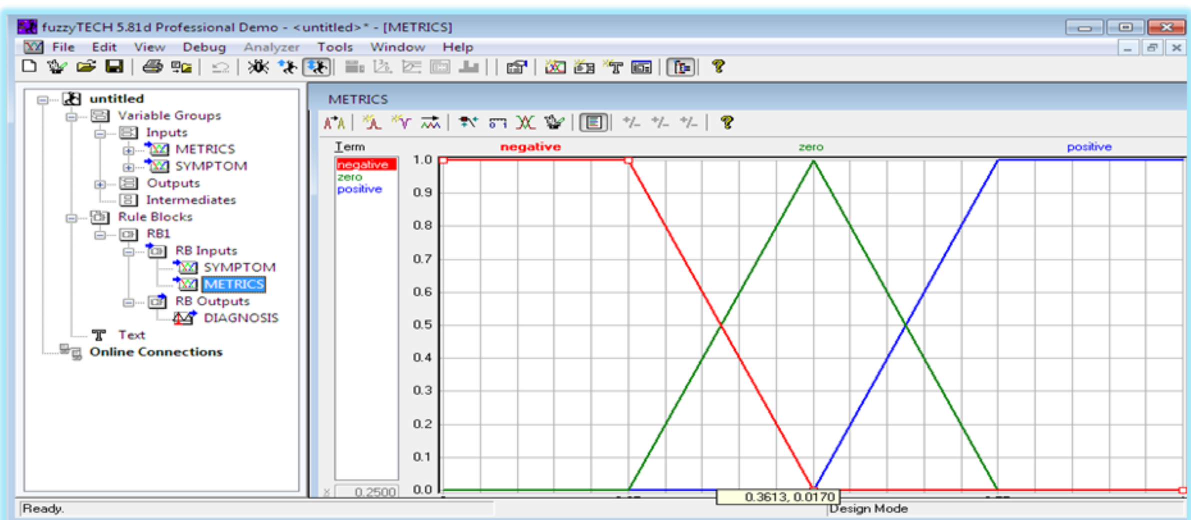

Fig.5: FuzzyTech 5.8: Triangular form Membership Function of METRICS = {negative, zero, positive}

IV. RESULT DISCUSSION

An essential feature of the fuzzy system is that a numerical value does not have to be fuzzified using only a membership function. In other words, a value
can belong to multiple sets at the same time. For instance, Software Symptom value can be considere as negative, zero and positive at the same time with different degree of memberships.

V. CONCLUSIONS

The existing system employs the use of predefined and frequently used architectural patterns. This process is not compatible with some Operating Systems (O.S) and often results to slow service delivery in the diagnoses of Software Rot metrics. Hence, there is need to appreciate the existing system, but as well develop an easy and efficient Fuzzy-base system for the diagnoses of Software Rot metrics.
The proposed system highlights the efficiency in Software Rot metrics diagnosis through Fuzzy Logic applications. Input datasets are gathered and channeled to a fuzzy set using fuzzy linguistic variables, fuzzy linguistic terms and membership functions (Fuzzification).
Thereafter, an inference is made based on a set of rules. Finally, the resulting fuzzy output is mapped to crisp outputs using the membership function in the deffuzification step.

REFERENCES

[1] N. Lehman 2007 "Analysis of Software Design Erosion Issues, International Journal of Advanced Research in Computer and Software Engineering", vol. 3, 4-10

[2] J. Bosch 2000 "Design and use of Software Architectures Adopting and Evolving a Product- Line Approach"

[3] T. Perry and B. Wolf, 2011 "Diagnosis of Software Erosion through Fuzzy Logic" Alarcos Research Group University of Castilla, La-Mancha, 1-22

[4] C. Parnas. 2004. "Diagnosis of Software Erosion through Fuzzy Logic", Alarcos Research Group University of Castilla, La-Mancha. 14-22.

[5] B. Jaktman, M. Liu, B. Leaney 2009 "Structural Analysis of the Software Architecture- A Maintenance Assessment Case Study", 22-24.

[6] E. Gamma, R. Helm, R. Johnson and J. 2005 Vlissides, "Design Patterns: Element of Reusable Object Oriented Software",

[7] D. Roberts, M. Liu, M. Holland, B. Jaktman and P. Villa," 2013. Analysis of Software Design Erosion Issues, International Journal of Advanced Research in Computer and Software Engineering", vol. 3, 4

[8] A. Jansen, J. Bosch 2005 "Software Architecture as a set of Architectural Design Decision", ISBN: 0-7695-2548-1, 109-120.

[9] V. Gurp, 2012 "Analysis of Software Design Rot Issues", International Journal of Advanced Research in Computer and Software Engineering, vol. 3, 2. 6-9

[10] V. Gurp and J. Bosch 2009 "Design Erosion: Problems and Causes", Journal of Systems and Software, vol. 61 105-119

[11] M. Liu, P. Toprac, L. Horton, T. Yuen 2011 "Examining the Architectural Design of Media Rich Cognitive Tools in Multimedia Problem-based Environment", 36