CHALMERS
UNIVERSITY OF TECHNOLOGY

Agentic Systems for Failure Attribution and Root Cause Analysis

Master's thesis 2026

JIAYI WANG

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Agentic Systems for Failure Attribution and Root Cause Analysis

Jiayi Wang

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Materials Science
Chalmers University of Technology
Gothenburg, Sweden 2026

Agentic Systems for Failure Attribution and Root Cause Analysis
Jiayi Wang

Supervisor Siyuan Chen, Department of Industrial and Materials Science
Examiner Ebru Turanoglu Bekar, Department of Industrial and Materials Science

Master's thesis 2026
Department of Industrial and Materials Science
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Agentic Systems for Failure Attribution and Root Cause Analysis
Jiayi Wang
Department of Industrial and Materials Science
Chalmers University of Technology

Abstract

Automotive warranty cases are valuable for quality follow-up, but they are not written as root-cause reports. A record can mix customer symptoms, workshop actions, replaced assemblies, parts context, and partial causal clues. The recorded repair scope can describe what was done in service while still hiding the component that most likely initiated the failure. Human reviewers can recover that meaning, but the work is slow and depends on engineering experience.

This thesis aims to develop a planner-guided agentic system for failure attribution and root cause analysis in warranty cases. Instead of treating repair text as a single classification input, the system separates text normalization, signal extraction, hypothesis formation, optional critique, dictionary-constrained mapping, historical-case retrieval, and final judgement. The intermediate outputs are stored with the final decision, so a reviewer can see the fields, comparison cases, and constraints behind the selected label.

The evaluation combines manual review analysis, a direct single-pass baseline, mechanism variants, field-level checks, retrieval checks, knowledge-use analysis, and final-label scoring. Manual review finds 333 CHG-related records inside a 628-record engine-exchange candidate set, showing how component-level failure patterns can sit inside a broad repair outcome. On the 357-record positive benchmark, target-label recovery increases from 29.4% with the direct single-pass baseline to 84.3% with the planner-guided workflow, while other non-target outputs fall from 67.2% to 7.6%. On the mixed 628-record basis, the workflow reaches 85.4% accuracy and 87.3% F1 for the target label. ANN retrieval preserves the exact top result in 99.4% of the current retrieval index, and knowledge entries are used in 490 of 628 completed records.

The results indicate that an agentic workflow can improve failed-component recovery and make the result easier to inspect than a single-pass request. It does not replace engineering judgement or independently prove physical root cause. Its role is to organize noise from workshops into a standardized label, structured support, and a shorter path for expert review.

Keywords artificial intelligence, industrial diagnostics, warranty text, multi-agent systems, failure diagnosis, human-in-the-loop review.

Acknowledgements

I would like to express my sincere gratitude to my examiner, Ebru Turanoglu Bekar, for her guidance and support throughout this thesis process. She helped me see beyond isolated technical details and understand the bigger picture of the project. When I was stuck on specific issues, she guided me to think about why they mattered, how they were connected to the project background, and how the results could be relevant in an industrial context. Her guidance encouraged me to think more critically about my work and also helped me improve my academic skills, communication, and professional development. I am truly grateful for her patient, thoughtful, and inspiring support.

I am also very grateful to my supervisor, Siyuan Chen, for his continuous guidance and encouragement during the project. His advice helped me improve both my technical understanding and the way I presented my work. In our meetings, he often helped me identify the key points, organize my ideas more clearly, and strengthen the logic of the thesis. His feedback was especially helpful in improving the structure, methodology description, and presentation of the results.

This thesis was conducted as part of the AIMOps project, which provided the industrial context and motivation for this work. I would also like to thank Mirko Mortensen Jovetic, Stefan Fahlgren, and Håkan Swedenborg for taking the time to discuss the project with me and share their feedback. Their suggestions helped me understand the problem from an industrial perspective and improve this thesis.

Finally, I would like to thank my friends for their support and encouragement. Their discussions gave me useful inspiration, and their care and help in daily life made this thesis journey easier and more meaningful. I am grateful for their companionship and kindness throughout this period.

Jiayi Wang, Gothenburg, 2026

List of Acronyms

Acc.	Accuracy
CHG	Cylinder head gasket
CMMS	Computerised Maintenance Management System
CSV	Comma-separated values
EM	Exact Match
F1	F1-score
FR	Fallback Rate
KB	Knowledge base
KSDR	Knowledge-Supported Decision Rate
LLM	Large language model
MWO	Maintenance work order
NLP	Natural language processing
P	Precision
R	Recall
RAG	Retrieval-augmented generation
T1P	Top-1 Preservation
T5O	Top-5 Overlap
TP	Token Precision

Nomenclature

k	Number of retrieved knowledge-base entries or similar cases.
s	Confidence or similarity score, depending on context.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Case background

Automotive warranty cases are valuable for quality follow-up because they connect repair events with workshop observations, parts use, claim categories, and recurring product problems. The same records can also be misleading when a case is treated as a direct statement of root cause. Warranty systems often record what was repaired or replaced, while engineering review needs to know which component most likely initiated the failure.

This difference between repair scope and failure attribution is central to the problem. A broad assembly-level repair label can be correct from a service perspective, but still hide the smaller component pattern that quality analysis needs. If such cases are counted only by the broad repair label, recurring component-level failures can be diluted inside a larger repair category, and review effort can be directed towards the outcome of the repair rather than the originating component.

Reliability-data standards treat service and claim records as operational evidence, while work-order studies show that such records need substantial interpretation before they can support reliability or quality analysis [1, 2, 3, 4]. Warranty analytics has long been used to learn from claims and repair histories, but useful signals are often spread across structured codes, parts lists, and short technician-written text [5, 68]. Recent manufacturing and maintenance-text studies show the same pattern. Operational text contains fault-related information, but the useful signal is rarely stored as a clean component-level judgement [7, 8].

Data used later in the evaluation shows this gap in a concrete setting. Cases grouped under a broad repair outcome can still require expert judgement to separate confirmed component-level failures from likely or ambiguous ones. The field that is easiest to count is not always the field that carries the attribution needed for quality follow-up. A reviewer has to decide whether the case points to a failed component, a broad repair action, or a symptom that still needs interpretation.

1.2 Problem description

Warranty cases rarely read as complete diagnostic statements. A single record can contain the customer symptom, the workshop action, replaced parts, service codes, and partial causal clues. Automotive diagnosis analytics describes this problem in repair verbatim and warranty claim records, where fault-diagnosis knowledge is

present but remains buried in unstructured text [9]. Industrial service-text studies report the same difficulty for technical language, abbreviations, sentence fragments, and manually assigned metadata [10, 11]. Automotive lifecycle research treats warranty records as heterogeneous data sources from which natural language processing can extract reliability-related signals for engineering review [12].

The practical difficulty is the gap between the field that is easy to record and the judgement that root cause analysis needs. Warranty systems often record the repair outcome, while quality follow-up needs the component that most likely initiated the failure. A replaced part can be a repair consequential damage rather than the failure source. A broad repair label can hide a more specific component-level pattern. Prior automotive text-mining work uses domain knowledge to relate symptoms, parts, and repair actions [9], while recent manufacturing-text research emphasizes dictionary-based standardisation of free-form failure descriptions [8]. These studies support the same lesson for warranty-case analysis. The final label has to be controlled, but the evidence behind that label cannot be discarded.

Failure attribution and root cause analysis are treated here as more than a text-classification task. Classification can help predict or correct structured service metadata [10], but a disputed warranty case also needs to show whether the selected label came from a symptom, a repair action, a parts clue, a similar historical case, or a domain rule. Current AI risk-management guidance emphasizes validity, reliability, transparency, interpretability, and accountability [13], and smart-manufacturing reviews argue that human involvement remains central when autonomous or semi-autonomous systems support manufacturing decisions [14]. In this setting, a useful method has to produce a constrained label and leave enough intermediate evidence for an engineer to check the decision.

1.3 Purpose and aim

A planner-guided agentic system is investigated as support for failure attribution and root cause analysis from warranty cases whose recorded repair scope can be broader than the originating failure. The goal is not to automate engineering judgement end to end. The goal is to make the path from warranty text to failed-component label easier to check.

The link between review difficulty, system design, and diagnostic effectiveness sets the scope of the work. Warranty cases become difficult when symptoms, actions, parts evidence, and broad repair labels point in different directions. The proposed system separates text normalization, signal extraction, hypothesis formation, optional critique, label mapping, retrieval, and final judgement instead of asking one model call to jump from raw text to a final label. Its value is judged by whether it identifies failed components while leaving the reasoning path useful for root cause analysis. Historical-case retrieval adds comparison cases, and knowledge-based constraints keep the final output inside the allowed label space.

1.4 Research questions

The research questions move from the review problem, to the proposed system design, to the measured diagnostic outcome.

RQ1. To what extent do warranty cases make failure attribution and root cause analysis difficult for human review?

RQ2. What advantages does the agentic system provide compared with a direct single-pass method?

RQ3. How effective is the agentic system in identifying failed components and supporting root cause analysis?

1.5 Delimitations

The empirical work is limited to the available automotive warranty and repair-related data in the local project workspace. The output space is defined by the standardized failed-component labels and root-cause groupings available for this setting. Results should be interpreted within this data and label space.

The work does not build a complete vehicle diagnostic system. Sensor streams, physical simulations, workshop inspection data, and full service histories are outside the scope, except for the repair text and component context fields used by the implementation. Extension to other warranty or service settings requires additional datasets, domain-specific label dictionaries, and reviewer validation.

1.6 Thesis outline

Chapter 1 introduces the warranty-case background, defines the attribution problem, states the purpose and research questions, and clarifies the delimitations. Chapter 2 reviews the background needed to understand warranty cases as diagnostic evidence, large-language-model interpretation of repair text, historical-case comparison, and agentic harnesses. Chapter 3 presents the methodology, including the planner-guided workflow, shared diagnostic state, retrieval branch, and knowledge constraints. Chapter 4 reports the evaluation through human-review difficulty, direct-baseline comparison, and failed-component prediction results. Chapter 5 discusses the findings, limitations, and future work. Chapter 6 summarizes the main conclusions.

2

Theoretical background

Warranty-case review starts from a practical mismatch. A case contains useful repair information, but symptoms, actions, parts, and claim labels are often mixed in the same operational record. A support system has to separate these roles before it can help with failed-component identification and root cause analysis. This background connects warranty-case records, language-model interpretation, historical comparison, label control, and the agentic harness used later in the method.

2.1 Warranty cases as diagnostic work records

Warranty cases are produced during service and claim handling before they become research data. They are useful because they connect a customer complaint, workshop observations, repair actions, parts use, claim categories, and quality follow-up. They are also difficult because these elements are not written as a clean root-cause report. The same case can describe how the issue appeared, what was inspected, what was replaced, and how the claim was recorded. A reviewer has to separate these roles before deciding which component best explains the failure.

2.1.1 Operational structure of warranty cases

Service and claim records belong to a wider reliability-data environment where collection, exchange, processing, and presentation affect later diagnosis [1, 2]. Work-order and repair-text studies treat such records as valuable field-generated industrial evidence with inconsistent wording, missing details, and local workshop terminology rather than clean descriptions of a physical failure [3, 4, 7, 10, 15, 16]. Bekar et al. make a related point in predictive maintenance. Useful diagnostic information often appears only after data has been pre-processed, structured, and interpreted with domain knowledge [17]. The same principle applies to warranty cases, even though the evidence here is dominated by repair text and parts context rather than sensor streams.

Maintenance work order records are commonly managed in a Computerised Maintenance Management System (CMMS). Woods et al. describe a typical maintenance work order as a combination of structured fields for asset identity, timing, labour, materials, and an unstructured work description [18]. Table 2.1 adapts this CMMS-oriented view to show the type of operational fields that also shape warranty-case analysis. The table is not used as a complete data schema. Its purpose is to show

why one warranty case has to be read as several weak signals rather than as one direct failure statement.

Table 2.1: Operational fields used in warranty-case analysis, adapted from Woods et al. [18].

Field group	Description	Field type
Functional location	Asset or system location where maintenance is performed	Code
Work description	Free-text description of the problem or work to be done	Text
Work order number	Unique identifier for the service or warranty case	Numeric
Work order type	Code indicating the maintenance category, such as corrective or preventive work	Code
Start and end dates	Metadata for the creation, scheduling, and execution of the work order	Date
Labour effort	Planned or actual labour effort	Numeric
Parts or materials used	Parts or materials used during the maintenance activity	Numeric

2.1.2 Warranty cases feed quality follow-up

Automotive warranty data connects a repair event with product quality, supplier follow-up, dealer processes, and service documentation. Warranty analytics helps organizations detect patterns that are difficult to see from individual workshop cases alone [5, 68, 11, 12, 19]. This makes warranty data attractive for quality follow-up, but the same operational origin also limits direct automation. Coded classifications, parts lists, service information, and repair-order narratives encode different viewpoints on the same event. Treating these fields as if they all stated one diagnostic fact makes the attribution problem appear simpler than it is.

Industrial maintenance decision support also has to fit the people who use it. Chen et al. show in a digital-twin maintenance study that stakeholder requirements concern supported decisions, implementation barriers, and the actions a system can help maintenance practitioners take [20]. This point carries over to warranty cases. A support method is useful only if it reduces review burden without hiding the reasons behind a label.

Repair-order text is especially important for human review because it often contains the only explanation of what happened in the workshop. It is compressed, action-oriented, multilingual in practice, and shaped by workshop routines rather than by a research protocol [15, 21, 22]. A reviewer must often infer whether a part mention is a suspected cause, an inspected item, a replacement action, or a consequence of a different failure. This is the practical basis for the motivation of the study. Warranty cases contain useful information, but they place a real interpretation burden on reviewers.

2.1.3 Failure attribution supports root cause analysis

Failure attribution maps a warranty case to the component label that best represents the failed part indicated by the available case information. Root cause analysis then uses this component-level attribution as one step towards understanding why the failure occurred. The two ideas are related but not identical. A failed-component label gives a stable review target. Root cause analysis asks how that component-level decision fits the wider symptom, repair, and quality context.

Earlier automotive diagnosis research supports this framing. Rajpathak's study describes field repair data as a combination of diagnostic events, repair labour codes, and technician verbatim [9]. Such verbatim can contain a fault description, a part mention, and a corrective action, while the diagnostic knowledge remains buried in field-reported and unstructured text. Rajpathak also notes that repair records include service actions that are not directly tied to the underlying fault. That distinction matters because a component mentioned during repair is not automatically the component that should receive the final attribution.

Workshop repair texts show the same ambiguity. A customer symptom code describes how the issue appeared, not necessarily the failed component. A cause code gives a broad claim classification. A parts list records material used during the job, including service items and exchanged assemblies that are not all primary failures. A failing-part description is a canonical warranty label, but it can be coarser than the physical mechanism. Attribution cannot rely on a single component term or one structured field. Each case has to be read as a combination of symptoms, work actions, part usage, and claim classification.

The literature shows a gap between recorded service activity and component-level attribution. Prior work provides warranty-analysis motivation, text-mining tools, and data-structuring approaches [8, 9, 23]. A narrower problem remains. A single field-reported warranty case has to be converted into a valid failed-component label without losing the case details needed for inspection.

2.2 Structuring repair text with language models

Raw warranty text does not directly provide the structure needed for attribution. Language models are relevant because they can interpret short technical phrases, local abbreviations, translated wording, and mixed symptom-action descriptions. Their role is not to act as autonomous diagnostic authority. Their role is to help transform unstructured repair text into intermediate fields that a reviewer can check.

2.2.1 From text processing to evidence construction

Earlier repair-text approaches often relied on rules, dictionaries, or supervised classifiers that worked well when the target field was clear and the vocabulary was stable. Large language models add the ability to interpret short repair narratives in context and organize intermediate evidence from text that was not written as a structured input. This capability is relevant to warranty cases because useful information is often distributed across customer complaints, technician observations, parts lists,

and coded fields [8, 10, 15]. The broader language-model capability builds on transformer architectures, pre-training, and text-to-text modeling [24, 25, 26, 27]. In adjacent automotive quality work, Chen et al. compare unsupervised anomaly detection models for sheet-metal glue-line inspection and frame false positives, limited data, and operational efficiency as central evaluation concerns [28]. Warranty text is a different modality, but the same quality-follow-up problem appears. Model output has to be tied to evidence that engineers can inspect.

For failure attribution, the useful output is not a fluent paragraph. The useful output is a structured account of symptoms, findings, repair actions, component clues, and candidate failure concepts. Field-level structure matters because a reviewer can inspect whether the model extracted the right component, whether it mixed a symptom with a cause, or whether it over-expanded a repair action. This connects directly to the later evaluation. Exact label correctness is important, but intermediate evidence quality also has to be checked.

Several simpler approaches leave important failure modes exposed. A direct classifier can learn useful label patterns, but it tends to compress symptoms, service actions, and component mentions into one answer before their roles have been separated. A rule-based dictionary system gives stable terminology, but it struggles when the same failure is written in different languages, abbreviations, or workshop styles. A retrieval-only pipeline can find similar cases, but similar wording does not prove the same component-level attribution. These limits do not make the approaches irrelevant. They explain why the method later combines text structuring, retrieval, mapping, and final checking instead of relying on one of them alone.

2.2.2 Contextual interpretation of repair evidence

Repair records rarely state a failure as one complete diagnostic sentence. A short phrase gains meaning from its position in the narrative, the surrounding action, and the fields attached to the case. In-context reasoning lets a model use task instructions, case evidence, terminology, label definitions, and selected examples at inference time [25, 29, 30, 31].

Technical repair text makes this contextual reading necessary. The same component mention can refer to a suspected part, an inspected part, a replaced part, consequential damage, or the component finally treated as responsible for the repair outcome. A direct classifier can compress these roles into one label too early. Contextual interpretation helps separate what the case says from what the final attribution claims.

Contextual interpretation has an interpretive value before it has a decisional value. It can produce a clearer account of symptoms, actions, inspected parts, replaced parts, and candidate attribution cues. That account still requires historical comparison, label constraints, and final checking before it becomes a failed-component label.

2.2.3 Unsupported outputs require human accountability

Large language model (LLM)-based systems can produce plausible but unsupported conclusions, especially when the input text is incomplete, translated, ambiguous,

or dominated by repair actions rather than explicit failure statements. In failure attribution, this is not only a generic hallucination issue. A model can over-select a prominent part name, infer a root-cause pattern without enough case-specific evidence, or confuse a replacement action with the failed component [32, 33, 34].

These risks explain why human review remains central. AI risk-management guidance emphasizes validity, reliability, transparency, interpretability, and accountability [13]. Human-in-the-loop manufacturing research also stresses that systems used in industrial decision support need to remain accountable to people who understand the operational context [14, 35]. A diagnostic system that always forces a specific label can appear complete while hiding weak evidence. A useful system should preserve uncertainty and make the evidence path inspectable.

This literature sets a practical boundary for LLM use. Language models can normalize technical phrasing and organize intermediate fields. They still need historical comparison, label constraints, and checks against unsupported conclusions. Fluent generation alone is not diagnosis.

2.3 Historical cases narrow the comparison problem

A structured field extraction still leaves an important question open. The question is whether the evidence in one case is strong enough to support a final label. Historical comparison and controlled output address this problem in different ways. Retrieval adds evidence beyond the current case, while constraints keep the final answer inside a label space that reviewers and evaluation procedures can use.

2.3.1 Case-based reasoning in warranty cases

Case-based reasoning provides the theoretical motivation for historical comparison. A new problem is interpreted through earlier cases that contain related problem descriptions, solutions, and outcomes [36]. Warranty analysis often follows this logic in practice. One repair note can be short or ambiguous, but a group of similar historical cases can show whether the same symptom-action pattern has previously been associated with a particular part, repair action, or quality issue.

Retrieved cases provide comparison, not proof. A retrieved case supports a candidate label when its symptoms, parts context, and repair narrative are consistent with the current case. It weakens a candidate when the closest cases point towards a different part or when similarity comes from superficial wording rather than diagnostic content. Retrieval value depends on how the returned cases are interpreted, not only on their similarity scores.

2.3.2 Similarity ranking supports review efficiency

Similarity-based retrieval represents each case so that comparable cases can be ranked against the current case. Representation can be dense, as in embedding-based semantic search, or more explicit, as in token-based comparison over repair

text and parts lists. For warranty cases, both views are useful because similar technical situations are often described with different wording, while parts context gives a more concrete trace of what was handled during the job [22, 37, 38, 39, 40].

Repair records make the limits of similarity especially important. Two cases can share symptom wording, inspection wording, or replacement wording while still involving different primary components. Conversely, two cases with the same failed component can appear in different languages, use different local abbreviations, or emphasize different repair stages. Parts-list overlap has the same ambiguity. A shared service item can reflect ordinary repair practice rather than the primary failure.

Retrieval-augmented generation follows a related idea in natural language processing by combining model generation with non-parametric access to external documents [41, 42, 43, 44]. For diagnostic attribution, the value is not simply that more text enters the prompt. The value is that the supporting cases remain visible. A reviewer can inspect which historical cases influenced a candidate label and whether their similarity is diagnostically meaningful.

Efficiency also matters for human review. A reviewer should not have to scan an unranked archive of previous repairs when only a small number of cases are diagnostically useful. Case-based diagnostic support has been used to narrow the review task and reduce diagnostic effort [45]. Related work on online maintenance prioritization also treats ranking as a way to improve decision-making efficiency when service choices have to be made quickly [46]. Here, ranking supports review by surfacing a limited set of relevant comparison cases before final judgement.

2.3.3 Approximate retrieval scales the case base

Approximate nearest-neighbour search is commonly used to retrieve similar cases efficiently from a larger case base. In warranty-case analysis, retrieval is a practical way to surface potentially relevant historical evidence rather than an automatic proof that two cases share the same failed component [47, 48, 49].

Scale creates the need for approximate search. Exact comparison against every historical case becomes inefficient as the case base grows, especially when each case has to be compared across several text-derived and parts-derived signals. Approximate methods first narrow the candidate set and then rank the remaining records in more detail. Locality-sensitive hashing and graph-based search are common ways to make this search practical [47, 48, 50, 51]. This trade-off is acceptable when retrieval is used to surface evidence for review rather than to make the final decision alone.

2.3.4 Knowledge constraints keep outputs controlled

Controlled output is a methodological requirement in warranty-case analysis. Without a controlled dictionary, a model can produce synonyms, overly detailed component names, or labels that sound meaningful but are invalid for the warranty process. Stable labels are also needed for batch evaluation, quality follow-up, and later review.

Knowledge constraints help connect language interpretation with the industrial label space. Candidate hypotheses can stay close to the language of repair evidence, while mapping rules translate them into canonical failed-component labels. Final judgement can then choose among controlled candidates instead of inventing a new label. This explains one concrete advantage over a direct model request because the system can separate interpretation from label-space control.

2.4 Agentic workflows organize diagnostic reasoning

Historical comparison and controlled labels are useful only when they are organized inside a workflow that preserves the diagnostic path. Work with LLMs in technical settings has moved beyond writing a single good instruction. Prompt design controls how a model is asked to perform a task, while context design controls what case evidence, label definitions, examples, and retrieved records the model sees. An agentic harness adds an execution structure around the model through shared state, routing, validation, retrieval, logging, and constraints. This view is consistent with work on augmented language models and compound AI systems, where model calls are combined with external tools, retrieval, stateful control, and system-level orchestration [31, 53]. Figure 2.1 summarizes this progression.

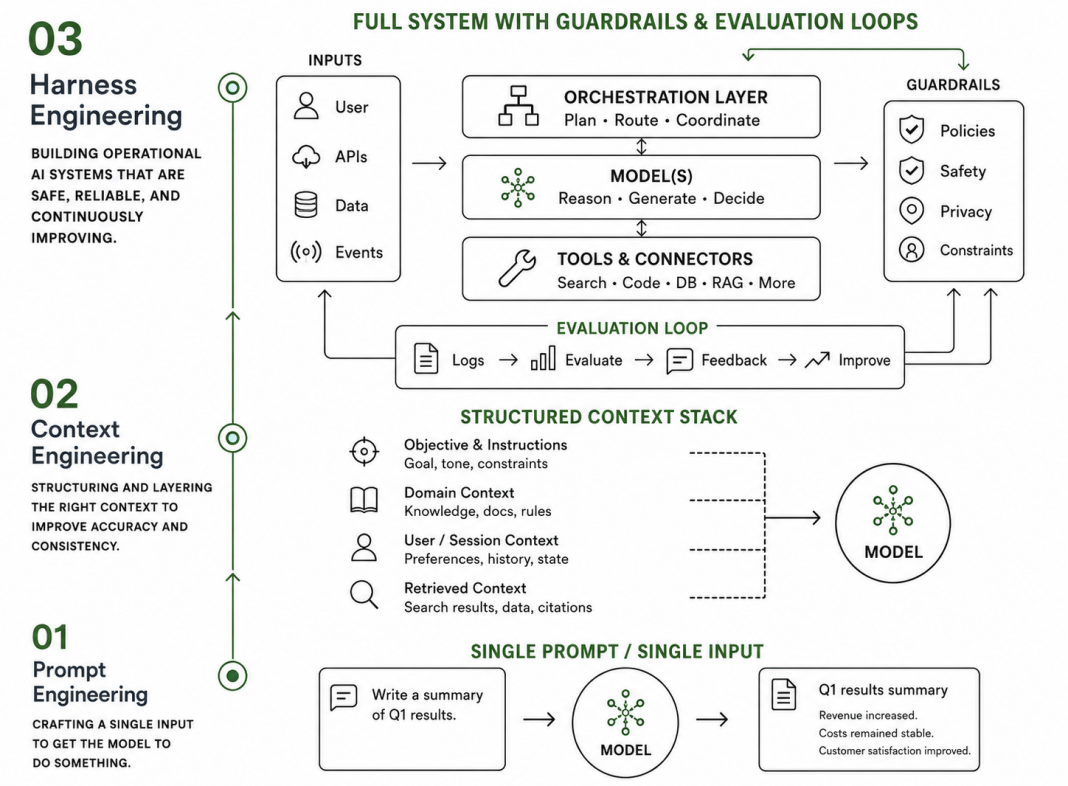

Figure 2.1: Conceptual progression from a single prompt, to structured context, to an operational harness with guardrails and evaluation loops.

2.4.1 From model calls to diagnostic workflows

Prompt-centred designs rely on a single model call to move from repair text to an answer. Context-centred designs improve this by giving the model a richer information environment, including task instructions, label definitions, case evidence, and retrieved examples. For industrial diagnosis, the information environment still needs an execution structure around it. Evidence has to be extracted before it is judged, candidate labels have to be mapped before they are accepted, and weak support has to remain visible instead of being hidden inside a fluent final answer. Prior work on language-agent patterns shows that reasoning can be paired with actions, tool calls, planning, and feedback rather than being left inside one unstructured response [54, 55, 56, 57].

Chen et al. identify a similar gap in AI-enhanced digital twins for maintenance. Industrial implementation is limited by scale, data integration, model integration, and workforce readiness, not only by model quality [52]. The same lesson applies to warranty-case analysis. A method has to fit a real review workflow, not only a benchmark classification task.

2.4.2 Planner-guided decomposition

Recent work on LLM agents describes planning, memory, state management, and feedback as core mechanisms [54, 55, 56, 57]. Here, agentic does not mean unrestricted decision-making. It means that the workflow separates responsibilities and uses the current state of evidence to decide which reasoning operation is needed next. In warranty cases, some cases need little more than mapping a clear part mention. Others need critique, historical comparison, and more cautious final judgement.

A state-oriented view avoids treating every step as an independent classifier. Extracted signals, tentative hypotheses, mapped labels, retrieved neighbours, critique notes, and confidence information remain visible to later stages. This record shows how the label became plausible, where the evidence stayed weak, and which constraints shaped the final judgement.

Planning and decomposition are useful when a task contains several different reasoning operations. Diagnostic attribution from warranty text requires evidence identification, hypothesis formation, conflict checking, mapping to a controlled label space, and a final decision about whether the evidence is strong enough. Broader work on multi-step reasoning and agent planning shows the value of explicit intermediate structure [30, 55, 57]. In an engineering setting, the useful part is not long free-form reasoning. It is the decomposition of the diagnostic task into operations that can be checked.

2.4.3 Evidence-grounded explainability in label assignment

Warranty cases contain plausible distractors. A replacement action, a service kit, or an exchange unit can appear diagnostic in isolation, while the final failed-component label depends on the relation between symptom, inspection, part context, and repair outcome. Feedback and reflection methods in LLM systems show one way to check

an initial answer before accepting it [58, 59]. In this setting, the check is narrower. It asks whether the current hypothesis is supported by case evidence, whether the mapping is too broad or too specific, and whether confidence should be lowered.

The evaluation follows the same path. Final-label metrics such as accuracy, precision, recall, and F1 describe whether the system identifies the target component label across a benchmark population [60]. Field-level metrics such as exact match and token precision check whether repair text has been converted into usable structured fields [61]. Retrieval diagnostics check whether similar cases remain close after approximate search, and output-control checks show whether the final answer stays inside the canonical label dictionary. These checks give a broader view than a single accuracy value.

2.5 Related Work

Recent work has investigated how language models, retrieval-augmented reasoning, and agentic AI systems can support diagnostic tasks by structuring evidence and making intermediate reasoning more transparent. This body of work provides a useful basis for positioning the present thesis, which applies these ideas to warranty case analysis and component-level failure attribution.

2.5.1 Industrial root cause analysis and fault diagnosis

Root cause analysis is an important task in industrial settings because failure information is often distributed across heterogeneous sources, such as repair notes, replaced parts, maintenance histories, sensor signals, and expert knowledge. Recent studies have explored the use of large language models for industrial fault diagnosis and explanation. FaultExplainer integrates process data, fault detection results, and LLM-based explanation to support interpretable fault detection and diagnosis in chemical processes [62]. Similarly, FaultGPT investigates fault diagnosis question answering from industrial vibration signals and generates diagnostic reports rather than only producing classification labels [63]. These studies show that language models can support fault analysis by translating complex technical evidence into more interpretable diagnostic outputs.

Many industrial fault diagnosis studies focus on condition monitoring, sensor-driven data, or signal-based fault detection [64, 65, 66]. In contrast, warranty analysis often depends on textual repair records, warranty claim forms, and replaced-part or service information rather than continuous sensor measurements [67, 68]. This creates a different diagnostic problem: the system must interpret incomplete and ambiguous repair narratives, while still producing a controlled failure attribution that can be reviewed by domain experts.

2.5.2 Agentic and retrieval-augmented reasoning for root cause analysis

Another related research direction is the use of agentic AI and multi-agent systems for industrial and operational reasoning. In manufacturing, LLM-enabled multi-agent systems have been proposed to improve adaptability, natural-language instruction understanding, and coordination between agents in complex production environments [69]. This is relevant to the present thesis because it shows how LLMs can be embedded into structured workflows rather than being used as isolated text generators.

For root cause analysis specifically, recent work has investigated whether large language models can reason over complex operational evidence. OpenRCA introduces a benchmark for evaluating LLMs on software failure root cause analysis using logs, metrics, and traces from enterprise systems [70]. Follow-up analysis of LLM-based RCA agents shows that agentic RCA systems may still fail because of hallucinated data interpretation, incomplete exploration, and weak agent-environment interaction [71]. These findings suggest that simply adding an LLM agent is not sufficient for reliable RCA. The reasoning process needs to be constrained, evidence-grounded, and designed so that intermediate steps can be inspected.

Retrieval-augmented generation is also closely related to this thesis. RAG methods combine language models with external knowledge sources, allowing generated outputs to be grounded in retrieved information rather than relying only on the model's internal parameters [72]. In an industrial diagnostic setting, this is particularly important because relevant evidence may come from historical cases, domain-specific terminology, component information, or previous repair patterns. Retrieval therefore supports both diagnostic relevance and traceability.

3

The proposed Agentic System Workflow

The method starts from the warranty-case problem defined in Chapters 1 and 2. Repair scope, text, parts context, and component attribution do not always point to the same conclusion. The chapter first explains how review difficulty is assessed, then defines the planner-guided artifact, retrieval and knowledge constraints, reproducibility settings, and the evaluation measures used in Chapter 4.

Parts of the thesis language were edited with assistance from a large language model (LLM). The research design, analysis, results, and conclusions remain my responsibility.

3.1 Warranty case analysis

The human-review problem is turned into an analysis setup before any model workflow is discussed. Warranty cases are not treated as clean diagnostic examples. They are treated as operational records whose repair scope, text evidence, and final component attribution have to be compared.

3.1.1 Difficulty assessment on human review

The difficulty assessment uses the same evidence roles that appear in real warranty cases. A case is treated as difficult when these sources point to different levels of detail or when the repair narrative mixes symptoms, diagnostic observations, service actions, and component clues in a way that makes attribution uncertain. This assessment is supported by automotive diagnostic-text research showing that repair verbatim, labour codes, observed faults, and service actions have to be separated before they can support diagnosis, and by CMMS studies showing that structured labels and free-text descriptions can encode different aspects of the same event [9, 10]. The same logic also motivates a field-level check: if symptoms, repair actions, and component clues are not extracted consistently, the final label becomes harder to justify. Information extraction evaluation and maintenance work-order annotation studies support this kind of intermediate check because they show that field recovery and expert agreement affect how reliably technical text can be used for later analysis [61, 73].

3.1.2 Task formulation

Diagnostic input is not a single clean sentence. It is a case representation assembled from field-reported repair text and structured context. Output is one canonical diagnostic label. A label is valid only when it belongs to the allowed dictionary used by the implementation. Industrial review and batch evaluation depend on stable labels rather than free-form component descriptions, so the dictionary constraint is part of the method. Correctly identifying the failing part also provides the component-level basis needed before later responsibility attribution can be discussed. Figure 3.1 shows why this component-level distinction matters before responsibility is assigned.

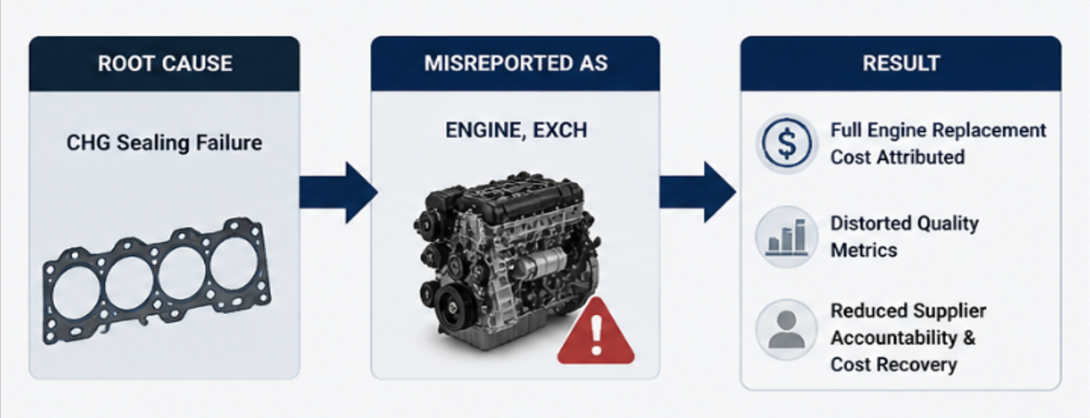

Figure 3.1: Consequence of mapping a local failure mechanism to an overly broad repair label.

3.2 System overview and architecture

The system is introduced after the review problem has been defined. At the system boundary, one warranty case enters as operational repair text and structured context. Output is a controlled diagnostic label together with the evidence needed for human checking.

One repair case is processed at a time. Each case enters the workflow as repair text plus parts context and is stored in a shared diagnostic state. Planner control reads this state and decides which operation should run next. Specialist operations add normalized text, extracted signals, hypotheses, mapped candidates, retrieval evidence, and final judgement information to the same state.

Architecture is separated into three responsibilities. Planner control handles order and stopping conditions. Perception operations make the technician-written record usable for later reasoning. Diagnosis operations compare evidence, apply constraints, and commit to a controlled label. Historical retrieval and knowledge rules sit beside this flow as evidence and constraint sources rather than independent decision makers.

Single-case inspection and batch evaluation use the same architecture. State fields are preserved in both settings, so a row-level prediction can be checked through its intermediate evidence rather than treated as an isolated model answer. The figure

should be read from left to right. Input evidence enters the planner-guided sequence, specialist operations write to shared state, retrieval adds historical comparison, and the final output contains a label, confidence, and reason.

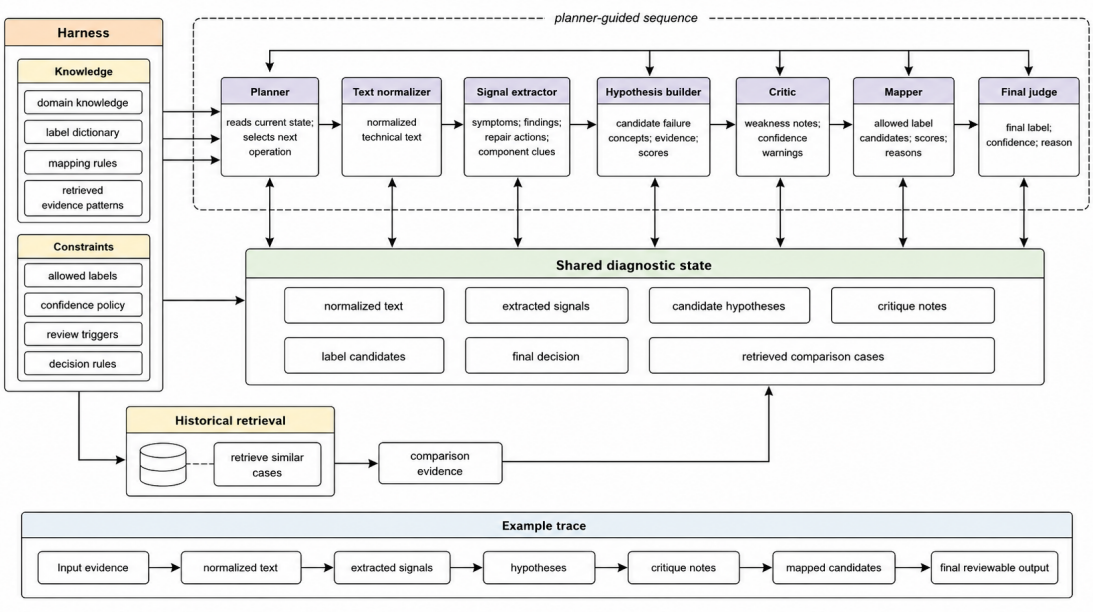

Figure 3.2: Planner-guided diagnostic operations with harness constraints, shared state, retrieval evidence, and an example trace.

Table 3.1: Key elements of the shared diagnostic state.

State element	Role in the evidence trace	Use
Input record	Repair narrative and parts context for one work order.	Input
Normalized text	Standardized technical wording passed to later operations.	Text normalizer
Diagnostic signals	Symptoms, repair actions, findings, and component clues extracted from the narrative.	Signal extractor
Candidate evidence	Hypotheses, critique notes, and mapped label candidates used before final judgement.	Hypothesis, critic, mapper
Historical evidence	Retrieved cases and retrieval diagnostics used for comparison.	Retrieval branch
Output for review	Final label, confidence, reason, stop status, and execution history.	Final judge, orchestrator

3.3 Planner-guided diagnostic workflow

The architecture only helps if each operation has a clear responsibility. Instead of asking for one direct answer, the workflow stores intermediate evidence and lets later operations read what earlier operations found.

3.3.1 The planner routes each operation

Planner control is the organizing layer of the workflow. At each step, it reads the current diagnostic state and selects the next specialist operation. Choice is bounded by deterministic safeguards. Mapping cannot run before hypotheses exist, and final judgement cannot run before candidate labels have been mapped. Routing combines state-based model selection with simple validity rules.

Critique is handled as a conditional operation rather than a fixed mandatory step. Planner control calls the critic when the case contains conflicting signals, weak candidate support, or a broad repair-action label that needs extra checking. Straightforward cases proceed directly to mapping after hypotheses, while complex cases receive critique notes before mapping.

Bounded control keeps the agentic part of the method narrow. Agents do not vote independently, and the system does not accept arbitrary generated labels. Planner control organizes the order of reasoning operations, while the allowed label dictionary and policy rules constrain the content of the final output. In this sense, agentic design means state-driven task organization rather than unrestricted autonomy.

3.3.2 Each operation writes evidence to state

Specialist operations are divided into perception and diagnosis. Text normalization turns compressed, multilingual, or irregular workshop phrasing into a cleaner technical narrative while preserving the meaning of the original record. Signal extraction reads the normalized and original text together and writes structured evidence to the state, including symptoms, explicit failures, repair actions, consequential damage, key patterns, component mentions, and confidence notes. These two operations do not decide the final label. Their purpose is to make the record easier to inspect and easier for later operations to use.

Diagnosis starts after the evidence has been extracted. Hypothesis builder uses the text and extracted signals to create up to three failure concepts, each with supporting evidence, a score, and a short explanation. When the critic is triggered, it checks whether the current reasoning confuses a repair action with the failed component, overlooks consequential damage, or rests on weak support. Mapper operation uses the extracted evidence, hypotheses, any critic notes, and selected knowledge entries to map engineering concepts to allowed labels. Final judge receives the mapped candidates together with the accumulated evidence and returns one controlled label, a confidence value, and a reason.

A typical chain can be read as a sequence of state updates. A repair record enters with a short narrative and parts context. Normalization produces a clearer version of the narrative. Signal extraction records that the case contains a reported symptom, a diagnostic observation, a repair action, and one or more component clues. Hypothesis building turns those clues into ranked failure concepts. If triggered, the critic adds warnings when the evidence points to a secondary repair outcome rather than the originating component. The mapper translates the remaining concepts into controlled labels, and the final judge selects the label that is best supported by the visible evidence. When historical retrieval is enabled, retrieved cases are added

to the same state as comparison evidence, but they do not replace the diagnosis operations.

3.4 Supporting evidence and harness control

Historical cases and domain rules reduce dependence on a single free-form model answer. Similar cases are stored as comparison evidence in the shared state, while selected knowledge entries and policy rules guide mapping into the allowed label space. These mechanisms support the final judgement, but they do not make the final decision by themselves.

3.4.1 Retrieval rank with human checking consideration

Retrieval runs as a read-only branch when parts context is available. It uses two warranty-case views. The material view represents parts handled during the job, and the work-description view represents symptoms, diagnostic observations, and repair actions. Each view is compared with the same view in a candidate historical case, and the final retrieval score is a weighted sum.

$$s_{\text{ret}}(q, c) = w_M \cdot \cos(M_q, M_c) + w_D \cdot \cos(D_q, D_c), \quad w_M + w_D = 1 \tag{3.1}$$

where M_q and M_c are material or parts-use vectors for the query and candidate case, and D_q and D_c are work-description vectors. In this implementation, the material view maps to `All Parts in Job (one row)` and the description view to `Repair Order Text (one row)`. Weights are fixed to $w_M = 0.70$ and $w_D = 0.30$.

The larger material weight is used because parts context is a stable indexing signal, even though it is not a ground-truth failure label. The description view keeps symptoms, tests, and repair actions in the ranking, but it receives a smaller weight because workshop repair text is short, multilingual, and unevenly written.

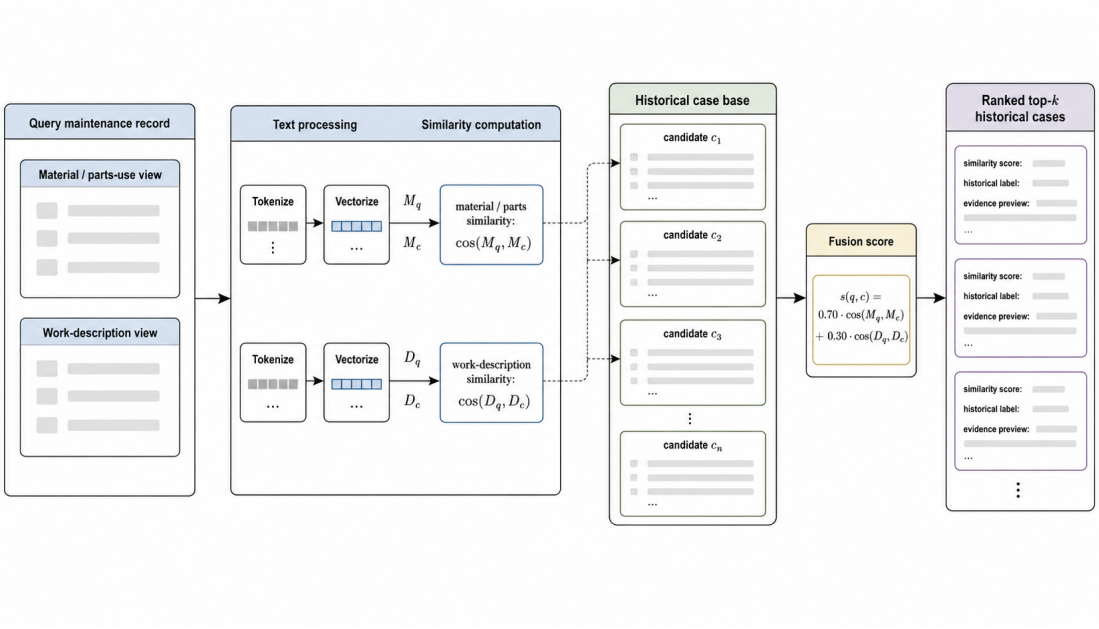

Figure 3.3: Historical-case retrieval scoring from warranty-case views to ranked comparison cases.

Retrieval supports exact ranking and approximate candidate selection. In exact mode, every historical record is scored. In ANN mode, SimHash buckets reduce the candidate set before the same ranking score is applied. If the approximate candidate set is too small, the branch falls back to exact search.

ANN is included for scalability rather than as a separate diagnostic model. As the historical case base grows, scoring every query against every stored case becomes less attractive, especially when both parts context and repair text have to be compared. The methodological check is not only whether ANN reduces the candidate set, but whether it still preserves the neighbours that exact retrieval would have returned. For this reason, the evaluation compares ANN against exact retrieval with top-1 preservation, top-5 overlap, and fallback rate. These measures show whether approximate search keeps the same comparison evidence available before any runtime advantage is claimed.

Retrieved cases are also ordered for human checking. The similarity score decides which historical cases are close. The priority score decides which results are most useful to inspect first. Priority combines ANN retrieval strength, result confidence, and a normalized review-impact factor.

$$p_{\text{review}}(q, c) = s_{\text{ANN}}(q, c) \cdot \gamma_c \cdot \iota_c \tag{3.2}$$

Here $s_{\text{ANN}}(q, c)$ is the retrieval score after approximate candidate selection, γ_c is the confidence value attached to the result, and ι_c is a normalized review-impact value. The impact term is used only as a relative review-severity signal, not as a reported amount. Retrieved cases stay visible as comparison records with similarity scores, historical labels, parts previews, and retrieval diagnostics.

3.4.2 The knowledge base and rules keep output controlled

Knowledge injection uses a small structured knowledge base instead of a long instruction block. Repeated repair patterns and human error-review notes are rewritten as general diagnostic entries. Each entry stores the canonical label it supports, a short failure description, indicative keywords, negative rules, and a priority value. These entries are stored as JSON objects so that they can be selected, logged, and checked separately from the prompt text.

Before mapping, the current repair text, normalized text, and extracted evidence are matched against the keyword fields in the JSON entries. Entries with keyword overlap are ranked with their priority value, and only the top entries are passed to the mapper. The selected entries guide interpretation of the current case, but they do not replace the current case evidence.

The allowed label dictionary and compact policy rules keep the final answer inside the industrial label space. Policy rules state that a repair action alone must not determine the diagnostic label, that primary failure should be distinguished from consequential damage, and that the final answer must be a valid dictionary label or `unknown`. Mapping limits candidate labels to the allowed dictionary, and the final judge selects one label from the mapped candidates with calibrated confidence.

Final judgement uses the accumulated state rather than starting a second free-form diagnosis. Extracted evidence, hypotheses, critic output, retrieval results, mapped candidates, and confidence information all remain part of the decision context. If evidence is weak or conflicting, the workflow can return `unknown`. When support is indirect, it can keep a valid label while lowering confidence.

3.5 Experimental design and metrics

Evaluation connects the workflow design to measurable outputs. The completed comparison tests the full workflow against a direct baseline. Intermediate checks examine field extraction, historical retrieval, knowledge use, output control, and the main validity limits of the evaluation.

3.5.1 Baseline design and comparison

Evaluation compares the full workflow with a simpler setting that keeps the same input records and target label space. The direct model baseline is the completed comparison setting. Other mechanism-level checks report retrieval preservation, review-priority ordering, knowledge activation, and controlled-output behaviour.

Direct model baseline follows a fixed execution path. As shown in Figure 3.4, the repair text is normalized, similar cases can be retrieved from the local case index, and the model is then asked to interpret the record and select a label in one pass. Preprocessing and retrieval prepare the context, but the diagnostic judgement is still concentrated in a single request. Retrieved cases, when enabled, enter as prompt context rather than as evidence stored in a shared diagnostic state.

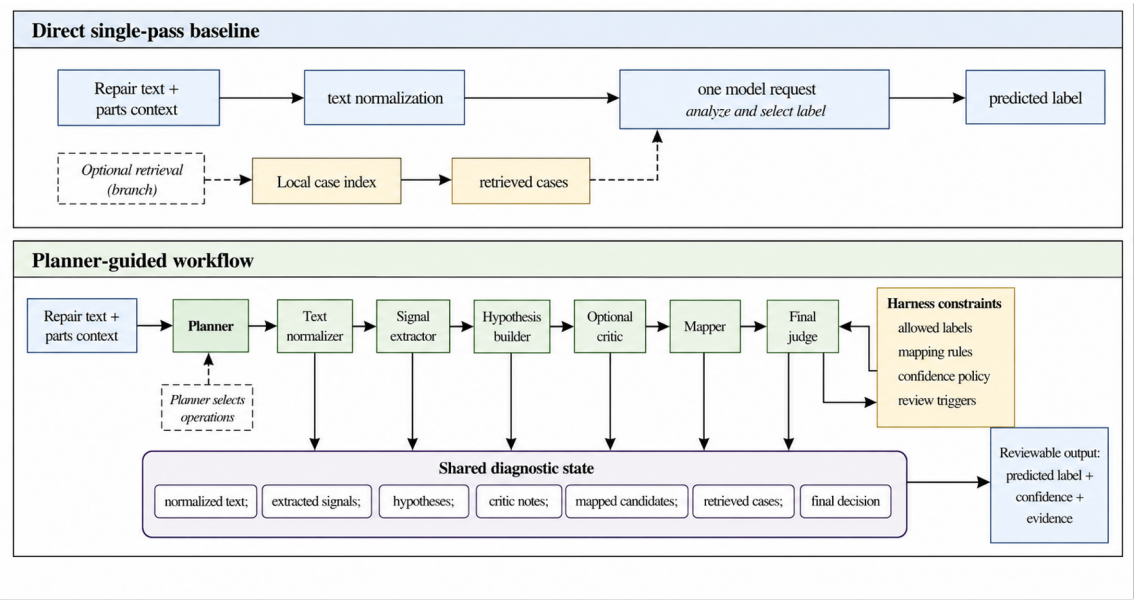

Figure 3.4: Direct single-pass baseline and planner-guided diagnostic workflow.

Baseline comparison differs from the full architecture in clear places while keeping the task itself unchanged. It has no planner that decides which operation is needed next, no state object that preserves intermediate evidence across operations, no conditional critic for uncertain or conflicting cases, and no separate mapper and final judge that enforce controlled output. A fixed one-pass diagnostic workflow is kept distinct from the planner-guided workflow described above.

The full workflow is treated as the reference setting for the remaining measurements. The specific measurements below examine final labels, extracted fields, retrieval behaviour, knowledge use, and output control.

3.5.2 Metrics evaluate the agentic system from different angles

The evaluation follows the route taken by the workflow instead of reducing the system to one final-label score. It first examines whether field-reported warranty text has been converted into usable evidence, then whether retrieval and knowledge support remain active inside the decision process, and finally whether the controlled output agrees with the reference bucket.

Before a label can be trusted, the repair text has to be represented as fields that carry the main technical content of the case. Evidence construction is measured with Exact Match (EM) and Token Precision (TP). EM checks complete field recovery, while TP checks token relevance when wording differs from the reference. This follows information-extraction evaluation practice such as the MUC evaluations, where system outputs were judged by whether the expected fields were recovered from text [61].

The workflow-level metrics focus on what separates the agentic design from a direct model request. Historical cases should remain close enough to guide human

checking, and selected knowledge should be visible in the decision path. Workflow support is measured with Top-1 Preservation (T1P), Top-5 Overlap (T5O), Fallback Rate (FR), and Knowledge-Supported Decision Rate (KSDR). T1P and T5O check whether ANN retrieval preserves the closest exact neighbours. ANN systems such as HNSW and FAISS are commonly evaluated by how well approximate search keeps the same nearest neighbours while improving retrieval efficiency [47, 49]. FR records how often approximate search returns to exact search. KSDR is introduced in this evaluation to check whether selected knowledge entries support the mapped candidate or final label. Its role follows the need for human-checkable industrial decision support [13, 14].

Endpoint quality is checked through the final diagnostic label. Structured evidence, retrieved cases, and knowledge constraints are useful only if they lead to a controlled label that can be compared with the reference. Final diagnosis is measured with Accuracy (Acc.), Precision (P), Recall (R), and F1-score (F1). Acc. measures overall agreement with the reference bucket, P measures the reliability of positive predictions, R measures the recovery of positive cases, and F1 summarizes the balance between P and R [60].

3.6 Validity considerations

The evaluation is designed around one warranty-case attribution task rather than a general diagnostic benchmark for all industrial maintenance text. Two data views are used for different purposes. A full candidate population provides the reporting basis for overall label performance because it contains both aligned positives and negative records. A manually reviewed positive reference set is used to check whether the workflow can recover a known target label. This positive reference is useful for target-label recovery, but it cannot measure false positives by itself. Overall label metrics are only reported on an evaluation basis that includes both positive and negative records.

Field-level reference is also limited in scope. It is built from repeated normalization of the same repair text by different large language models and then consolidated into a pilot consensus reference, rather than a fully expert-audited slot annotation. Results are tied to the current label dictionary, knowledge base, prompt versions, retrieval settings, and warranty case distribution. Reusing the method in another component family or label hierarchy would require checking these resources again before comparing performance. For reproducibility, the model configuration, prompts, knowledge base, label dictionary, retrieval index, evaluation scripts, and generated batch outputs should be preserved together.

4

Results and Analysis

Results are organized around the evaluation methods introduced in Chapter 3. The system is tested on label recovery, field construction, historical retrieval, knowledge support, and final-output control. The chapter separates these measurements because each one answers a different part of the warranty-case attribution problem.

4.1 Data complexity creates review difficulties

4.1.1 Aligned data scopes define the evaluation basis

One record corresponds to one historical warranty claim where an engine exchange was recorded. Each record contains a repair narrative, parts context, structured service fields, and an expert review bucket for the CHG attribution task. The engine-exchange candidate population contains 628 records. This population is the main evaluation universe because all records were originally recorded under the same broad repair scope. Raw repair text, job identifiers, chassis identifiers, supplier names, and other source-specific information are excluded from the analysis.

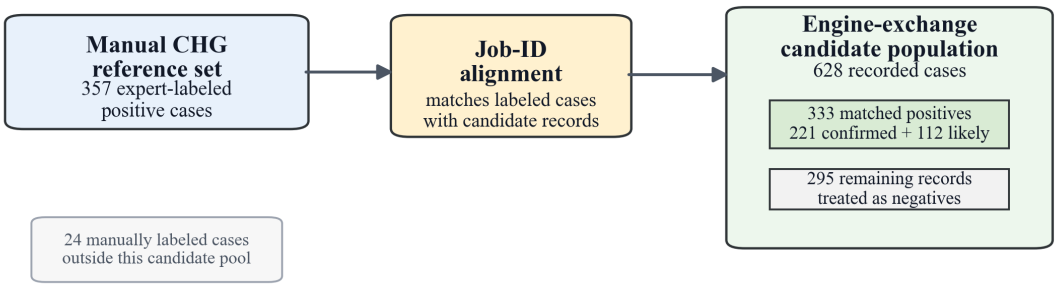

Figure 4.1: Job-ID alignment between the manually labeled CHG reference set and the engine-exchange candidate population.

The positive reference set is built from manual CHG annotation. It contains 357 cases that were reviewed as belonging to the CHG-related class. These expert-labeled CHG cases are then aligned with the engine-exchange candidate population through job identifiers. After alignment, 333 of the manually labeled CHG cases are present inside the 628-record engine-exchange population. The remaining 24 manually labeled CHG cases are outside this particular engine-exchange candidate

pool and are not counted in the mixed-label evaluation. Inside the 628-record population, the aligned positive cases are split into 221 confirmed CHG cases and 112 likely CHG cases. The other 295 records are treated as negative for the binary CHG-detection evaluation. Figure 4.1 shows this alignment.

Table 4.1: Evaluation bases and reported outputs in Chapter 4.

Evaluation basis	N	Purpose	Metrics
Manual CHG benchmark	357	Recovery test on expert-positive cases	Baseline comparison and target-label recovery
Mixed-label basis	628	Main population for final-label evaluation	Acc., P, R, F1, output control, and field availability
Aligned positive subset	333	Positive cases within the 628-record set	Recall and false-negative analysis
Retrieval index	357	Exact-versus-ANN retrieval comparison	T1P, T5O, and FR
Field-level pilot basis	628	Intermediate evidence-field evaluation	EM and TP

Table 4.1 is used as the map for the rest of the chapter. Target-label recovery is reported on the manually reviewed positive benchmark. Overall label quality and output control are reported on the 628-record mixed-label basis. Retrieval preservation is tested on the 357-record retrieval index, while field-level checks use the available batch fields from the mixed-label basis.

4.1.2 Manual review reveals attribution uncertainty

Table 4.2: Human-review evidence available for the CHG benchmark.

Human-review dimension	Evidence
Reviewer basis	One senior engineering review pass was used to construct the CHG-positive reference set.
Review workload	The 357-case CHG reference required about eight hours of review, roughly 1.3 minutes per case on average.
Recorded-label gap	357 of 357 CHG-positive cases were still recorded as *ENGINE, EXCH* in the source failing-part field.
Label certainty	232 cases were marked as CHG failure and 125 cases were marked as likely CHG.
Ambiguity indicators	The benchmark includes dense parts context, abbreviated repair wording, local-language text, and cases where symptom, action, and cause information are mixed.
Disagreement evidence	No independent second-reviewer record is available, so uncertainty is observed through the confirmed/likely split instead.

The manually reviewed CHG benchmark gives a closer view of why human review is not a simple reading task. All 357 cases in this benchmark are still recorded in the source field as *ENGINE, EXCH*. The source label identifies the repair scope, but it does not identify the component-level failure attribution. The reviewer has to infer whether the broad engine-exchange outcome was driven by CHG-related evidence in the repair narrative, parts context, and coded service information.

The expert reference also contains a certainty split. Among the 357 manually reviewed CHG cases, 232 are marked as engine exchange due to CHG failure and 125 are marked as likely engine exchange due to CHG. The likely group is not a separate class in the final binary evaluation, but it is important for judging review difficulty. It shows that a large share of cases can support the target attribution without giving the reviewer the same level of evidence as a fully confirmed case. Table 4.2 summarizes the review evidence that can be reported from the available annotation record.

4.2 Intermediate decomposition produces usable diagnostic support

The workflow is not evaluated only by its final label. Its intermediate parts are checked because the system is designed to turn an unstructured warranty case into diagnostic support that an engineer can check. Field construction shows whether the repair text becomes usable structured evidence. Historical comparison shows whether similar cases remain stable enough to support review. Knowledge support shows whether domain entries are actually used in completed decisions.

4.2.1 Field-level evidence makes intermediate reasoning visible

Field-level results use the same 628-record mixed-label basis as the canonical-label evaluation. One row corresponds to one engine-exchange repair case, and the relevant batch-output fields are the normalized repair text and the extracted diagnostic signals stored in the shared state. This view reads the system at the perception level. It checks whether the workflow has converted an unstructured repair narrative into stable diagnostic fields before retrieval, mapping, and final judgement are considered.

The available output shows that the shared state is populated for most records. Component clues are present for 583 of 628 records, repair actions for 573 records, symptoms for 564 records, and explicit failure evidence for 537 records. Consequential damage is much less frequent, with 191 non-empty fields, because many repair narratives do not describe secondary damage. These counts are used only as field-availability context. A non-empty field still needs a correctness check because a broad component, a copied repair action, or a symptom written as a cause can all distort later diagnosis.

For field-level correctness, the reference is built from repeated normalization of the

same repair text by different large language models. The repeated outputs are compared field by field, and recurring normalized phrases are statistically consolidated into a pilot consensus reference. This gives a practical reference for the current experiment without claiming that every slot has been manually annotated by an engineer.

Exact Match (EM) is recorded separately for each extracted field, so a failure in component extraction is not hidden by better performance on longer action descriptions. Token precision then checks whether the extracted words are still technically relevant when the whole field does not match exactly.

Table 4.3 reports pilot values for field-level correctness. The table should be read as an intermediate evaluation of the perception stage, not as a final expert-audited annotation result.

Table 4.3: Field-level correctness results on the 628-record basis.

Field	Exact Match	Token precision
Component clues	74.8%	90.6%
Failure evidence	62.5%	84.2%
Symptoms	68.1%	86.7%
Repair actions	56.9%	80.4%

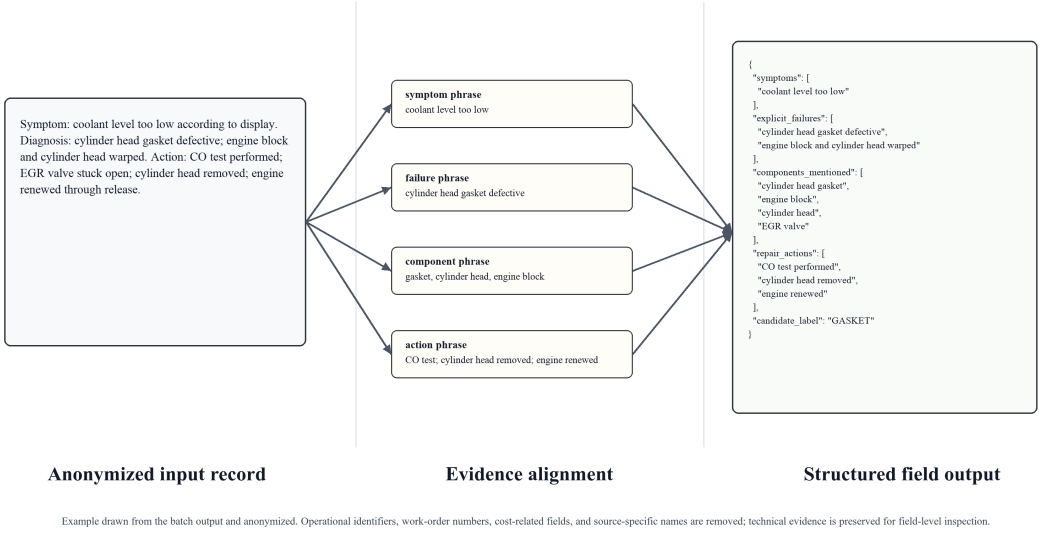

Figure 4.2: Anonymized example of field-level evidence construction from one batch-output repair record.

The pattern in Table 4.3 suggests that the workflow is better at finding relevant field content than matching the complete field exactly. Token precision is higher than Exact Match for all four fields, which means many mismatches still contain useful diagnostic words. Component clues are the strongest field because they are short and closer to canonical part terminology. Repair actions are weaker because service

narratives often contain several actions in one phrase, and the boundary between the required action evidence and surrounding workshop repair detail is harder to keep clean. Failure evidence and symptoms sit between these two cases, which fits their mixed role. They are more diagnostic than repair actions, but less compact than component names.

Figure 4.2 shows one anonymized example from the batch output. The record belongs to the 628-record basis and is one of the cases where the final controlled label is *GASKET*. Operational identifiers, work-order numbers, source-specific names, and all cost-related fields are removed. The example keeps the technical content needed for field-level inspection, including a coolant-level symptom, an explicit cylinder-head gasket failure, related component mentions, and repair actions. The important point is that the output is not only a fluent rewritten sentence. It is a set of separate fields that can be checked against the consensus reference and then used by later retrieval and mapping stages.

The example also shows why the field-level view is useful for error analysis. The symptom field should remain about observed coolant level behaviour. The explicit-failure field should contain the gasket diagnosis and related warped engine components, rather than a broad engine-exchange outcome. The repair-action field should keep actions such as diagnostic testing and cylinder head removal separate from the failed part itself. When a final label is wrong, these intermediate fields make it possible to see whether the problem started from normalization, signal extraction, mapping, or final judgement.

4.2.2 Retrieval keeps comparison cases close to exact search

The retrieval experiment uses exact search as the reference on the 357-record retrieval index. ANN retrieval first selects candidate records with SimHash locality-sensitive buckets and then applies the same weighted cosine score as exact retrieval. The diagnostics show whether ANN keeps the same leading neighbours and how often the fallback to exact search is needed.

Table 4.4: ANN retrieval preservation diagnostics on the 357-record retrieval index.

Diagnostic	Result
Top-1 preserved	355/357 (99.4%)
Mean top-5 overlap	99.0%
Full top-5 match	344/357 (96.4%)
Exact fallback	8/357 (2.2%)
Largest top-1 score gap	0.054

Table 4.4 shows that the approximate branch preserves the retrieved evidence rather than replacing it with a different comparison set. Top-1 preservation reaches 99.4%, and full top-5 agreement is observed in 344 of 357 queries. The few partial-overlap cases do not indicate a different retrieval objective. They occur when locality-sensitive buckets return a slightly different candidate set before the same weighted cosine scoring is applied.

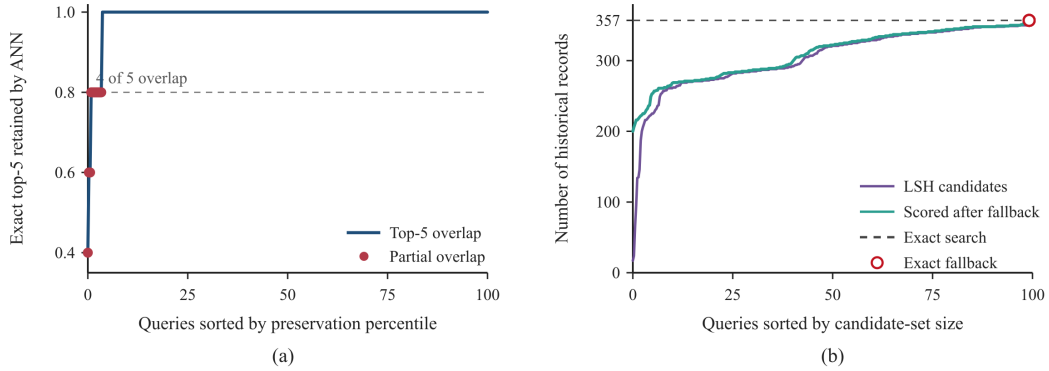

Figure 4.3: ANN retrieval preservation compared with exact retrieval on the 357-record retrieval index.

Figure 4.3 gives the distributional view behind the table. Panel (a) shows that nearly all queries keep the full top-5 set. Only a small number of cases fall below complete overlap, and most of those still retain four of five exact neighbours. Panel (b) shows the candidate-set behaviour. The raw locality-sensitive buckets often reduce the number of records to score, but the conservative fallback raises weak candidate sets back to exact search.

Because the retrieval index contains 357 records, runtime speed-up is not claimed as a measured result. The result here concerns whether ANN preserves the same comparison evidence as exact retrieval.

ANN preservation is then combined with review-priority ordering. Retrieval similarity decides which historical records are close to the current case. Review priority decides which completed outputs should be inspected first. The ranking uses the ANN top similarity, the stored confidence value, and a normalized review-impact score. This keeps the ordering evidence-based. A case with strong historical similarity, high confidence, and high impact moves up the review queue, while a similar but lower-impact case stays visible without taking the same review position.

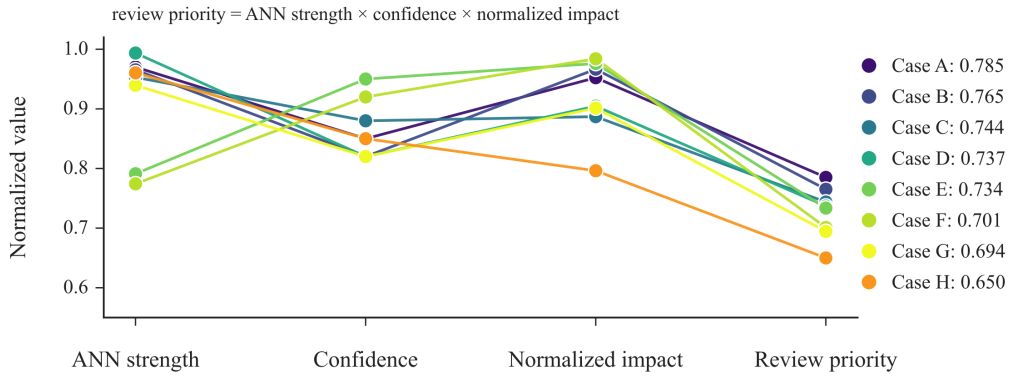

Figure 4.4: Data-derived review-priority ordering from completed batch outputs. Case identifiers and raw impact values are omitted.

Figure 4.4 uses actual completed batch outputs with non-zero confidence. The cases are anonymized as Case A–Case H. Each line shows how ANN strength, confidence, and normalized impact combine into the final review-priority score. The shape of the lines matters more than the case names. Some records have very strong historical similarity but lower confidence or lower impact, so they fall behind records where all three signals are high. This makes retrieval usable as a review queue rather than a passive list of similar cases.

Taken together, Table 4.4 and Figures 4.3–4.4 show two retrieval results. ANN keeps the comparison evidence close to exact search on the 357-record index. Completed batch outputs can also be ordered for review by combining retrieval strength, confidence, and normalized impact. This result gives retrieval-stability and review-ordering evidence. It does not by itself isolate the causal effect of retrieval on canonical-label accuracy or field-level evidence quality.

4.2.3 Knowledge entries are active in most completed cases

Knowledge injection is evaluated on the 628-record basis through selected knowledge entries, mapped candidates, and the final label. The funnel asks a narrower question at each step. It asks whether knowledge was used, whether it supported the final decision, and whether the supported decision also matched the expert bucket.

Figure 4.5 shows that 490 of 628 records use at least one selected knowledge entry. This corresponds to 78.0% of the completed batch. The number narrows to 438 records where the final decision is supported by the selected knowledge evidence, and to 382 records where that supported decision is also correct under the binary evaluation bucket. The gap between activation and supported correctness is useful. Knowledge use is not treated as correctness by itself. It is evaluated as an evidence path that still has to agree with the reference decision.

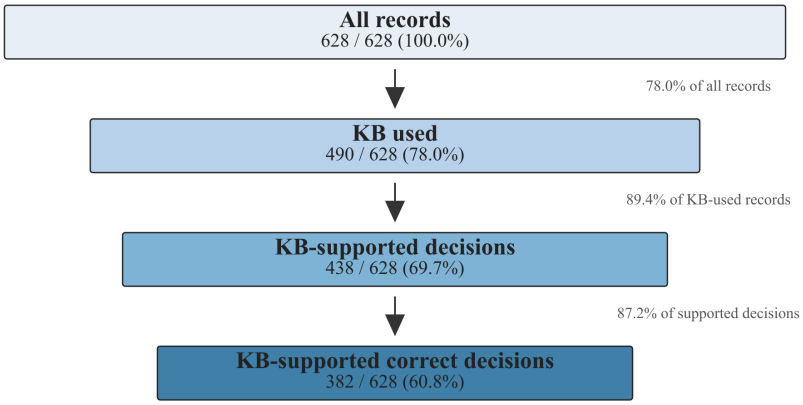

Figure 4.5: Knowledge-injection funnel on the 628-record basis.

4.2.4 Mechanism variants separate constraint and retrieval effects

Mechanism-variant comparison is reported on the 357-record positive benchmark. The output categories are target label recovered, unknown fallback, and other output. Resolved output is the non-unknown share of the same benchmark, and the invalid label rate is reported separately as a label-control measure. Figure 4.6 reports the same positive-benchmark count for all variants.

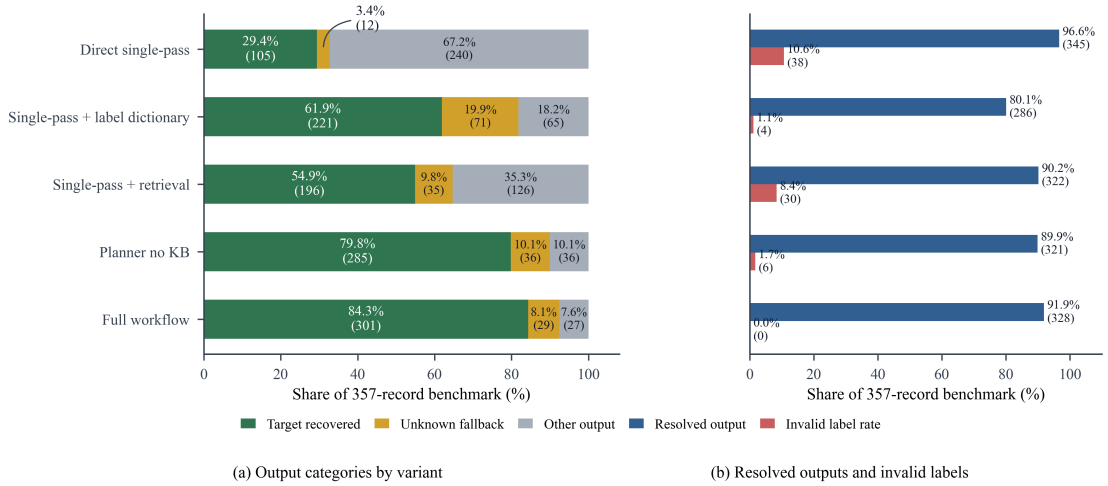

Figure 4.6: Mechanism-variant results on the 357-record positive benchmark.

4.3 Final-output results and error patterns

Final-output evaluation brings the intermediate support back to the prediction task. The first result checks positive-case recovery against the direct single-pass baseline. The mixed-label evaluation then tests whether the same workflow can recover CHG cases without surfacing too many negative records as positive.

4.3.1 Positive-case recovery improves over direct baseline

Target-label recovery is evaluated using the 357 manually annotated CHG cases. Because every record in this benchmark belongs to the positive reference set, the evaluation focuses on cases where the target condition is already known to be present. In this setting, the goal is to measure how often the method returns the canonical target label *GASKET*, rather than a fallback label or a broader repair label.

The direct baseline returns *GASKET* for 105 of 357 cases, giving a target-label recovery of 29.4%. The planner-guided setting returns *GASKET* for 301 of 357 cases, giving a recovery of 84.3%. In the direct baseline, 12 cases end in *unknown* and 240 cases are assigned to other non-target outputs. In the planner-guided setting, 29 cases end in *unknown* and 27 cases remain in the other non-target category. The comparison points to a clear workflow advantage. When the relevant failure

pattern is present, the planner-guided setting is much more likely to return the target canonical label instead of a broader or wrong-specific output.

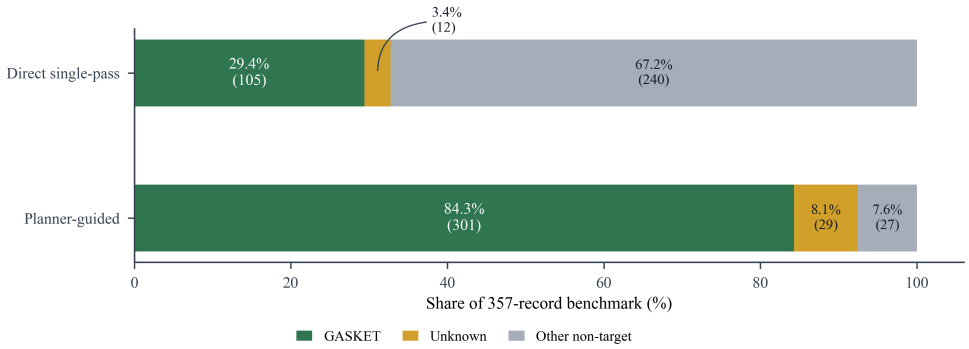

Figure 4.7: Distribution on the manually labeled positive benchmark. Target-label recovery counts *GASKET* outputs among 357 CHG-related cases.

Figure 4.7 shows the same result as two stacked output distributions. In the direct single-pass baseline, most outputs fall into the other non-target category, meaning that the direct request often commits to a broader or wrong-specific label even when the case has been manually labeled as CHG-related. In the planner-guided workflow, most outputs move into *GASKET*. The non-target category is kept separate from *unknown* so unresolved outputs and wrong-specific labels are not mixed together.

The result should not be read as evidence that the planner-guided workflow has 84.3% accuracy. The benchmark contains no negative records, so any method that always predicts *GASKET* would also score high on this specific measure. The comparison is a workflow-level reference rather than a component-level ablation. It shows how much the completed workflow improves over a direct single-pass route, while the preceding mechanism checks examine field construction, retrieval, knowledge support, and output control separately.

Diagnostic effectiveness is also tested on the full 628-record candidate population by comparing the controlled output with the expert bucket. Positive predictions are final *GASKET* labels. Negative predictions are all other valid canonical labels together with *unknown*. This setup tests both recovery and selectivity.

4.3.2 Overall final-label results

On the 628-record basis, the workflow gives 315 true positives, 74 false positives, 18 false negatives, and 221 true negatives. The confusion matrix in Figure 4.8 is a binary canonical-label evaluation. A final *GASKET* label is positive. Any other allowed label, together with *unknown*, is negative for this CHG task.

Canonical-label accuracy is 85.4%, calculated as 536/628. The numerator contains the 315 true positives and 221 true negatives. It summarizes overall agreement with the expert binary bucket, but it hides the balance between missed positives and over-attribution. Accuracy gives a compact system-level score, while precision and recall are needed to explain the failure mode.

Precision is 81.0%, calculated as 315/389. Among these 389 surfaced records, 315 are expert-positive and 74 are expert-negative. Precision answers a review-effort question. When the system says a record belongs to the CHG target label, how often is that surfaced case supported by the expert bucket? The remaining false positives show that some negative engine-exchange records still contain CHG-like wording strong enough to attract the final label.

Recall is 94.6%, calculated as 315/333. The denominator is the 333 aligned positive records inside the engine-exchange candidate population. The workflow misses 18 of these positive records. This high recall is consistent with the manual-positive benchmark because the planner-guided workflow is good at recovering the CHG target label when the evidence pattern is present. The main weakness is less about missed positives and more about distinguishing true CHG attribution from similar-looking negative records.

F1 is 87.3%. This score balances the high recall with the lower precision. It is the most useful single metric for the mixed-label setting because both types of error matter. False negatives hide relevant CHG cases from review, while false positives add records that the expert bucket does not support. The F1 score indicates that the planner-guided workflow is effective overall, but still needs tighter selectivity around cases where repair text mentions gasket-related or cylinder-head-related evidence without supporting the target attribution.

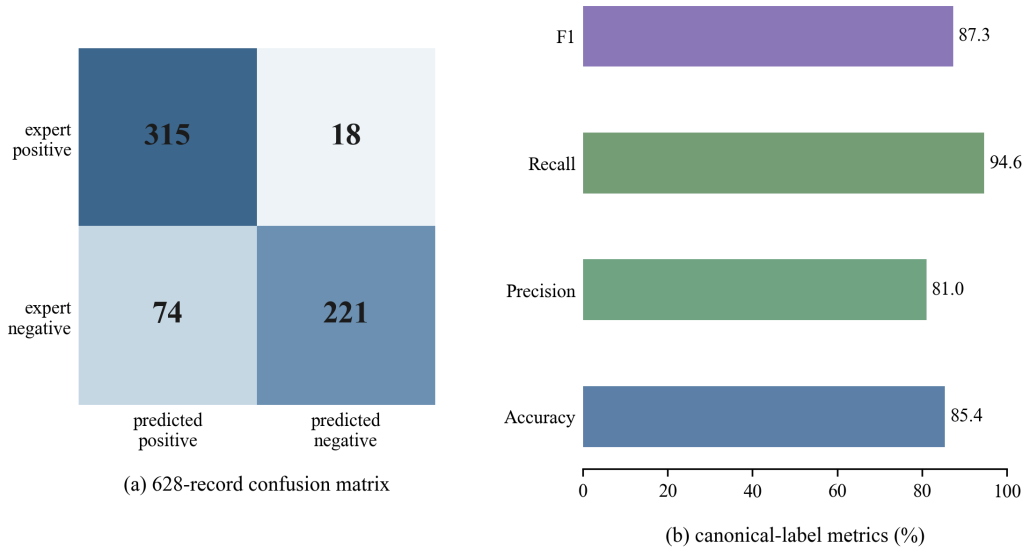

Figure 4.8: Canonical-label confusion matrix and metrics on the 628-record basis.

Final-label constraints are measured from the completed 628-record output. The distribution groups each final answer into a specific component label, an exchange-unit label, the *unknown* fallback, or an invalid label.

Figure 4.9 shows that 491 records end with a specific component label, 82 records end with an exchange-unit label, and 55 records end with the *unknown* fallback. No invalid label appears in the final output. The distribution gives a more compact

reading than a list of separate rates. It shows full dictionary control while keeping two review zones visible, broad exchange-unit labels and unresolved *unknown* outputs.

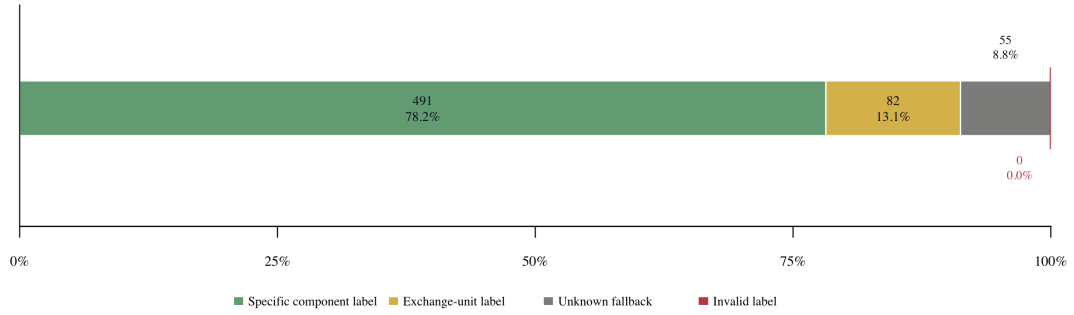

Figure 4.9: Controlled-output distribution on the 628-record basis.

Canonical validity is 100.0% for the available outputs in this evaluation. This metric is independent of correctness. The constraint layer keeps labels inside the controlled output space, while final accuracy still depends on evidence interpretation and mapping.

4.3.3 Error patterns in final outputs

On the 628-record basis, the mixed-label evaluation leaves 74 false positives and 18 false negatives. False negatives are expert-positive records that receive a broader or adjacent canonical label rather than *GASKET*. This is a canonical-label error rather than an invalid-output error. The system stays inside the dictionary but selects a label that does not match the target attribution.

False positives show the opposite problem. Expert-negative records can still be labelled *GASKET* when the repair narrative contains strong CHG-like surface evidence such as gasket, cylinder-head, coolant-pressure, or head-removal language. These cases are useful for error analysis because they show where evidence and reference label disagree, but they reduce precision and must not be reported as recovered CHG cases.

Confidence values need the same kind of contextual reading. High confidence can still appear in some false-positive cases. Confidence should be read as a review cue, not as a calibrated probability. The stronger technical safeguard is the combination of canonical labels, preserved evidence, retrieved comparisons, and explicit error review.

4.4 Summary of Findings

Aligned data makes the review problem concrete. The 628-record engine-exchange candidate set contains 333 CHG-related records after manual review, even though the original grouping describes a broader repair outcome. The practical issue is

not whether an engine exchange happened, but whether the record contains enough information to establish the component-level failure behind that outcome.

The planner-guided workflow changes the output distribution on the positive manual benchmark. Target-label recovery rises from 29.4% in the direct single-pass baseline to 84.3%, while other non-target outputs fall from 67.2% to 7.6%. The mechanism variants show the same pattern in smaller steps. Dictionary constraints reduce invalid labels, retrieval reduces some non-target outputs, and the full workflow combines those effects with planner-guided state updates.

The intermediate checks explain where the gain comes from. Field-level results show that the system usually captures relevant diagnostic words, with token precision above 80% across the pilot checks, while lower Exact Match scores reflect phrase-boundary and normalization differences. ANN retrieval preserves the exact top result in 99.4% of the current retrieval index, and knowledge entries are used in 490 of 628 completed records. These results support the use of decomposition, comparison cases, and compact domain rules as review support around the final label.

The mixed-basis endpoint result is strong but still leaves cases for expert judgement. The workflow reaches 85.4% accuracy and 87.3% F1 on the 628-record basis, with no invalid labels in the available outputs. Most remaining errors are false positives, which means the system is better at bringing possible CHG cases forward than at rejecting every record with similar surface evidence. The result fits the intended role of the workflow. It narrows and structures review work, but it does not settle ambiguous root cause questions by itself.

5

Discussion

Chapter 4 shows that the task is difficult for people, that a planner-guided system changes the behaviour of a direct model request, and that the final outputs are useful but still imperfect. These findings lead to three interpretations, followed by the practical and theoretical contributions, the main boundaries of the work, and the most useful follow-up work.

5.1 Answers to the research questions

5.1.1 RQ1 Warranty-case complexity makes attribution difficult

Warranty cases make failure attribution and root cause analysis difficult because the recorded repair scope is often broader than the component-level cause. The strongest evidence is the alignment between the engine-exchange candidate set and the manual CHG review. Within 628 engine-exchange candidate records, 333 are confirmed or likely CHG-related. These cases were not missing from the warranty system. They were present, but their component-level meaning was hidden under a broader service outcome. Manual review gives the same point from a work-practice angle. A senior engineer needed about eight hours to review the positive CHG-oriented set. That time was spent separating symptoms, diagnostic actions, replaced assemblies, parts context, and likely originating component evidence. The difficulty is not simple text reading. It is the judgement needed to decide whether a case describes the failure source, the repair response, or both.

Human review is difficult not only because of textual variation and incomplete field reporting, but because the source fields record a broad repair outcome. An expert has to reconstruct component-level attribution from the repair narrative, parts context, and service coding, and a substantial share of cases can only be assigned at a likely-attribution level. A support system is useful in this setting when it helps organize the case and keeps the final decision open to checking.

5.1.2 RQ2 Agentic decomposition improves label recovery

Agentic decomposition helps because it breaks a difficult reading task into smaller operations whose outputs can be checked. A direct single-pass request has to normalize the text, infer the failure concept, map it to a label, and decide confidence at

once. The planner-guided workflow separates those steps and stores their outputs in the shared diagnostic state.

Figure 4.6 makes the mechanism comparison more explicit. The direct single-pass setting recovers the target label in only 29.4% of the positive benchmark, while most outputs fall into other non-target labels. Adding a label dictionary raises recovery to 61.9% and sharply reduces invalid labels, but it also increases fallback because the model has less freedom to produce an unsupported label. Adding retrieval reaches 54.9% recovery and reduces other non-target outputs compared with the direct request, which suggests that historical cases help steer the model but do not solve mapping by themselves.

Planner-guided decomposition gives the largest separate gain before the full system is used. Without the knowledge base, recovery already reaches 79.8% and other non-target outputs fall to 10.1%. The full workflow then reaches 84.3% recovery, removes invalid labels in the available results, and keeps the final answer tied to normalized text, extracted signals, hypotheses, retrieved cases, mapped candidates, confidence, and a final reason. The advantage is not only a higher recovery rate, but a more useful diagnostic record around the label.

5.1.3 RQ3 Effectiveness is strongest in recovery and control

On the mixed basis, the system is effective for failed-component identification, especially when the priority is to recover CHG-related cases. Across the 628-record basis, the workflow reaches 85.4% accuracy and 87.3% F1, with high recall. This matches the positive-benchmark result because the system brings likely CHG cases forward instead of leaving them under broad engine-exchange wording.

Intermediate checks support the endpoint result. Structured fields are available for most records, and pilot token precision remains above 80% across the checked fields. The system usually captures relevant diagnostic words even when an extracted field does not exactly match the reference phrase. Historical retrieval keeps close agreement with exact search on the current index, and knowledge entries are used in most completed records. Root cause analysis support depends on this path, not only on the final label.

Remaining errors show where ambiguity remains. The main error pattern is false positives. Some records resemble CHG cases because they share symptoms, repair actions, or parts context, but they do not match the expert bucket. Output control works at the dictionary level, with no invalid labels in the available results, but valid labels do not remove diagnostic ambiguity. The system supports root cause analysis by narrowing and structuring the problem. It does not prove the physical root cause on its own.

5.2 Contribution to practice

In practice, the contribution is a workflow for turning broad warranty outcomes into component-level review tasks. Engine-exchange records are useful for service reporting, but they can hide more specific failure patterns. Many CHG-related cases sit

inside a broader engine-exchange population. The proposed workflow gives engineers a way to surface these cases, inspect the supporting fields, and separate originating component evidence from later repair actions. Its practical value therefore lies in standardizing noisy field information into reviewable failure-attribution mappings. By helping identify the correct failing part and the reason behind a repair or claim, the workflow can support quality teams in prioritizing actions for recurring issues observed in ongoing production.

At artifact level, the workflow also shows how LLM-based support can be fitted into an industrial review setting without asking the model to act as the final diagnostic authority. Normalized text, extracted signals, candidate labels, retrieved cases, knowledge entries, confidence, and final reasons are stored together. This makes the output more useful than a single label because an engineer can see where the decision came from and which part of the case still looks weak.

This practical role is also reflected in review prioritization. Historical retrieval and knowledge support do not replace expert judgement, but they can reduce the amount of undirected reading. Similar cases, confidence, and review-impact signals can help decide which outputs deserve attention first. The result is a batch-review aid that fits quality follow-up work better than a one-pass classifier or a free-form model answer.

5.3 Contribution to theory

This workflow is a clearer way to frame agentic systems for industrial failure attribution. Agentic design is treated here as task decomposition and state management, not as unrestricted autonomy. The workflow separates text interpretation, hypothesis formation, mapping, retrieval, and final judgement so that each step can be evaluated in relation to the final label.

Several research streams are connected here, although they are often discussed separately. Warranty analytics explains the attribution problem, technical text processing explains the need for structured fields, case-based reasoning explains historical comparison, and LLM-agent research explains planner-guided decomposition. Together, these ideas form one diagnostic workflow for field-reported warranty cases affected by workshop and market variation.

A final theoretical contribution is the evaluation structure. The results do not rely only on final accuracy. They separate positive-case recovery, field-level extraction, retrieval preservation, knowledge-supported decisions, output control, and mixed-basis label performance. This gives a broader view of what an agentic system contributes. It shows not only whether it predicts the right label, but also whether it builds a usable diagnostic path towards that label.

5.4 Limitations

Data is drawn from one industrial warranty setting. Wording style, available fields, label dictionary, and repair habits in this setting shape the results. A workflow that

works on these warranty cases still needs testing on other product lines and service contexts before wider claims can be made.

Reference labels also have limits. They are necessary for evaluation, but they are not the same as independent physical failure verification. A prediction can disagree with the reference because the system is wrong, because the repair text lacks enough evidence, or because the reference label describes a broader service outcome than the component evidence. Error analysis is needed alongside aggregate scores.

Field-level references have a narrower status as well. They are consolidated from repeated normalization by different large language models as a pilot consensus reference, not from a fully expert-audited slot annotation.

Retrieval helps by showing similar cases, but similarity is not causality. Two warranty cases can share service wording, parts use, or repair actions while coming from different underlying failures. The ANN results show that the approximate branch preserves the cases returned by exact retrieval in the current index. They do not prove that every retrieved neighbour is diagnostically meaningful.

Knowledge constraints keep outputs controlled, but they also set the boundary of what the system can say. If a useful label is missing or too broad, the workflow has to choose a weaker label or lower its confidence. The knowledge base, mapping rules, and confidence policy need regular review with domain experts, especially when the system is moved to a new product family.

Implementation also depends on model behaviour. Prompt wording, model version, API settings, and JSON parsing can affect intermediate outputs. The stored state makes these effects easier to check, but it does not remove them. Reproducible evaluation needs fixed prompt versions, model settings, data splits, retrieval settings, and parsing rules.

5.5 Future work

A broader evaluation with fixed data splits and repeated runs would be the most useful follow-up. More product lines and a larger set of expert-reviewed cases would make it possible to test whether the observed recovery, precision, and error patterns hold outside the current CHG-focused setting.

Human evaluation should also be expanded. Engineers can judge whether extracted signals match the repair narrative, whether retrieved cases are truly useful comparisons, and whether the confidence information helps them decide which records to review first. This kind of review is closer to the practical goal than label agreement alone.

Retrieval can be tested more fully once the historical case base grows. The current ANN check focuses on preservation against exact retrieval, not measured runtime gain. A larger index would make it possible to measure speed, tune the parts-text weighting, and test whether the review-priority score selects the cases engineers actually want to see first.

Knowledge-base and mapping-rule maintenance should become part of a feedback loop. When reviewers correct a label or reject a retrieved comparison, that feedback

can be turned into updated rules, new examples, or revised confidence policies. This would move the system from a one-time batch tool towards a maintained industrial review aid.

6

Conclusion

Automotive warranty cases can support quality follow-up, but they rarely state root cause directly. A broad service outcome can be correct in the warranty system and still hide the component-level failure that engineers need for attribution. The manual review in this work makes that gap visible. In total, 333 CHG-related records are found within a 628-record engine-exchange candidate set, and a substantial part of the review effort goes into separating repair actions from likely originating component evidence.

The planner-guided agentic workflow improves this setting by breaking the case reading task into smaller steps. Normalized text, extracted signals, hypotheses, mapped labels, retrieved comparison cases, knowledge entries, and the final decision are kept together. This structure changes the behaviour of a direct request. On the 357-record positive benchmark, target-label recovery rises from 29.4% to 84.3%, and other non-target outputs fall from 67.2% to 7.6%. The mixed 628-record evaluation gives 85.4% accuracy and 87.3% F1, with no invalid labels in the available outputs.

Intermediate results are as important as the endpoint scores. Field-level checks show that the system usually captures relevant diagnostic terms, ANN retrieval keeps the same top comparison case in 99.4% of the current index, and knowledge entries are active in 490 of 628 completed records. These results show how the workflow supports root cause analysis. It does not only return a label, but also gathers the surrounding material an engineer needs to check that label.

The system still has clear limits. The reference labels are based on available warranty evidence, not independent physical teardown. Some false positives remain because different failures can share symptoms, repair actions, and parts context. The field-level reference is a pilot consensus reference rather than a fully expert-audited slot annotation. Larger datasets, repeated runs, broader expert review, and testing on other product families are needed before the findings can be treated as general industrial evidence.

Agentic systems are promising for failure attribution when they are built as review support rather than autonomous diagnosis. Their value lies in standardizing field-reported variation, helping reviewers identify the correct failing part or reason for repair, and prioritizing quality actions in running production. Engineering judgement remains central, but the amount of unstructured reading before that judgement can be reduced.

Bibliography

[1] International Organization for Standardization, *ISO 14224:2016: Petroleum, petrochemical and natural gas industries – Collection and exchange of reliability and maintenance data for equipment*, 3rd ed., Geneva, Switzerland: ISO, 2016. url: https://www.iso.org/standard/64076.html.

[2] International Organization for Standardization, *ISO 13374-1:2003: Condition monitoring and diagnostics of machines – Data processing, communication and presentation – Part 1: General guidelines*, Geneva, Switzerland: ISO, 2003. url: https://www.iso.org/standard/21832.html.

[3] M. Hodkiewicz and M. T. W. Ho, "Cleaning historical maintenance work order data for reliability analysis," *Journal of Quality in Maintenance Engineering*, vol. 22, no. 2, pp. 146–163, 2016. doi: https://doi.org/10.1108/JQME-04-2015-0013.

[4] T. B. Sexton, M. P. Brundage, M. L. Hoffman, and K. C. Morris, "Hybrid datafication of maintenance logs from AI-assisted human tags," in *Proceedings of the 2017 IEEE International Conference on Big Data*, pp. 1769–1777, 2017. doi: https://doi.org/10.1109/BigData.2017.8258120.

[5] M. R. Karim and K. Suzuki, "Analysis of warranty claim data: A literature review," *International Journal of Quality & Reliability Management*, vol. 22, no. 7, pp. 667–686, 2005.

[6] S. Wu, "Warranty data analysis: A review," *Quality and Reliability Engineering International*, vol. 28, no. 8, pp. 795–805, 2012. doi: https://doi.org/10.1002/qre.1282.

[7] K. Zhong, T. Jackson, A. West, and G. Cosma, "Natural Language Processing Approaches in Industrial Maintenance: A Systematic Literature Review," *Procedia Computer Science*, vol. 232, pp. 2082–2097, 2024. doi: https://doi.org/10.1016/j.procs.2024.02.029.

[8] Y. Cho, "Automated Structuring and Analysis of Unstructured Equipment Maintenance Text Data in Manufacturing Using Generative AI Models: A Comparative Study of Pre-Trained Language Models," *Applied Sciences*, vol. 16, no. 4, article 1969, 2026. doi: https://doi.org/10.3390/app16041969.

[9] D. Rajpathak, "An ontology based text mining system for knowledge discovery from the diagnosis data in the automotive domain," *Computers in Industry*, vol. 64, no. 5, pp. 565–580, 2013. doi: https://doi.org/10.1016/j.compind.2013.03.001.

[10] A. Deloose, G. Gysels, B. De Baets, and J. Verwaeren, "Combining natural language processing and multidimensional classifiers to predict and correct CMMS metadata," *Computers in Industry*, vol. 145, article 103830, 2023. doi: https://doi.org/10.1016/j.compind.2022.103830.

[11] M. Sharp, M. Brundage, R. Sexton, and M. Navinchandran, "Discovering critical KPI factors from natural language in maintenance work orders," *Journal of Intelligent Manufacturing*, vol. 32, pp. 1721–1735, 2021. doi: https://doi.org/10.1007/s10845-021-01772-5.

[12] A. Uglanov, F. Campean, A. Abdullatiff, D. Neagu, A. Doikin, D. Delaux, and P. Bonnaud, "An NLP-based framework for early identification of design reliability issues from heterogeneous automotive lifecycle data," *Procedia CIRP*, vol. 128, pp. 728–733, 2024. doi: https://doi.org/10.1016/j.procir.2024.05.098.

[13] National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1, 2023. doi: https://doi.org/10.6028/NIST.AI.100-1.

[14] D. B. Kim, M. S. Bajestani, J. Y. Lee, S.-J. Shin, G.-Y. Kim, S. M. M. Sajadieh, and S. Noh, "Human-in-the-loop in smart manufacturing (H-SM): A review and perspective," *Journal of Manufacturing Systems*, vol. 82, pp. 178–199, 2025. doi: https://doi.org/10.1016/j.jmsy.2025.05.020.

[15] M. P. Brundage, T. B. Sexton, M. Hodkiewicz, A. A. Dima, and S. Lukens, "Technical language processing: Unlocking maintenance knowledge," *Manufacturing Letters*, vol. 27, pp. 42–46, 2021. doi: https://doi.org/10.1016/j.mfglet.2020.11.001.

[16] Y. Li, Y. Liu, J. Zhang, L. Cao, and Q. Wang, "Automated analysis and assignment of maintenance work orders using natural language processing," *Automation in Construction*, vol. 165, article 105501, 2024. doi: https://doi.org/10.1016/j.autcon.2024.105501.

[17] E. Turanoglu Bekar, P. Nyqvist, and A. Skoogh, "An intelligent approach for data pre-processing and analysis in predictive maintenance with an industrial case study," *Advances in Mechanical Engineering*, vol. 12, no. 5, 2020. doi: https://doi.org/10.1177/1687814020919207.

[18] C. Woods, M. Selway, T. Bikaun, M. Stumptner, and M. Hodkiewicz, "An ontology for maintenance activities and its application to data quality," *Semantic Web*, vol. 15, no. 2, pp. 319–352, 2024. doi: https://doi.org/10.3233/SW-233299.

[19] J. Buddhakulsomsiri and A. Zakarian, "Sequential pattern mining algorithm for automotive warranty data," *Computers & Industrial Engineering*, vol. 57, no. 1, pp. 137–147, 2009. doi: https://doi.org/10.1016/j.cie.2008.11.006.

[20] S. Chen, J. P. González Sánchez, E. Turanoglu Bekar, J. Bokrantz, A. Skoogh, and P. V. Lopes, "Understanding stakeholder requirements for digital twins in manufacturing maintenance," in *Proceedings of the 2023 Winter Simulation Conference*, pp. 2008–2019, 2023. doi: https://doi.org/10.1109/WSC60868.2023.10408657.

[21] M. Stewart, M. Hodkiewicz, W. Liu, and T. French, "MWO2KG and Echidna: Constructing and exploring knowledge graphs from maintenance data," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 2024. doi: https://doi.org/10.1177/1748006X221131128.

[22] S. M. R. Naqvi, M. Ghufran, C. Varnier, J.-M. Nicod, K. Javed, and N. Zerhouni, "Unlocking maintenance insights in industrial text through semantic search," *Computers in Industry*, vol. 157–158, article 104083, 2024. doi: https://doi.org/10.1016/j.compind.2024.104083.

[23] J. Buddhakulsomsiri, Y. Siradeghyan, A. Zakarian, and X. Li, "Association rule-generation algorithm for mining automotive warranty data," *International Journal of Production Research*, vol. 44, no. 14, pp. 2749–2770, 2006. doi: https://doi.org/10.1080/00207540600564633.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017. url: https://arxiv.org/abs/1706.03762.

[25] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020. url: https://arxiv.org/abs/2005.14165.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019. url: https://aclanthology.org/N19-1423/.

[27] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. url: https://www.jmlr.org/papers/v21/20-074.html.

[28] S. Chen, S. Bandaru, S. Marti, E. Turanoglu Bekar, and A. Skoogh, "Comparison of unsupervised image anomaly detection models for sheet metal glue lines," *Engineering Applications of Artificial Intelligence*, vol. 153, article 110740, 2025. doi: https://doi.org/10.1016/j.engappai.2025.110740.

[29] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," in *Advances in Neural Information Processing Systems*, vol. 35, pp. 27730–27744, 2022. url: https://arxiv.org/abs/2203.02155.

[30] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems*, vol. 35, pp. 24824–24837, 2022. url: https://arxiv.org/abs/2201.11903.

[31] G. Mialon, R. Dessi, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Roziere, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz, E. Grave, Y. LeCun, and T. Scialom, "Augmented language models: A survey," *Transactions on Machine Learning Research*, 2023. url: https://openreview.net/forum?id=jh7wH2AzKK.

[32] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023. doi: https://doi.org/10.1145/3571730.

[33] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *ACM Transactions on Information Systems*, vol. 43, no. 2, pp. 1–55, 2025. doi: https://doi.org/10.1145/3703155.

[34] Z. Lin, S. Guan, W. Zhang, H. Zhang, Y. Li, and H. Zhang, "Towards trustworthy LLMs: A review on debiasing and dehallucinating in large language models," *Artificial Intelligence Review*, vol. 57, article 243, 2024. doi: https://doi.org/10.1007/s10462-024-10896-y.

[35] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, J. Teevan, R. Kikin-Gil, and E. Horvitz, "Guidelines for human-AI interaction," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2019. doi: https://doi.org/10.1145/3290605.3300233.

[36] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI Communications*, vol. 7, no. 1, pp. 39–59, 1994. doi: https://doi.org/10.3233/AIC-1994-7104.

[37] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of EMNLP-IJCNLP*, pp. 3982–3992, 2019. doi: https://doi.org/10.18653/v1/D19-1410.

[38] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of EMNLP*, pp. 6769–6781, 2020. doi: https://doi.org/10.18653/v1/2020.emnlp-main.550.

[39] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988. doi: https://doi.org/10.1016/0306-4573(88)90021-0.

[40] S. E. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009. doi: https://doi.org/10.1561/1500000019.

[41] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020. url: https://arxiv.org/abs/2005.11401.

[42] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "REALM: Retrieval-augmented language model pre-training," in *Proceedings of the 37th International Conference on Machine Learning*, pp. 3929–3938, 2020. url: https://arxiv.org/abs/2002.08909.

[43] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. van den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. de Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. Rae, E. Elsen, and L. Sifre, "Improving language models by retrieving from trillions of tokens," in *Proceedings of the 39th International Conference on Machine Learning*, PMLR, vol. 162, pp. 2206–2240, 2022. url: https://proceedings.mlr.press/v162/borgeaud22a.html.

[44] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," arXiv preprint arXiv:2312.10997, 2023. doi: https://doi.org/10.48550/arXiv.2312.10997.

[45] Y.-T. Tsai, "Applying a case-based reasoning method for fault diagnosis during maintenance," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 223, no. 10, pp. 2431–2441, 2009. doi: https://doi.org/10.1243/09544062JMES1588.

[46] M. L. Hoffman, E. Y. Song, M. P. Brundage, and S. Kumara, "Online maintenance prioritization via Monte Carlo tree search and case-based reasoning," *Journal of Computing and Information Science in Engineering*, vol. 22, no. 4, article 041005, 2022. doi: https://doi.org/10.1115/1.4053408.

[47] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2020. doi: https://doi.org/10.1109/TPAMI.2018.2889473.

[48] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pp. 380–388, 2002. doi: https://doi.org/10.1145/509907.509965.

[49] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021. doi: https://doi.org/10.1109/TBDATA.2019.2921572.

[50] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011. doi: https://doi.org/10.1109/TPAMI.2010.57.

[51] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pp. 604–613, 1998. doi: https://doi.org/10.1145/276698.276876.

[52] S. Chen, E. Turanoglu Bekar, J. Bokrantz, and A. Skoogh, "AI-enhanced digital twins in maintenance: Systematic review, industrial challenges, and bridging

research-practice gaps," *Journal of Manufacturing Systems*, vol. 82, pp. 678–699, 2025. doi: https://doi.org/10.1016/j.jmsy.2025.07.006.

[53] M. Zaharia, O. Khattab, L. Chen, J. Q. Davis, H. Miller, C. Potts, J. Zou, M. Carbin, J. Frankle, N. Rao, and A. Ghodsi, "The shift from models to compound AI systems," Berkeley Artificial Intelligence Research Blog, 2024. url: https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/.

[54] P. Zhao, Z. Jin, and N. Cheng, "An in-depth survey of large language model-based artificial intelligence agents," arXiv preprint arXiv:2309.14365, 2023. doi: https://doi.org/10.48550/arXiv.2309.14365.

[55] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations*, 2023. url: https://arxiv.org/abs/2210.03629.

[56] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems*, vol. 36, 2023. url: https://proceedings.neurips.cc/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

[57] X. Huang, W. Liu, X. Chen, X. Wang, H. Wang, D. Lian, Y. Wang, R. Tang, and E. Chen, "Understanding the planning of LLM agents: A survey," arXiv preprint arXiv:2402.02716, 2024. doi: https://doi.org/10.48550/arXiv.2402.02716.

[58] A. Madaan, N. Tandon, P. Clark, and Y. Yang, "Self-Refine: Iterative refinement with self-feedback," in *Advances in Neural Information Processing Systems*, vol. 36, 2023. url: https://arxiv.org/abs/2303.17651.

[59] N. Shinn, F. Cassano, A. Gopinath, K. R. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 36, 2023. url: https://arxiv.org/abs/2303.11366.

[60] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011. url: https://researchnow.flinders.edu.au/en/publications/evaluation-from-precision-recall-and-f-measure-to-roc-informednes/.

[61] N. Chinchor, "MUC-4 evaluation metrics," in *Fourth Message Understanding Conference (MUC-4): Proceedings of a Conference Held in McLean, Virginia, June 16–18, 1992*, 1992. url: https://aclanthology.org/M92-1002/.

[62] A. Khan, R. Nahar, H. Chen, G. E. Constante Flores, and C. Li, "FaultExplainer: Leveraging large language models for interpretable fault detection and diagnosis," *Computers & Chemical Engineering*, vol. 199, p. 109152, 2025. doi: https://doi.org/10.1016/j.compchemeng.2025.109152.

[63] J. Chen, R. Huang, Z. Lv, J. Tang, and W. Li, "FaultGPT: Industrial fault diagnosis question answering system by vision language models," *arXiv preprint*

arXiv:2502.15481, 2025. doi: https://doi.org/10.48550/arXiv.2502.15481.

[64] A. K. S. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical Systems and Signal Processing*, vol. 20, no. 7, pp. 1483–1510, 2006. doi: https://doi.org/10.1016/j.ymssp.2005.09.012.

[65] R. X. Gao, J. Kruger, M. Merklein, H.-C. Möhring, and J. Váncza, "Artificial Intelligence in manufacturing: State of the art, perspectives, and future directions," *CIRP Annals*, vol. 73, no. 2, pp. 723–749, 2024. doi: https://doi.org/10.1016/j.cirp.2024.04.101.

[66] D. Neupane, M. R. Bouadjenek, R. Dazeley, and S. Aryal, "Data-driven machinery fault detection: A comprehensive review," *arXiv preprint arXiv:2405.18843*, 2024. doi: https://doi.org/10.48550/arXiv.2405.18843.

[67] A. Sureka, S. De, and K. V. Indukuri, "Mining automotive warranty claims data for effective root cause analysis," in *Proc. 13th International Conference on Database Systems for Advanced Applications (DASFAA)*, pp. 621–626, 2008. doi: https://doi.org/10.1007/978-3-540-78568-2_54.

[68] S. Wu, "Warranty data analysis: A review," *Quality and Reliability Engineering International*, vol. 28, no. 8, pp. 795–805, 2012. doi: https://doi.org/10.1002/qre.1282.

[69] J. Lim, B. Vogel-Heuser, and I. Kovalenko, "Large language model-enabled multi-agent manufacturing systems," *arXiv preprint arXiv:2406.01893*, 2024. doi: https://doi.org/10.48550/arXiv.2406.01893.

[70] J. Xu, Q. Zhang, Z. Zhong, S. He, C. Zhang, Q. Lin, D. Pei, P. He, D. Zhang, and Q. Zhang, "OpenRCA: Can large language models locate the root cause of software failures?," in *Proc. 13th International Conference on Learning Representations (ICLR)*, 2025. Available: https://openreview.net/forum?id=M4qNIzQYpd.

[71] T. Kim, W. Park, H. Yun, and K. Lee, "Why do AI agents systematically fail at cloud root cause analysis?" *arXiv preprint arXiv:2602.09937*, 2026. doi: https://doi.org/10.48550/arXiv.2602.09937.

[72] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2024. doi: https://doi.org/10.48550/arXiv.2312.10997.

[73] E. Hastings, T. B. Sexton, M. P. Brundage, and M. Hodkiewicz, "Agreement behavior of isolated annotators for maintenance work-order data mining," in *Proceedings of the 2019 Annual Conference of the Prognostics and Health Management Society*, Scottsdale, AZ, USA, 2019. url: https://www.nist.gov/publications/agreement-behavior-isolated-annotators-maintenance-work-order-data-mining.

A

Prompt and State Examples

This appendix documents representative prompt and state examples from the local `failing_part_agent` artifact. The examples are taken from the prompt builders in `app/llm/prompts.py`, the agent implementations in `app/agents/`, the shared state model in `app/state.py`, and an intermediate batch output in `data/engine_exch_agent_result_10.csv`. Case identifiers, report numbers, vehicle identifiers, supplier information, dates, and administrative fields are removed. Technical wording is preserved only where it is needed to show how the workflow handles diagnostic evidence.

A.1 Shared Diagnostic State

The diagnostic artifact uses one mutable state object for each repair case. Every specialist operation reads from this state and writes back only the evidence it is responsible for. The state acts as a trace of the route from repair text to final label.

Table A.1: Core fields in the shared diagnostic state.

State field	Role in the workflow
`raw_text`	Original maintenance narrative for one repair case.
`all_parts_in_job`	Parts-use context used by retrieval and later review.
`normalized_text`	Cleaned technical wording produced before evidence extraction.
`extracted_signals`	Symptoms, explicit failures, repair actions, component mentions, and confidence notes.
`hypotheses`	Candidate engineering failure concepts before mapping to business labels.
`critic_notes`	Warnings about repair-action confusion, weak support, or consequential damage.
`mapped_candidates`	Canonical label candidates with score and reason.
`retrieval_results`	Similar historical cases used as comparison evidence.
`final_answer` and `confidence`	Controlled output selected by the final judge.
`history`	Ordered record of agent outputs for review and debugging.

Listing A.1 shows an anonymized state excerpt from a case where the repair text contained cooling-system pressure, coolant loss, a positive cylinder-head-gasket test, cylinder-head removal for damage assessment, and later engine replacement. The

example is useful because the repair action could easily point towards a broad engine exchange label, while the diagnostic evidence points to a sealing failure.

Listing A.1: Anonymized shared state excerpt for one repair case.

```
{
  "raw_text": "[redacted workshop text: pressure in cooling system; coolant loss; positive
      cylinder-head-gasket reactivity test; cylinder head removed for damage assessment; engine
      replacement required]",
  "all_parts_in_job": "[redacted parts context: coolant, engine exchange, gasket set, sealing
      parts, fasteners]",
  "normalized_text": "Pressure in cooling system or coolant loss. Fault finding performed. Cooling
       system pressure test OK. Cylinder head gasket test using reactivity method positive. EGR
      cooler and charge air cooler checked OK. Cylinder head removed for damage assessment.
      Engine replacement required. Subsequently tested OK.",
  "extracted_signals": {
    "symptoms": ["pressure in cooling system", "coolant loss"],
    "explicit_failures": ["cylinder head gasket failure confirmed by positive reactivity test"],
    "repair_actions": ["cooling system pressure test", "cylinder head gasket test", "EGR cooler
        and charge air cooler check", "cylinder head removed", "engine replacement"],
    "consequential_damage": ["possible engine damage requiring complete replacement"],
    "key_patterns": ["positive gasket test", "cooling system pressure", "cylinder head removal", "
        engine replacement after damage assessment"],
    "components_mentioned": ["cooling system", "cylinder head gasket", "EGR cooler", "charge air
        cooler", "cylinder head", "engine"],
    "confidence_notes": ["gasket failure is explicit; engine replacement may be repair escalation
        rather than primary failure"]
  },
  "mapped_candidates": [
    {
      "label": "GASKET",
      "score": 95.0,
      "reason": "Positive gasket test and coolant-pressure symptoms support primary sealing failure
          ; engine replacement is treated as later repair escalation."
    }
  ],
  "final_answer": "GASKET",
  "confidence": 0.92
}
```

A.2 Planner Routing Agent

The planner is the control layer that decides which specialist operation should run next. In the implementation, the routing prompt receives a compact snapshot instead of the full repair record. The snapshot tells the planner whether the state already contains normalized text, extracted signals, hypotheses, critic notes, mapped candidates, and recent history. This design keeps routing focused on process state rather than on final diagnosis.

Listing A.2: Planner prompt excerpt adapted from the routing prompt builder.

```
System:
You are PlannerAgent for a multi-agent diagnostic system.
Read shared state and choose ONE best next agent dynamically.
Do not run a fixed sequence.
Return JSON only with keys: next_agent, reason, goal.

Allowed next_agent values:
TextNormalizerAgent, SignalExtractorAgent, HypothesisBuilderAgent,
CriticAgent, MapperAgent, FinalJudgeAgent.

Important routing rule:
If the case contains overheating, coolant loss, water loss, cylinder leak checks,
```

```
cylinder head removal, CO test, pressure build-up, or sealing-related diagnostic
context, do not skip critical review before finalization when engine exchange is
a serious candidate.

User:
Current shared state snapshot:
{
  "current_step": 2,
  "has_normalized_text": true,
  "has_extracted_signals": false,
  "hypothesis_count": 0,
  "critic_count": 0,
  "mapped_count": 0,
  "history_tail": [
    {"agent": "TextNormalizerAgent", "output": {"normalized_text": "..."}}
  ]
}

Choose the next best agent for this state now.
```

For the state shown above, the routing output is expected to move the workflow from normalized text to structured evidence. A representative output is shown in Listing A.3. The important point is that the planner does not choose a final label. It only selects the next operation and records a short goal for that operation.

Listing A.3: Planner output example.

```
{
  "next_agent": "SignalExtractorAgent",
  "reason": "Normalized technical text is available, but no structured diagnostic signals have
      been extracted.",
  "goal": "Extract symptoms, explicit failures, repair actions, component mentions, and confidence
       notes before hypothesis formation."
}
```

A.3 Signal Extraction Agent

The signal extractor turns normalized repair text into structured diagnostic evidence. Its prompt explicitly prevents early mapping to canonical business labels. This separation matters because the same component can appear as a symptom source, an inspected component, a repair action, consequential damage, or the final attributed part.

Listing A.4: Signal extractor prompt excerpt.

```
System:
You are SignalExtractorAgent.
Extract structured evidence from normalized repair text.
Return JSON only with keys:
symptoms, explicit_failures, repair_actions, consequential_damage,
key_patterns, components_mentioned, confidence_notes.

Use arrays where possible.
Keep uncertain items explicit in confidence_notes.
Do NOT map components to canonical business labels in this stage.
Extract raw component mentions only.

User:
Use normalized text first; reference raw text if needed.

Normalized text:
Pressure in cooling system or coolant loss. Cooling system pressure test OK.
```

```
Cylinder head gasket test using reactivity method positive. EGR cooler and
charge air cooler checked OK. Cylinder head removed for damage assessment.
Engine replacement required. Subsequently tested OK.

Focus on extracting:
- symptoms actually observed
- explicit diagnosed defects
- repair actions
- possible consequential damage
- component mentions
- important diagnostic patterns such as pressure build-up, CO test,
  cylinder leak checks, head removal, turbo defect, and coolant loss
```

Listing A.5 gives the corresponding style of structured output. The extracted evidence keeps the positive gasket test and the later engine replacement separate. That separation gives later stages a clear basis for distinguishing primary failure from repair escalation.

Listing A.5: Signal extractor output example.

```
{
  "symptoms": [
    "pressure in cooling system",
    "coolant loss"
  ],
  "explicit_failures": [
    "cylinder head gasket failure confirmed by positive reactivity test"
  ],
  "repair_actions": [
    "cooling system pressure test",
    "cylinder head gasket test using reactivity method",
    "EGR cooler and charge air cooler check",
    "cylinder head removed for damage assessment",
    "engine replacement",
    "subsequently tested OK"
  ],
  "consequential_damage": [
    "possible engine damage requiring complete replacement"
  ],
  "key_patterns": [
    "cooling system pressure test OK",
    "cylinder head gasket test positive",
    "cylinder head removal for damage assessment",
    "engine replacement after damage assessment"
  ],
  "components_mentioned": [
    "cooling system",
    "cylinder head gasket",
    "EGR cooler",
    "charge air cooler",
    "cylinder head",
    "engine"
  ],
  "confidence_notes": [
    "gasket failure is directly supported by the positive test",
    "engine replacement suggests repair escalation or consequential damage"
  ]
}
```

A.4 Knowledge Base Injection

The knowledge base used by the mapper is stored separately from the prompt templates. It is not a collection of complete repair cases. Manual review first turns

individual examples into reusable diagnostic patterns. For instance, a case that mentions coolant loss, pressure build-up, a positive sealing test, and later engine replacement is summarized as a sealing-failure pattern. The JSON entry stores the canonical label, the conditions that support it, the conditions that should weaken it, and a priority used during selection.

Listing A.6 shows a shortened entry adapted from `app/knowledge/part_kb.json`. The structure is intentionally compact: `keywords` describe evidence that can trigger the entry, while `negative_rules` prevent the same label from being selected when the surface wording points to another primary cause or only to a repair action.

Listing A.6: Simplified JSON knowledge-base entry used for selective prompt injection.

```
[
  {
    "label": "GASKET",
    "description": "General gasket or cylinder-head-gasket related sealing failure. Common in
        coolant loss, pressure build-up, positive coolant tests, and cylinder-head removal for
        diagnosis. Severe downstream engine damage or engine replacement can still belong to this
        label when sealing evidence is primary.",
    "keywords": [
      "coolant loss",
      "water loss",
      "high pressure in cooling system",
      "co test positive",
      "cooling system pressure test",
      "cylinder head gasket",
      "head gasket",
      "cylinder head removed",
      "no external leaks"
    ],
    "negative_rules": [
      "turbocharger explicitly defective as primary cause",
      "coolant loss clearly caused by water pump leakage",
      "coolant loss clearly caused by hose or coolant pipe leakage",
      "engine replacement alone without sealing-failure context"
    ],
    "priority": 0.99
  }
]
```

At runtime, the loader scores entries by keyword overlap with the current case and adds the priority value. Only selected entries are injected into the mapper prompt. Listing A.7 shows the form of the injected context. The mapper receives selected knowledge together with the current state, allowed labels, and policy rules; it does not receive the full knowledge base.

Listing A.7: Mapper prompt excerpt showing selected KB injection.

```
Map candidates using state + selected KB + policy:

State:
{
  "normalized_text": "... coolant loss ... pressure test ... cylinder head removed ... engine
      replacement ...",
  "extracted_signals": {
    "symptoms": ["coolant loss", "pressure in cooling system"],
    "explicit_failures": ["positive sealing-related diagnostic test"],
    "repair_actions": ["cylinder head removed", "engine replacement"],
    "consequential_damage": ["possible engine damage"]
  },
  "hypotheses": [
    {
```

```
      "failure_concept": "sealing failure with downstream engine damage",
      "score": 0.95
    }
  ]
}

Selected KB entries:
{
  "kb": [
    {
      "label": "GASKET",
      "description": "General gasket or cylinder-head-gasket related sealing failure...",
      "keywords": ["coolant loss", "pressure test", "cylinder head removed"],
      "negative_rules": ["engine replacement alone without sealing-failure context"],
      "priority": 0.99
    }
  ]
}

Allowed labels:
["GASKET", "TURBOCHARGER", "WATER PUMP", "..."]

Policy:
R1: Repair action alone must not determine failing part.
R2: Use exchange labels only when the replaced unit itself is the attributed failing part.
R3: Distinguish primary failure from consequential damage.
R9: Mapper and FinalJudge must enforce dictionary constraints strictly.
```

The mapper output can be checked because the selected knowledge entry is visible next to the mapped candidate. In the example above, the knowledge entry does not prove the label by itself. It explains why sealing evidence should remain competitive even when a later repair action mentions engine replacement.

A.5 Final Judgement Agent

The final judge selects one canonical label from the allowed dictionary or returns `unknown`. Its prompt is deliberately conservative. It tells the agent to use the accumulated state, mapped candidates, critic output, and policy rules, rather than starting diagnosis again from the raw text. It also lowers confidence when evidence is indirect, mixed, or weak.

Listing A.8: Final judge prompt excerpt.

```
System:
You are FinalJudgeAgent.
Choose exactly one final label from the allowed dictionary or unknown.
Do not bypass shared state.
Do not re-reason from scratch.
Base your decision on existing shared state, mapped candidates, critic output,
and policy rules.

Decision rules:
1) Final answer must be exactly one allowed label or unknown.
2) Do not select engine exchange only because the engine was damaged or replaced.
3) In overheating, coolant-loss, CO-test-positive, pressure-build-up,
   cylinder-leak, or cylinder-head-removal context, prefer gasket over engine
   exchange unless the engine unit itself is explicitly identified as the
   primary failed unit.
4) Treat engine damage or engine replacement as possible consequential damage
   unless the text explicitly attributes the engine unit itself as root cause.
5) If the final label is supported mainly by indirect evidence, avoid extremely
   high confidence.
```

```
User:
Finalize from shared state only:
{
  "extracted_signals": {"symptoms": ["pressure in cooling system", "coolant loss"], "...": "..."},
  "hypotheses": [
    {
      "failure_concept": "cylinder head gasket failure with combustion gas leakage into cooling
          system",
      "score": 95.0
    }
  ],
  "critic_notes": [
    {
      "issues": ["engine replacement may be consequential damage"],
      "confidence_warnings": ["avoid over-attributing engine exchange"]
    }
  ],
  "mapped_candidates": [
    {"label": "GASKET", "score": 95.0, "reason": "positive gasket test and coolant-pressure
        symptoms"}
  ]
}
```

The batch output for the anonymized case stores `GASKET` as the selected label with confidence `0.92`. Listing A.9 shows the form of the final judge output that is kept in the interactive state history. The reason is short by design; longer explanations stay distributed across extracted signals, hypotheses, critic notes, and mapped candidates.

Listing A.9: Final judge output example.

```
{
  "final_answer": "GASKET",
  "confidence": 0.92,
  "reason": "The strongest evidence is the positive cylinder-head-gasket reactivity test together
      with pressure in the cooling system and coolant loss. Engine replacement is treated as a
      repair escalation after damage assessment, not as primary engine-unit failure."
}
```

Together, these examples show how the artifact keeps the diagnosis easy to check. The planner decides which operation is needed, the signal extractor records diagnostic evidence without mapping it too early, and the final judge selects a controlled label only after candidate evidence and constraints have been stored in shared state.

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY