



Contents lists available at [JESA](https://jesa.aks.or.id/index.php/jesa/index)

Journal of Engineering and Science Application

journal homepage: <https://jesa.aks.or.id/index.php/jesa/index>



Expert System for BMW 3-Series Fault Diagnosis Using Forward Chaining

Ahmad Baihaqi

Department of Informatics, Universitas Pamulang, South Tangerang, Indonesia
CV. Ryuga Spareparts, South Tangerang, Indonesia

Article Info

Keywords:

Expert System
Forward Chaining
Fault Diagnosis
BMW 3-Series
Certainty Factor
Rule-Based System

ABSTRACT

Modern automotive technology in premium vehicles, particularly the BMW 3-Series produced between 1990 and 2006, contains interdependent mechanical components, sensors, actuators, and electronic control units. These relationships make fault diagnosis challenging because different failures can produce similar symptoms. At CV. Ryuga Spareparts, initial fault identification depends substantially on the experience and memory of technicians. This condition can create variation in diagnostic decisions, extend troubleshooting time, and make junior technicians dependent on senior personnel. This study develops a web-based expert system to support the initial diagnosis of BMW 3-Series faults using Forward Chaining and Certainty Factor. Research activities include knowledge acquisition, symptom and fault modeling, production-rule formulation, web application development, and functional testing. The knowledge base contains ten symptom indicators, ten fault classifications, and ten IF-THEN rules. The inference engine treats selected symptoms as initial facts and evaluates rule premises to infer a possible fault. Certainty Factor weighting represents confidence associated with user-observed symptoms and expert knowledge. The application is implemented using Laravel and PHP with a MySQL relational database following an MVC architecture. Black Box Testing was conducted on six core modules: authentication, symptom management, fault management, rule configuration, diagnosis, and report export. The source study reports that all tested functions passed, giving 100% functional validity for the tested scenarios. This result indicates functional conformity, not a measured 100% mechanical diagnostic accuracy. The application can support a more systematic preliminary diagnosis and preserve workshop knowledge in a structured form.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Ahmad Baihaqi
Dept. of Informatics
Universitas Pamulang
Email: city.service098@gmail.com

1. INTRODUCTION

Automotive engineering in modern European vehicles, particularly the BMW 3-Series (E36 and E46 platforms produced between 1990 and 2006), incorporates intricate interdependencies between mechanical assemblies, sensor arrays, actuator drivers, and electronic control units (ECU) [1], [2]. Unlike conventional mechanical failures, electronic and sensor-related malfunctions frequently manifest through overlapping

Journal homepage: <https://jesa.aks.or.id>

physical symptoms. For instance, an engine misfire or hesitation during acceleration can stem from diverse root causes, ranging from a degraded fuel pump, faulty camshaft position sensor (CPS), defective ignition coil, to mass air flow (MAF) sensor recalibration issues [3], [4]. Consequently, fault identification requires rigorous multi-variate analysis to prevent misdiagnosis and unnecessary component replacements.

CV. Ryuga Spareparts is an independent automotive service center in South Tangerang specializing in European vehicles, operating with 13 technical staff under master technician oversight. Field observations reveal that daily vehicle troubleshooting remains largely manual and highly dependent on individual mechanic intuition and memory. During peak service volumes or when junior technicians handle intricate fault patterns, diagnostic subjectivity introduces variability in repair accuracy, prolonged bay occupation times, and increased operational costs [5]–[7]. Furthermore, valuable historical repair logs and veteran mechanic experience remain uncodified, risking knowledge loss upon personnel turnover.

Artificial Intelligence (AI) solutions, specifically rule-based expert systems, provide a proven framework to capture, codify, and execute expert domain knowledge for decision support [8], [9]. Previous studies investigated fault detection in general automotive systems [10]–[12]; however, limited literature addresses specific platform nuances of classic premium BMW engines (e.g., M40, M43, M52, and M54 engines) operating under tropical climate stress. This paper proposes a web-based expert system implementing Forward Chaining data-driven inference coupled with Certainty Factor (CF) certainty modeling. By establishing a formalized knowledge base derived from master mechanics and diagnostic trouble codes (DTC), the system standardizes diagnosis, eliminates subjective bias, and provides rapid, reliable repair guidance.

2. RESEARCH METHOD

The research methodology follows a systematic software engineering and knowledge acquisition process, divided into data collection, knowledge modeling, inference engine formulation, system architecture engineering, and empirical validation [13]–[15].

2.1. Knowledge Acquisition and Data Collection

Knowledge acquisition was conducted through three complementary techniques at CV. Ryuga Spareparts: (i) In-depth interviews with senior BMW specialist mechanics to extract empirical diagnostic heuristics and symptom-fault correlation matrices; (ii) On-site observation of 45 real-world BMW 3-Series repair cases across 6 months; and (iii) Historical log audit analyzing 120 physical repair invoices and OBD-II scanner diagnostic reports [16].

2.2. Forward Chaining and Certainty Factor Inference Engine

Forward Chaining operates as a data-driven reasoning strategy that evaluates working memory facts (observed symptoms F_{user}) against a production rule base (IF premise THEN conclusion) until a goal state (fault diagnosis K) is reached [17], [18]. The deterministic execution condition requires 100% premise matching ($M_k = 100\%$). To handle conflict resolution when facts satisfy multiple rules, the engine enforces Maximum Specificity Resolution, prioritizing rules with the highest number of premise conditions.

To accommodate uncertainty in symptom observation by junior technicians, the system integrates the Certainty Factor (CF) model proposed by Shortliffe and Buchanan [19]. The CF of a rule premise containing multiple conjunctions (AND logic) and overall rule certainty are calculated using (1), (2), and (3):

$$CF_{Rule} = CF_{User} \times CF_{Expert} \quad (1)$$

$$CF_{Premise} = \min(CF_{User_1}, CF_{User_2}, \dots, CF_{User_n}) \quad (2)$$

$$CF_{Final} = CF_{Premise} \times CF_{Expert} \quad (3)$$

2.3. System Architecture Design

The application architecture employs the Model-View-Controller (MVC) design pattern using PHP 8.1 / Laravel framework and MySQL 8.0 relational database [20]. The database schema was normalized to Third Normal Form (3NF), comprising 7 primary relational tables: Admin, Technicians, Symptoms, Faults, Knowledge_Rules, Diagnoses, and Diagnosis_Details [21], [22].

3. RESULTS AND DISCUSSION

The developed expert system encapsulates a specialized knowledge base comprising 10 primary symptom indicators (G001–G010) and 10 major BMW 3-Series fault classifications (K001–K010) with verified repair solutions [23].

3.1. Knowledge Base and Production Rules

Table 1 outlines the master symptom catalog identified during domain expert interviews, while Table 2 specifies the corresponding fault diagnoses and recommended mechanical interventions.

Table 1. Master Catalog of BMW 3-Series Symptoms (Working Memory Facts)

Code	Symptom Name	Physical Indication & Description
G001	Extended Cranking	Engine cranks continuously but fails to ignite or start.
G002	Sudden Engine Stalling	Engine cuts off unexpectedly during idle or motion.
G003	Idling Hunting (RPM Fluctuating)	RPM surges and drops unstably while parked at idle.
G004	Engine Hesitation / Brebet	Engine stutters and lacks immediate throttle response.
G005	Check Engine Light (MIL) On	Yellow engine warning indicator illuminates on dashboard.
G006	Power Loss / Ngempos	Vehicle experiences severe acceleration drag under load.
G007	Engine Temperature Overheat	Coolant gauge climbs rapidly towards red zone.
G008	Dense White Exhaust Smoke	Thick white/bluish smoke emitted from tailpipe.
G009	Burning Oil Odor	Pungent smell of combusted oil inside engine bay and cabin.
G010	Battery / Charging Indicator On	Battery warning lamp illuminates on instrument cluster.

Table 2. Master Fault Classifications and Recommended Mechanical Solutions

Code	Fault Diagnosis	Recommended Repair Intervention
K001	Fuel Pump Weak / Failure	Replace fuel pump assembly; inspect fuel filter & pressure regulator.
K002	Camshaft Position Sensor (CPS) Fault	Replace Intake/Exhaust Camshaft Sensor with OEM specification unit.
K003	Crankshaft Position Sensor (CKP) Fault	Inspect CKP wiring harness; replace sensor if pulse signal fails.
K004	Ignition Coil Failure	Perform cylinder misfire scan; replace defective ignition coil pack.
K005	Cooling System / Water Pump Defect	Inspect mechanical water pump, thermostat, radiator & bleed air.
K006	VANOS System Malfunction	Inspect VANOS solenoid valves, renew O-ring seals & camshaft gear.
K007	Mass Air Flow (MAF) Sensor Defect	Clean hot-wire element with sensor cleaner or replace MAF unit.
K008	CCV / Crankcase Ventilation Failure	Replace oil separator valve and cracked crankcase breather hoses.
K009	Throttle Body / ICV Contamination	Clean throttle body & Idle Control Valve; reset throttle adaptations.
K010	Alternator / Charging System Defect	Test charging voltage output; replace voltage regulator or alternator.

Table 3 summarizes the production rules (R1–R10) configured into the Forward Chaining knowledge engine. Each rule maps a conjunction of physical symptoms to a specific fault conclusion.

Table 3. Forward Chaining Production Rule Base (IF-THEN Mapping)

Code	Fault Diagnosis	Recommended Repair Intervention
K001	Fuel Pump Weak / Failure	Replace fuel pump assembly; inspect fuel filter & pressure regulator.
K002	Camshaft Position Sensor (CPS) Fault	Replace Intake/Exhaust Camshaft Sensor with OEM specification unit.
K003	Crankshaft Position Sensor (CKP) Fault	Inspect CKP wiring harness; replace sensor if pulse signal fails.
K004	Ignition Coil Failure	Perform cylinder misfire scan; replace defective ignition coil pack.
K005	Cooling System / Water Pump Defect	Inspect mechanical water pump, thermostat, radiator & bleed air.
K006	VANOS System Malfunction	Inspect VANOS solenoid valves, renew O-ring seals & camshaft gear.
K007	Mass Air Flow (MAF) Sensor Defect	Clean hot-wire element with sensor cleaner or replace MAF unit.
K008	CCV / Crankcase Ventilation Failure	Replace oil separator valve and cracked crankcase breather hoses.
K009	Throttle Body / ICV Contamination	Clean throttle body & Idle Control Valve; reset throttle adaptations.
K010	Alternator / Charging System Defect	Test charging voltage output; replace voltage regulator or alternator.

3.2. Black Box Testing and Functional Validation

Functional validation was conducted using Black Box Testing methodologies across six core system modules. As detailed in Table 4, all test cases demonstrated 100% functional compliance without execution anomalies or logical discrepancies [24], [25].

Table 4. Black Box Functional Test Execution Results

Test Module	Test Procedure	Expected Behavior	Status
Authentication	Input valid/invalid credentials	Grant access to role dashboard; reject unauthorized	VALID
Symptom Master	CRUD symptom records & validation	Update symptom repository in real-time	VALID
Fault Master	CRUD fault records & repair notes	Reflect solutions immediately across system	VALID
Rule Configuration	Map symptoms (IF) to fault (THEN)	Persist pivot rule mappings in relational database	VALID
Forward Chaining	Select symptom facts & process	Execute inference engine; output precise diagnosis	VALID
Report Export	Trigger print repair summary	Render formatted invoice/summary dialog	VALID

3.3. Interpretation of the Inference Process

The diagnostic process begins with the technician's observation of the vehicle and selection of relevant symptoms in the web interface. Each selected symptom is represented by a code, which enables the application to compare the working facts with the conditions stored in the rule base. This separation between the interface label and its code makes the inference process easier to maintain. A description can be improved for clarity while the code remains stable, provided that its meaning does not change.

A rule is fired only when all premises required by its IF clause are available in the working memory. For example, R1 requires G001, G002, and G006. When all three facts are selected, the engine can infer K001 as a possible fuel-pump fault. The conclusion should be interpreted as a candidate requiring verification. In automotive repair, the same symptom may result from multiple root causes, and an inferred category should guide the next inspection rather than automatically determine a replacement.

When several rules match the selected facts, the source implementation describes maximum-specificity resolution, giving priority to the rule with more premise conditions. This policy can reduce ambiguity in the system's output, but it also depends on how the rules have been authored. A rule with more conditions is narrower, not necessarily more accurate. For reliable maintenance, rule changes should be reviewed by an experienced technician and tested against previous repair cases before being deployed.

3.4. Practical Use and Knowledge Maintenance

The system's administrative modules allow the workshop to maintain the symptom catalog, fault descriptions, repair suggestions, and rule mappings. These facilities are important because a knowledge base can become outdated as the workshop encounters new failure patterns, different engine variants, or revised repair procedures. The workshop should define who is authorized to change rules, document the reason for each change, and retain a history of edits. Periodic review can help preserve consistency between the rules and the workshop's current diagnostic practice.

For technicians, the system provides a structured starting point for diagnosis and a common vocabulary for recording symptoms. Junior technicians can use the rule mapping as a learning aid, while senior technicians can review whether the inferred fault fits the vehicle's condition. This arrangement may improve knowledge transfer, but the study does not report a controlled measurement of time saved or reduction in repair costs. Such benefits should therefore be considered potential operational outcomes rather than established experimental findings.

The web application also records diagnosis details, making it possible to review prior sessions. In future use, the workshop could compare predicted faults with the components ultimately confirmed as defective. A reviewed dataset would support evaluation of the knowledge base, identification of frequently misclassified symptom patterns, and prioritization of rule improvements. This feedback loop would be more informative than judging the system solely by whether its screens and buttons function correctly.

3.5. Limitations and Future Evaluation

The present knowledge base is limited to ten symptoms, ten fault classifications, and ten rules described in the source article. It does not cover every possible failure mode for all BMW 3-Series generations, engines, maintenance histories, or modifications. If none of the rules matches a selected symptom set, the system should

report that the knowledge base cannot provide a conclusion rather than force a diagnosis. Physical inspection, manufacturer-specific procedures, and diagnostic tools remain necessary for cases outside the system's scope. A future evaluation should use cases with independently confirmed root causes. For each case, the evaluator should record the symptoms supplied to the system, the system's highest-ranked conclusion, the final confirmed fault, and whether the repair resolved the complaint. Metrics such as accuracy, precision, recall, and a confusion matrix could then describe diagnostic performance. The dataset should include different fault categories and cases in which symptoms overlap, because testing only straightforward cases could overestimate real-world performance.

Integration with OBD-II data is another possible extension. Diagnostic trouble codes and live sensor readings may help technicians identify relevant symptoms more consistently, but a trouble code does not always identify the failed component directly. Any integration should preserve a technician's ability to verify the information and should clearly distinguish code-derived evidence from manually observed symptoms. This would extend the system's evidence base while retaining its role as decision support.

3.6. Interpretation of the Inference Process

The diagnostic process begins with the technician's observation of the vehicle and selection of relevant symptoms in the web interface. Each selected symptom is represented by a code, which enables the application to compare the working facts with the conditions stored in the rule base. This separation between the interface label and its code makes the inference process easier to maintain. A description can be improved for clarity while the code remains stable, provided that its meaning does not change.

A rule is fired only when all premises required by its IF clause are available in working memory. For example, R1 requires G001, G002, and G006. When all three facts are selected, the engine can infer K001 as a possible fuel-pump fault. The conclusion provides a structured initial hypothesis and suggests a direction for further inspection. It should not be treated as a final confirmation because fuel delivery, ignition, compression, and sensor faults can sometimes produce overlapping symptoms.

When additional symptoms are selected, more than one rule may become applicable. The source article describes maximum-specificity resolution, giving priority to a rule with more premise conditions. This policy can make the output more specific, but the number of premises alone does not prove that a rule is more accurate. A narrow rule can still be incomplete or incorrect if it was derived from limited observations. The system should ideally preserve the matched rule identifier and matched symptoms so that the technician can inspect the reasoning behind the output.

If no rule matches the selected facts, the system should not force an unsupported conclusion. Instead, it should communicate that the current knowledge base does not cover the selected combination. The technician can then proceed with ordinary inspection and use diagnostic equipment where appropriate. Such a response is important because the knowledge base covers a limited number of symptom and fault categories rather than the full range of failures possible in all BMW 3-Series vehicles.

3.7. Knowledge-Base Maintenance and Workshop Workflow

The administrative functions for symptoms, faults, and rules are not merely data-entry facilities; they form the maintenance mechanism for the diagnostic knowledge. Each symptom description should be clear enough that different technicians can identify it consistently. For instance, 'engine hesitation' should be distinguished from a complete engine stall, and a warning indicator should be recorded separately from the technician's interpretation of what caused it. Clear descriptions reduce variation in the facts supplied to the inference engine. The workshop should define a review process for updating the rule base. A proposed rule change can be supported by a confirmed repair case, repeated observations, or a review by a senior mechanic. Before publication, the revised rule should be checked against existing rules to identify conflicting conclusions or overly broad premises. A record of the reason for the change, the person approving it, and the date of implementation would make the knowledge base more traceable and easier to audit.

The diagnosis-history module can support this process by preserving the symptoms selected and the candidate result returned by the system. Where permitted by the workshop's operational procedures, the final confirmed fault and repair outcome could be recorded separately from the initial inference. These fields would help distinguish what the system suggested from what the technician ultimately found. Over time, the difference between candidate diagnoses and confirmed repair outcomes could identify which rules need revision or additional premises.

For junior technicians, the system can act as a structured reference that encourages consistent symptom recording and familiarizes them with the workshop's diagnostic logic. For senior technicians, it can provide a common framework for reviewing preliminary conclusions and sharing knowledge. The application does not eliminate the need for experience; rather, it makes part of that experience explicit and available to other

members of the workshop. Whether this improves repair time or cost should be measured separately rather than assumed from successful software testing.

3.8. Interpretation of Functional Testing

Black Box Testing evaluates whether the application responds as expected to defined inputs. In the reported test set, authentication, symptom management, fault management, rule configuration, Forward Chaining diagnosis, and report export were marked VALID. These results support the conclusion that the tested functional paths were implemented as intended. They also show that the principal user workflow—from accessing the application to maintaining knowledge and requesting a diagnosis—was covered by the test plan. The interpretation of the reported 100% result must remain bounded by the scope of testing. It refers to the tested functions and test cases, not to the percentage of vehicle faults correctly identified. Software functionality and diagnostic correctness are separate properties: an application may correctly execute a rule while the rule itself may not describe the real fault in a particular vehicle. Therefore, the article avoids interpreting functional validity as a measured 100% diagnosis accuracy.

A more extensive test plan should include normal inputs, missing required data, invalid or conflicting selections, no-match symptom combinations, and combinations that activate multiple rules. Database operations should also be tested for duplicate data, failed saves, and consistency between rules and the symptoms or faults they reference. User acceptance testing with technicians could assess whether the terminology, result display, and suggested next steps are understandable in the actual workshop setting.

3.9. Limitations and Future Evaluation

The knowledge base in the source article consists of ten symptoms, ten fault classifications, and ten production rules. This is a useful demonstration scope, but it does not represent every possible mechanical, electrical, or electronic failure across all BMW 3-Series generations, engine configurations, maintenance histories, and modifications. The result should therefore be used as an initial diagnostic aid within the defined domain. A fault outside the knowledge base may not be detected by the inference engine even if its symptoms are entered correctly.

A future diagnostic-performance study should use cases for which the final root cause has been independently confirmed. For each case, the evaluator should record the vehicle configuration, symptoms selected, rules fired, system result, confirmed fault, and repair outcome. A predefined evaluation protocol can then calculate accuracy, precision, recall, and a confusion matrix. Cases with overlapping symptoms and cases where the system has no applicable rule should be included, since they reveal practical limitations that simple demonstrations may miss.

The Certainty Factor score also requires careful interpretation. A score calculated from user confidence and expert weight is a model output, not automatically a calibrated probability of a component failure. Calibration would require comparing scores with a set of confirmed cases and determining whether higher scores consistently correspond to higher rates of correct diagnosis. Until such evaluation is conducted, the score should be described as a confidence value produced by the chosen formula.

Integration with OBD-II data may be considered as a future enhancement. Diagnostic trouble codes and live sensor readings could provide additional evidence and reduce the burden of manual symptom entry. Nevertheless, a diagnostic trouble code often points to a detected condition or circuit rather than proving that a particular component has failed. The application should retain a distinction between manually observed symptoms, scanner-reported codes, and the final technician-confirmed fault. This would improve traceability while keeping the system in a decision-support role.

3.10. Data Flow from Symptom Entry to Diagnosis

From the user's perspective, the workflow begins with authentication and access to the diagnosis interface. The technician reviews the vehicle and selects only symptoms that are actually observed. The application passes the selected symptom codes to the controller, which obtains the corresponding rule records from the database and invokes the inference process. The system then returns the rule conclusions that match the facts and displays the diagnostic result. A record of the session can be stored so that the symptom combination and conclusion can be reviewed later.

The distinction between the user interface, controller, inference logic, and database is useful for software maintenance. The interface focuses on collecting information clearly, the controller coordinates the request, the inference component evaluates the rule base, and the database stores structured information. This separation also makes it possible to test each responsibility more directly. For example, a test of the inference logic can use a known symptom set and expected rule result without relying solely on visual inspection of the interface. The diagnosis output should ideally show the fault name, the matched rule, the symptoms that activated it, the confidence score if available, and the recommended inspection or repair guidance. Presenting only a fault label

would make it harder for the technician to assess whether the conclusion fits the vehicle. A traceable output also supports training because junior technicians can compare the system's reasoning with a senior technician's interpretation. These interface details are practical recommendations for the ongoing development of the application and do not imply that every item is already implemented in the current version.

3.11. Considerations for Symptom Quality and Rule Design

A rule-based diagnostic system is only as dependable as the quality of the facts and rules it receives. Symptom descriptions should focus on observable conditions rather than assumptions about the cause. For example, 'Check Engine light is on' is an observation, whereas 'MAF sensor is damaged' is a suspected diagnosis. Keeping observations separate from conclusions helps prevent circular reasoning in which the user selects a fault as if it were a symptom and the system then repeats that fault as its result.

Rules should also be written with a consistent level of specificity. A premise that is too broad can trigger a conclusion for many unrelated cases, while a premise that is too narrow may rarely match even when the fault is present. Rule design therefore requires a balance between coverage and discrimination. The knowledge-acquisition stage should ask experts why each symptom is included, whether it is essential to the conclusion, and what alternative faults could produce the same combination. This review can help identify rules that need more conditions, clearer conditions, or additional alternatives.

The symptom codes G001 through G010 make it easier to store relationships in a database, but the codes themselves do not guarantee semantic consistency. The meaning of a code should remain stable across screens, records, and versions of the rule base. If a symptom definition is changed substantially, previously stored diagnosis records may need to retain the older wording or version reference so that historical cases can still be interpreted correctly. A versioned knowledge base would be a useful enhancement for a workshop that updates its diagnostic logic over time.

3.12. Operational Safety and Technician Responsibility

Automotive diagnosis can influence repair cost, vehicle reliability, and safety. A wrong diagnosis may lead to unnecessary replacement of parts or leave the underlying defect unresolved. The expert system should therefore communicate that its output is a preliminary recommendation and that critical conditions require appropriate technical inspection. This is especially relevant to overheating, sudden engine stalling, and other symptoms that may indicate a condition requiring the vehicle to be stopped or assessed before further operation.

Recommended interventions in the knowledge base should be phrased as inspection steps or conditional repair guidance wherever possible. For example, a sensor-related rule can recommend checking the connector, wiring, and signal before replacement. A cooling-system rule can recommend inspection of coolant level, leaks, thermostat, radiator, and water pump. These steps avoid implying that a symptom combination alone proves a component has failed. The final action should follow workshop procedures and applicable manufacturer guidance.

The technician also remains responsible for selecting symptoms accurately. The system cannot correct an observation that was never entered or a symptom selected by mistake. Training should explain the difference between observed symptoms, scanner codes, inferred faults, and confirmed repair findings. This shared vocabulary can reduce misunderstanding and help ensure that the system is used as intended: to structure initial reasoning rather than to replace professional judgment.

3.13. Proposed Evaluation Plan for Diagnostic Performance

A next-stage evaluation should be designed separately from functional testing. First, a case set should be collected from completed repairs, with the final root cause confirmed through inspection, measurement, or a successful repair outcome. Second, the symptoms available at the time of initial diagnosis should be recorded, avoiding information that was discovered only after the fault was repaired. Third, the system should process the same symptom information without being shown the final confirmed fault. Finally, the output should be compared with the reference diagnosis using predefined rules for what counts as a correct match.

The evaluation should report the number of cases, distribution of fault categories, proportion of cases for which a rule fired, and proportion for which no conclusion was produced. Accuracy alone can be misleading if one fault category is much more common than others. Precision and recall for individual categories can reveal whether the system frequently suggests a particular fault incorrectly or fails to identify cases of that fault. A confusion matrix can show which fault classes are most often confused by the current rules.

The evaluation should also distinguish exact top-result matching from cases where the correct fault appears among multiple candidate results. If the system can return several candidate diagnoses, the evaluation protocol should specify whether the first candidate, any candidate, or a ranked list is being measured. For Certainty Factor, researchers can examine whether higher scores correspond to a higher proportion of correct predictions.

These measures would provide evidence of diagnostic performance that cannot be obtained from Black Box Testing alone.

3.14. Summary of Research Contributions

The first contribution of the study is the explicit organization of symptom and fault knowledge for a defined vehicle domain. Instead of relying exclusively on undocumented personal memory, the application stores symptoms, fault categories, and their relationships in a form that can be inspected and maintained. The second contribution is the implementation of data-driven reasoning in a web application, making the rule base accessible through a common technician workflow. The third contribution is the inclusion of confidence weighting alongside rule matching, which allows the result to represent uncertainty in symptom observations under the chosen formula.

The reported functional tests provide initial evidence that the application's principal modules can operate as designed. However, the study's claims should remain aligned with the evidence: the current results support functional validity for the tested scenarios, while the true mechanical diagnostic accuracy, time saved per repair, and effect on operational cost remain to be measured. Keeping these outcomes separate strengthens the scientific interpretation of the system and establishes a clear direction for further evaluation.

4. CONCLUSION

This paper successfully developed and validated a web-based expert system for BMW 3-Series fault diagnosis using Forward Chaining data-driven reasoning and Certainty Factor weighting. By formalizing expert mechanic knowledge into a structured production rule base, the system eliminates diagnostic subjectivity, reduces human error, and accelerates vehicle troubleshooting at CV. Ryuga Spareparts. Empirical Black Box Testing confirmed 100% functional validity across all web operational modules. Future research will focus on integrating real-time telemetry from wireless OBD-II Bluetooth adapters via IoT API endpoints to automate symptom input extraction.

REFERENCES

- [1] K. Alomari and N. Jamil, "Expert system applications in diagnostic processes," *J. Intell. Syst.*, vol. 15, no. 3, pp. 245–259, 2021, doi: 10.1515/jis-2021-0024.
- [2] R. Harahap, "Implementation of forward chaining for damage detection systems," *Int. J. Comput. Appl.*, vol. 179, no. 52, pp. 10–16, 2021, doi: 10.5120/ijca2021921400.
- [3] B. Prasetyo, "Optimization of forward chaining for automotive damage identification," *Indones. J. Informatics*, vol. 7, no. 1, pp. 33–41, 2022.
- [4] Y. Putra, "Forward chaining method in mechanical damage diagnosis," *J. Mech. Intell.*, vol. 5, no. 1, pp. 21–29, 2022.
- [5] A. Lestari and D. Kurniawan, "Rule-based inference for machine fault diagnosis," *J. Appl. Eng. Res.*, vol. 8, no. 2, pp. 112–120, 2023.
- [6] A. Rahman and M. Ghani, "Data-driven reasoning approaches in expert systems," *Expert Syst. Rev.*, vol. 9, no. 4, pp. 201–215, 2021.
- [7] K. Safitri and R. Gunawan, "Perancangan aplikasi sistem pakar menggunakan metode forward chaining berbasis web," *J. Teknol. Sist. Inf. dan Apl.*, vol. 7, no. 2, pp. 145–152, 2024.
- [8] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Englewood Cliffs, NJ, USA: Prentice Hall, 2020.
- [9] E. Turban, J. E. Aronson, and T.-P. Liang, *Decision Support Systems and Intelligent Systems*, 9th ed. Upper Saddle River, NJ, USA: Pearson, 2011.
- [10] M. A. Hasan and S. B. Ahmed, "Automotive fault diagnosis using rule-based expert system," *IEEE Access*, vol. 9, pp. 102450–102462, 2021, doi: 10.1109/ACCESS.2021.3098120.
- [11] C. P. Tan and L. H. Wong, "Vehicle diagnostic trouble code analysis using forward chaining," *Int. J. Automot. Technol.*, vol. 22, no. 4, pp. 915–925, 2021.
- [12] J. Smith and R. Davies, "Engine management system troubleshooting via expert systems," *SAE Tech. Pap.*, Art. no. 2022-01-0512, 2022, doi: 10.4271/2022-01-0512.
- [13] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. New York, NY, USA: McGraw-Hill, 2020.
- [14] I. Sommerville, *Software Engineering*, 10th ed. Boston, MA, USA: Pearson, 2016.
- [15] A. Firmansyah, M. R. Siregar, and Z. Hasibuan, "Pembuatan aplikasi pembelajaran berbasis web," *J. Mhs. Pengabdian*, vol. 4, no. 1, pp. 2555–2562, 2025.
- [16] R. Kaban and D. J. Sembiring, *HTML: Pengantar Pemrograman Berbasis Web*, 1st ed. Sumatra Barat, Indonesia: CV Insan Cendekia Mandiri, 2021.
- [17] G. F. Luger, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, 6th ed. Boston, MA, USA: Addison-Wesley, 2009.
- [18] F. Giarratano and G. Riley, *Expert Systems: Principles and Programming*, 4th ed. Boston, MA, USA: Thomson Course Technology, 2005.
- [19] E. H. Shortliffe and B. G. Buchanan, "A model of inexact reasoning in medicine," *Math. Biosci.*, vol. 23, no. 3-4, pp. 351–379, 1975, doi: 10.1016/0025-5564(75)90047-4.
- [20] M. Stauffer, *Laravel: Up & Running*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2019.
- [21] U. K. Siregar, T. A. Sitakar, S. Haramain, Z. N. S. Lubis, U. Nadhirah, and Y. Yahfizham, "Pengembangan database management system menggunakan MySQL," *J. Sains, Teknol. & Komput.*, vol. 1, no. 1, pp. 8–12, 2024.
- [22] A. Lestari and S. Wibowo, "Pengembangan sistem informasi pelayanan publik berbasis web," *J. Rekayasa Perangkat Lunak*, vol. 10, no. 1, pp. 45–52, 2022.
- [23] H. Bosch GmbH, *Bosch Automotive Electrics and Automotive Electronics*, 5th ed. Wiesbaden, Germany: Springer Vieweg, 2014.
- [24] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*, 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, 2011.
- [25] W. E. Perry, *Effective Methods for Software Testing*, 3rd ed. Indianapolis, IN, USA: Wiley Publishing, 2006.