RULE-BASED EXPERT SYSTEM FOR SOFTWARE VERSION CLASSIFICATION WITH DECISION RULES

BY

Humberto Cuteso Matumueni [1], Lamento Emilio Adão[2], Joana Bartolomeu[3], Maria Afonsina Álvaro[4], Domingos Pascoal Sebastião[5]

1. Department of Engineering and Technological Innovation, Higher Polytechnic Institute of Soyo, Angola
2. Coordination of the Computer Engineering course

Corresponding author E-mail:hkuteso@hotmail.com

ABSTRACT

Software version control is an essential activity in software engineering, enabling the management, tracking, and maintenance of software products throughout their lifecycle. Traditional version control systems such as Concurrent Versions System (CVS), Apache Subversion (SVN), and Git provide mechanisms for storing, controlling, and recovering software versions; however, they do not perform intelligent classification of software releases. This study proposes the development of an expert system for software version classification using artificial intelligence techniques. The methodology involved knowledge acquisition through questionnaires applied to experts, the construction of IF...THEN production rules, the use of decision tables, and implementation using the Exsys Corvid platform. The expert system classifies software versions into categories such as Alpha, Beta, Stable, Outdated, and Critical based on predefined criteria. To evaluate the proposed approach, a comparative analysis was conducted between the expert system and traditional evaluation methods using Cohen's Kappa coefficient. The results demonstrated a fair level of agreement (K = 0.333) between evaluators, indicating that the expert system is capable of supporting decision-making processes related to software version management. The study concludes that integrating expert system techniques into software version control environments can improve consistency, automation, and decision support in software release management.

Keywords: Expert Systems, Software Version Control, Artificial Intelligence, Production Rules, Software Engineering, Decision Tables, Exsys Corvid

1. INTRODUCTION

Modern software development demands efficient mechanisms for controlling, managing, and verifying different versions of computer systems. As software projects

become more complex and collaborative, the need arises for technologies capable of tracking changes, recovering previous versions, coordinating development teams, and ensuring stability throughout the software lifecycle. Version control technologies play a fundamental role in software engineering, allowing programmers to work simultaneously on the same project without compromising the integrity of the source code. These tools enable the storage of the history of changes made, the control of updates, and the efficient management of different software versions.

Among the main technologies used worldwide, Apache Subversion stands out. Apache Subversion is a centralized version control system developed to overcome limitations existing in CVS (Concurrent Versions System), and SVN (Subversion) are version control systems used to manage changes in software files, documentation, and collaborative projects. SVN (Subversion) was developed to replace the limitations of CVS, offering greater reliability and advanced features. It allows for history storage, recovery of previous versions, collaborative management, and efficient control of changes made to software projects. The technology has become widely used in business environments due to its stability and ease of administration. (CollabNet, 2000)

Git is a distributed version control system created to support the development of the Linux operating system. Git stands out for its speed, security, flexibility, and decentralized work capacity, allowing each user to have a complete copy of the repository. Currently, it is one of the most widely used technologies in collaborative software development worldwide. (Linus Torvalds, 2005)
The Concurrent Version System (CVS) was one of the first popular version control systems used in software engineering. CVS allowed, for the first time, multiple programmers to work simultaneously on the same project, facilitating collaboration in development teams. Despite its limitations compared to modern technologies, CVS had great historical importance in the evolution of version control systems. (Dick Grume, 1986)
In general, these technologies have contributed significantly to the evolution of software engineering, providing efficient mechanisms for managing, controlling, and verifying software versions in collaborative and distributed environments.

2. THEORETICAL FOUNDATION

Software Version

According to ITIL (2013), Release Units and Release Packages are defined as: a) Release Unit: small quantities of software; b) Release Package: a set of release units or even a single unit, but larger in size. Each Release Unit or Release Package of a software version may have variations in its structure. This variation can cause impacts in the production environment, which requires analysis by a specialist who classifies each version by its criticality as High, Medium, and Low, but this classification is subjective and may not reflect the real criticality of a software version.

Expert System

According to Wagner (2017), Expert Systems are knowledge-based for solving problems in a given domain in the same way as a human expert. To represent the knowledge of human experts, the ES must possess not only a set of information, but also the ability to use it in problem solving. This ability represents a series of decision rules that the expert uses to solve problems. (Pannu, 2015).

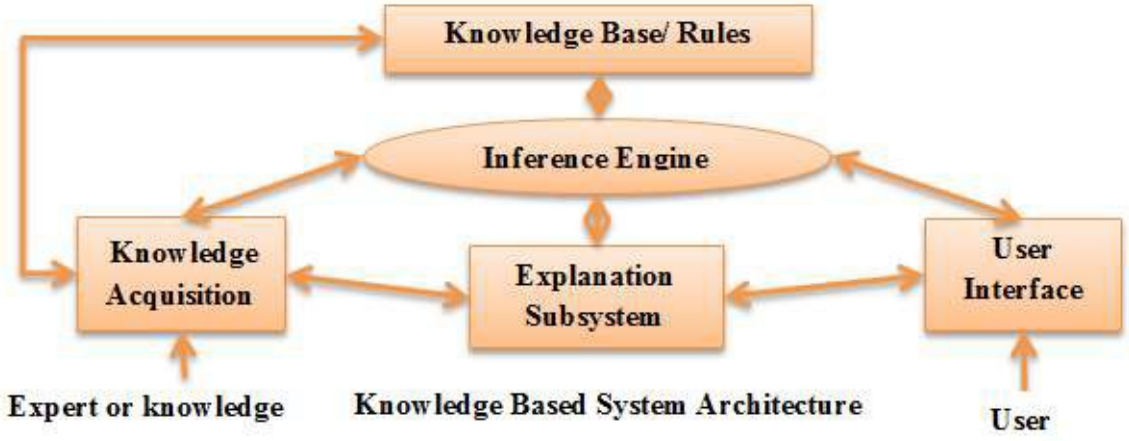

Figure 1. Knowledge Based System architecture
Source:(Quintana et, ali, 2015; (Riberric and Vedrina,2009)

The representation of knowledge by decision rules is used in expert systems. The justification is the naturalness it represents for humans, since the condition-action pair for reasoning and deciding is also used by human beings (Dymova; 2016; Guptal, 2017).

3. METHODOLOGY

The methodology used in this study is based on the construction of an expert system for software version classification, using artificial intelligence and knowledge acquisition techniques. The methodological process integrated different instruments and techniques, namely the questionnaire, IF...THEN type production rules, decision tables and the Exsys Corvid tool.

Initially, a questionnaire was applied with the aim of collecting information and specialized knowledge about criteria used in the classification of software versions. The questionnaire made it possible to identify important factors such as structural changes, inclusion of new features, bug fixes and software stability.

In summary, the sequence of development, application, and validation of the decision rules and the expert system is demonstrated.

1. Application of a questionnaire to the experts
2. Consensus of the experts regarding the criticality classifications
3. Classification of the versions, after consensus
4. Elaboration of the production rules based on the consensus
5. Validation tests of the production rules
6. Comparison of the experts' classification with the classification performed by the expert system

Based on the information collected, production rules of the IF...THEN type were defined, a technique introduced by Edward Feigenbaum in 1970. These rules allowed the representation of specialized knowledge in a logical and structured way, enabling the system to perform automatic inferences.

Rule 1
IF the software is in internal testing phase
THEN classify as Alpha Version
Rule 2
IF the software has been made available to external users for testing
THEN classify as Beta Version
Rule 3
IF the software does not present critical flaws
THEN classify as Stable Version
Rule 4
IF a more recent version is available
THEN classify as Outdated Version
Rule 5
IF the software presents security flaws
THEN classify as Critical Version

Subsequently, the Decision Table, proposed by Morton A. Harrison in 1965, was used to organize the different combinations of conditions and their respective actions. The decision table facilitated the logical modeling of the expert system, allowing for greater clarity and consistency in the decision-making process.

Table 1. Decision table

Conditions/Actions	R1	R2	R3	R4	R5	Else
Software in internal testing phase	1	0	0	0	0	
Software made available to external users for testing	0	1	0	0	0	
Software does not present critical flaws	0	0	1	0	0	
A more recent version is available	0	0	0	1	0	
Software presents security flaws	0	0	0	0	1	
Alpha Version	X					
Beta Version		X				
Stable Version			X			
Outdated Version				X		
Critical Version					X	
Other						X

Source: Autor

Finally, the system was implemented using Exsys Corvid, a tool specialized in the development of rule-based production expert systems. Exsys Corvid enabled the creation of the knowledge base, implementation of logical rules, and construction of the inference engine responsible for the automatic classification of software versions.

Expert System Implementation

Figure 2. Main screen for the rule-based expert system for software version classification
Source:author

A combinação destas técnicas permitiu desenvolver um sistema especialista eficiente, The combination of these techniques allowed the development of an efficient, organized expert system capable of supporting the intelligent management of software versions.

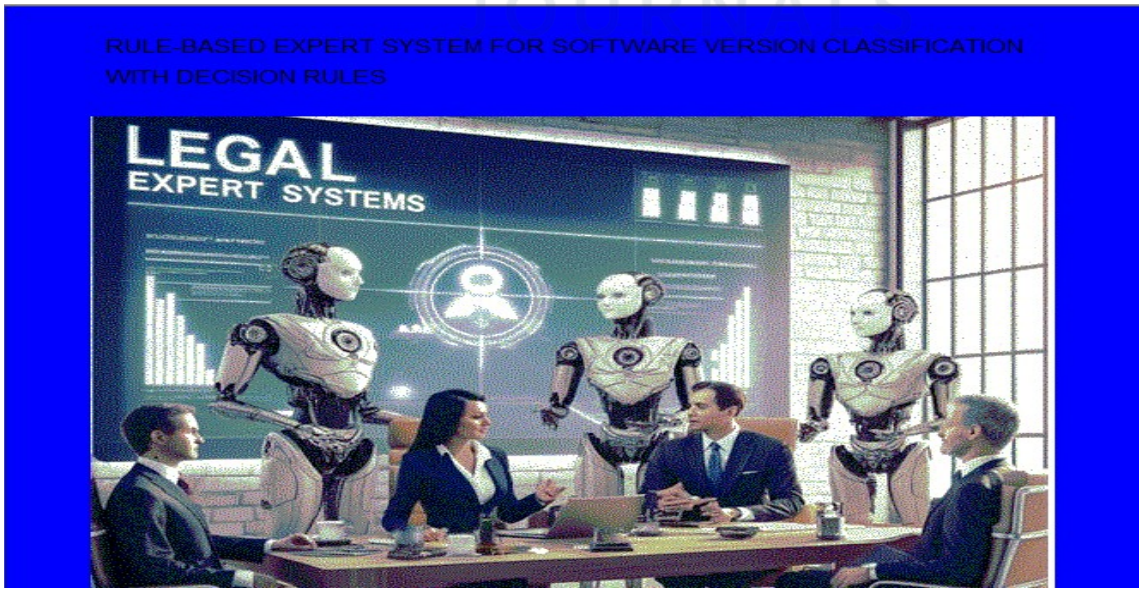

Figure 3. Input data for the rule-based expert system for software version classification.
Source:author

Figure 4. Input data for the rule-based expert system for software version classification.
Source:author

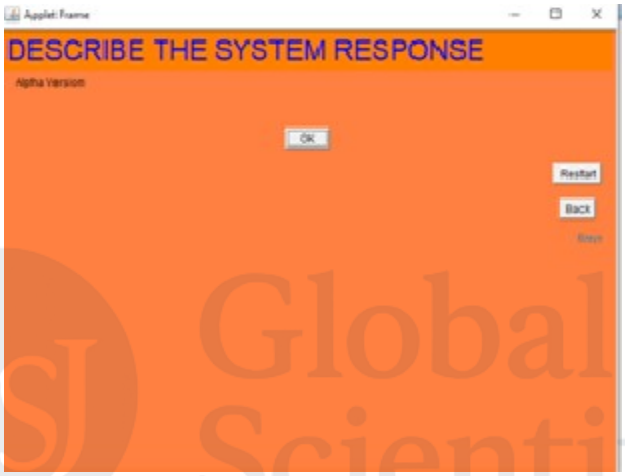

Figure 5. Output the rule-based expert system for software version classification results screen.
Source:author

4.DISCUSSION OF RESULTS

The comparative analysis between the proposed expert system and traditional version control systems and the Concurrent Version System (CVS), allowed the identification of significant differences in operation, applied intelligence, and decision-making capacity.

Unlike these traditional systems, the proposed expert system introduces artificial intelligence mechanisms through IF...THEN type production rules, decision tables, and logical inference implemented in Exsys Corvid. The system not only stores or controls versions but also performs automatic version classification based on predefined criteria.

To perform an evaluation between the old verification method and the expert system, we selected 6 systems with the following 5 steps:

Step 1: Classification Results

Suppose two evaluators classify software as either Safe or Unsafe.

Table 2. Classification results

Case	Evaluator A	Evaluator B
1	Safe	Safe
2	Safe	Safe
3	Safe	Unsafe
4	Unsafe	Unsafe
5	Unsafe	Unsafe
6	Unsafe	Safe

Source: author

Step 2: Build the Agreement Table

Table 3. Classification for groups

	B: Safe	**B: Unsafe**	**Total**
A: Safe	2	1	3
A: Unsafe	1	2	3
Total	3	3	6

Soure: author

Interpretation:

Both said **Safe**: 2 cases

Both said **Unsafe**: 2 cases

Disagreements: 2 cases

Step 3: Calculate Observed Agreement (Po)

Observed agreement is the proportion of cases where the evaluators agreed.

$$P_o = \frac{2+2}{6} \quad P_o = \frac{4}{6} \quad P_o = 0.667$$

So the evaluators agreed on **66.7%** of the cases.

Step 4: Calculate Expected Agreement by Chance (Pe)

Probability both choose **Safe**:

$$\left(\frac{3}{6}\right)\left(\frac{3}{6}\right) = 0.25$$

Probability both choose **Unsafe**:

$$\left(\frac{3}{6}\right)\left(\frac{3}{6}\right) = 0.25$$

Therefore,

$$P_e = 0.25 + 0.25 = 0.50$$

Step 5: Calculate Cohen's Kappa

Cohen's Kappa is:

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

Substituting the values:

$$\kappa = \frac{0.667 - 0.50}{1 - 0.50}$$

$$\kappa = \frac{0.167}{0.50}$$

$$\kappa = 0.333$$

Result

$$\boxed{\kappa = 0.333}$$

Interpretation

A commonly used interpretation scale is:

Table 4. Interpretation of scale

Kappa	Interpretation
< 0.00	Poor agreement
0.00–0.20	Slight agreement
0.21–0.40	Fair agreement
0.41–0.60	Moderate agreement
0.61–0.80	Substantial agreement
0.81–1.00	Almost perfect agreement

Source: Author

Since: $\kappa=0.333$

the agreement between the two evaluators is considered **fair agreement beyond chance**.

Summary

Number of cases: **6**

Observed agreement: **66.7%**

Expected agreement by chance: **50.0%**

Cohen's Kappa: **0.333**

Interpretation: **Fair agreement**

5. CONCLUSION

The development of the proposed expert system demonstrated the feasibility of applying artificial intelligence techniques to software version classification. Through the use of knowledge acquisition methods, production rules, decision tables, and the Exsys Corvid platform, it was possible to represent expert knowledge in a structured manner and automate the classification process of software versions.

Unlike traditional version control systems such as CVS, SVN, and Git, which focus primarily on managing and tracking software changes, the proposed system introduces an intelligent layer capable of analyzing predefined conditions and automatically classifying software releases according to their characteristics.

This capability contributes to more consistent decision-making and reduces the dependence on subjective human evaluation.

The evaluation performed using Cohen's Kappa coefficient showed a fair agreement between classifications, indicating that the expert system can provide reliable support in software version management processes. Although the results are promising, future work may include expanding the knowledge base, incorporating fuzzy logic and machine learning techniques, and testing the system in real-world software development environments to improve classification accuracy and adaptability.

Overall, the research confirms that expert systems can be effectively applied to software engineering problems, providing intelligent support for software version classification and contributing to improved software release management practices.

6. REFERENCES

CollabNet. Apache Subversion Project Documentation. 2000. Available at: https://subversion.apache.org/. Accessed: 31 May 2026.

FEIGENBAUM, E. A. Artificial Intelligence Handbook. Stanford University, 1970.

GUPTA, A.; SINGHAL, R. Expert Systems and Rule-Based Reasoning Techniques. International Journal of Computer Applications, 2017.

HARRISON, M. A. Introduction to Switching and Automata Theory. New York: McGraw-Hill, 1965.

ITIL. Service Transition. London: The Stationery Office, 2013.

LIA, S. Knowledge Representation Using Production Rules in Expert Systems. Journal of Artificial Intelligence Research, 2017.

PANNU, A. Artificial Intelligence and Expert Systems. International Journal of Advanced Research in Computer Science and Software Engineering, v. 5, n. 8, 2015.

TORVALDS, L. Git Distributed Version Control System. 2005. Available at: https://git-scm.com/. Accessed: 31 May 2026.

WAGNER, W. Expert Systems: Principles and Programming. 2017.

DYMOVA, L.; SEVASTJANOV, P.; KACZMAREK, K. Rule-Based Expert Systems and Decision Support. Springer, 2016.

GRUNE, D. Concurrent Versions System (CVS): Historical Development and Applications. 1986.

S. QUINTANA-Amate, P. Bermell-Garcia, and a. Tiwari, ‘Transforming expertise into Knowledge-Based Engineering tools: A survey of knowledge sourcing in the context of engineering design’, Knowledge-Based Syst., 2015.

S. RIBARIĆ, D. Marčetić, and D. S. Vedrina, 'A knowledge-based system for he non-destructive diagnostics of façade isolation using the information fusion of visual and IR images', Expert Syst. Appl., vol. 36, no. 2 PART 2, pp. 3812–3823, 2009.

Landis, J. R., & Koch, G. G. (1977). *The measurement of observer agreement for categorical data*. Biometrics, 33(1), 159–174. https://doi.org/10.2307/2529310

SANYANG, M. L., Sapuan, S. M., Jawaid, M., Ishak, M. R., & Sahari, J. (2015). Development of expert system for biobased polymer material selection. *BioResources, 10*(4), 6041–6059. The study developed an expert system using Exsys Corvid software for material-selection decision support.

LIA, M. (2017). *Knowledge Representation Using Production Rules in Expert Systems*. Journal of Artificial Intelligence Research, 5(2), 45–51.

TORVALDS, L. (2005). *Git Version Control System*. Git Official Website