Multi-Agent System for Diagnosing Defects on a Car Assembly Line

Felipe Merenda Izidorio - 64379

Thesis presented to the School of Technology and Management in the scope of the Master in Informatics.

Supervisors:

Prof. Paulo Jorge Pinto Leitão

Prof. José Fernando Lopes Barbosa

Prof. Gleifer Vaz Alves

This document does not include the suggestions made by the board.

Bragança

2024-2025

Multi-Agent System for Diagnosing Defects on a Car Assembly Line

Felipe Merenda Izidorio - 64379

Thesis presented to the School of Technology and Management in the scope of the Master in Informatics.

Supervisors:

Prof. Paulo Jorge Pinto Leitão

Prof. José Fernando Lopes Barbosa

Prof. Gleifer Vaz Alves

This document does not include the suggestions made by the board.

Bragança

2024-2025

Dedication

I dedicate this work to God for His blessings and guidance. I would also like to express my gratitude to my mother, Veridiana Bellini Merenda, my father, Ricardo Izidorio, my brother, Rafael Merenda Izidorio, and all my family who believed in me from the very beginning. Finally, I dedicate this work to my friends who supported me throughout this journey.

Acknowledgment

This work was partially supported by the HORIZON-CL4-2021-TWIN-TRANSITION-01 openZDM project, under Grant Agreement No. 101058673. Also, it was supported by national funds through FCT/MCTES (PIDDAC): CeDRI, UIDB/05757/2020 (DOI: 10.54499/UIDB/05757/2020) and UIDP/05757/2020 (DOI: 10.54499/UIDP/05757/2020); and SusTEC, LA/P/0007/2020 (DOI: 10.54499/LA/P/0007/2020).

I would like to express my sincere gratitude to my supervisor, Prof. Paulo Leitão, for his guidance, support, and expertise throughout this research. I also extend my thanks to my co-supervisors, Prof. José Barbosa and Prof. Gleifer Alves, for their valuable contributions, insightful feedback, and continuous support during the development of this work.

I am deeply grateful to Prof. Rui Pedro for his technical insights, which significantly contributed to the advancement of this research.

Special acknowledgment to Júlio César Guimarães Costa, whose foundational work on zero-defect manufacturing systems provided the basis upon which this dissertation was built. I am grateful to the automotive manufacturing company that provided access to real production data under confidentiality agreements, enabling the validation of the proposed multi-agent architecture in an authentic industrial context.

I also want to thank the openZDM consortium, especially those representing the vehicular industry, for providing useful shop-floor knowledge.

Abstract

Traditional approaches to diagnosing geometric defects in automotive assembly lines are based on isolated methods, which have limitations in terms of robustness and early detection of anomalies. This dissertation presents a hierarchical multi-agent architecture for collaborative defect diagnosis, organized into three layers: Point Agents perform local analysis by applying multiple diagnostic algorithms; Station Agents coordinate groups of agents within each station; Inter-Station Agent provides a systemic view by identifying correlations between stations. Coordination uses correlation-based clustering and leader election, enabling efficient aggregation of diagnostics. Communication flows hierarchically and laterally between correlated agents. This organization provides scalability, modularity, and robustness by confining local failures. Experimental validation demonstrates that the collaborative architecture achieves superior accuracy compared to isolated methods, showing that the complementarity between distributed algorithms provides more robust diagnostics and early warning capabilities.

Keywords: Multi-Agent Systems, Distributed Architecture, Industrial Diagnosis, Machine Learning, Industry 4.0

Resumo

Abordagens tradicionais para diagnóstico de defeitos geométricos em linhas de montagem automotivas baseiam-se em métodos isolados, apresentando limitações em robustez e detecção precoce de anomalias. Esta dissertação apresenta uma arquitetura multi-agente hierárquica para diagnóstico colaborativo de defeitos, organizada em três camadas: Point Agents realizam análise local aplicando múltiplos algoritmos diagnósticos; Station Agents coordenam grupos de agentes dentro de cada estação; Inter-Station Agent proporciona visão sistémica identificando correlações entre estações. A coordenação utiliza clustering baseado em correlações e eleição de líderes, permitindo agregação eficiente de diagnósticos. A comunicação flui hierarquicamente e lateralmente entre agentes correlacionados. Esta organização proporciona escalabilidade, modularidade e robustez pelo confinamento de falhas locais. A validação experimental demonstra que a arquitetura colaborativa alcança precisão superior comparada a métodos isolados, evidenciando que a complementaridade entre algoritmos distribuídos proporciona diagnósticos mais robustos e capacidade de alerta precoce.

Palavras-chave: Sistemas Multi-Agente, Arquitetura Distribuída, Diagnóstico Industrial, Aprendizagem de Máquina, Indústria 4.0.

Contents

1 Introduction **1**

1.1 Contextualization 2

1.2 Objectives 3

1.3 Document Structure 4

2 State of the Art **7**

2.1 Industrial Diagnosis Systems 7

2.2 Machine Learning for Industrial Diagnosis 9

2.3 Multi-Agent Systems in Industrial Applications 10

2.4 Towards Integration of MAS and ML for Diagnosis 13

3 Case Study **17**

3.1 Automotive Assembly Line Context 17

3.1.1 Automated Measurement System 18

3.1.2 Inspection Stations 19

3.1.3 Challenges and Opportunities 19

3.2 Data Sources and Description 21

3.2.1 Data Structure 21

3.2.2 Interpretation of Limits and Classification Zones 23

3.2.3 Relevant Features for the Multi-Agent System 23

3.2.4 Potential Diagnostic Algorithms 24

3.3 Summary 25

4 Multi-Agent System Architecture **27**

4.1 Overview of the Architecture . . . 27

4.2 Detailed Components and Operation . . . 29

4.2.1 Agents: Types and Responsibilities . . . 30

4.2.2 Communication Protocols . . . 34

4.2.3 Correlation and Clustering Strategy . . . 39

4.2.4 Diagnosis Algorithms . . . 43

4.2.5 System Reports and Decision Support . . . 47

4.3 Workflow and Data Flow . . . 50

4.3.1 Main Data Flow (Intra-Station) . . . 51

4.3.2 Inter-Station Flow . . . 52

4.3.3 Example of Complete Execution . . . 52

5 Experimental Setup and Results **55**

5.1 Experimental Setup . . . 55

5.1.1 Environment and Tools . . . 56

5.1.2 Diagnosis Algorithms . . . 58

5.1.3 Summarization Techniques . . . 60

5.1.4 Experiment Design . . . 65

5.2 Results . . . 74

5.2.1 Diagnostic Validation with Classification Anchor . . . 75

5.2.2 Early Warning Capability . . . 76

5.2.3 Complementarity between Algorithms . . . 78

5.2.4 Computational Performance and Latency . . . 80

5.2.5 Cluster Formation and Effectiveness . . . 82

5.2.6 Comparison of Configuration Profiles and Trade-offs . . . 83

5.3 Results Discussion . . . 85

5.3.1 Trade-offs Between Configuration Profiles . . . 85

5.3.2 Limitations of Individual Diagnostic Algorithms . . . 86

5.3.3 Quantitative Analysis of Collaborative Overhead 88
5.3.4 State of the Art Positioning 89

6 Conclusions **93**
6.1 Summary of Contributions 93
6.2 Practical Implications for Industry 95
6.3 Limitations of the Study 96
6.4 Directions for Future Work 97
6.5 Final Considerations 100

A Appendix: Implementation Availability **A1**

List of Tables

5.1 Comparison of Multi-Agent System Configuration Profiles 66
5.2 Detailed validation metrics for Anomaly Detection using Classification as anchor 76
5.3 Early warning capability statistics across configuration profiles 78
5.4 Detailed counts of complementarity scenarios across profiles 79
5.5 Detailed latency statistics across configuration profiles 81
5.6 Distribution of confidence levels in cluster reports 83
5.7 Comprehensive comparison of configuration profiles across all evaluated metrics 84
5.8 Computational overhead analysis by configuration profile 88

List of Figures

2.1 Properties of MAS for industrial applications. 12

2.2 Research landscape for diagnostic approaches in industrial applications. . . 16

3.1 Sequence of inspection stations analyzed in the body shop area. 18

3.2 Schematic representation of the gap and flush parameters. 19

3.3 Measurement classification zones based on specified limits. 23

4.1 Overview of the multi-agent system architecture, showing the hierarchical organization in three layers: Inter-Station Layer, Station Agents, and Point Agents. 28

4.2 State diagram of the Station Agent showing data processing flows, correlation analysis, and report generation. 31

4.3 State diagram of the Point Agent illustrating the data analysis flow and cluster communication mechanisms, including leader election and role-specific behaviors. 32

4.4 State diagram of the Inter-Station Agent, highlighting its cyclic correlation analysis process. 33

4.5 Sequence diagram of general system communication, showing interactions between the three types of agents. 35

4.6 Sequence diagram of the leader election protocol and communication inside the cluster. 36

4.7 Overview of the correlation and clustering algorithm, showing the transformation of agent data into a correlation matrix, adjacency graph, and final clusters. 40
4.8 Architecture of the diagnostic algorithm system, showing the base abstract classes, the algorithm registry, and the analytical results structure. 44
4.9 Hierarchical architecture of the reporting system 47
4.10 Summarization architecture 49

5.1 Overview of the case study with the multi-agent system architecture. . . . 69
5.2 Diagnostic validation metrics comparing the performance of Anomaly Detection against the Classification anchor across three configuration profiles. 75
5.3 Early warning capability through configuration profiles. 77
5.4 Distribution of complementarity scenarios across configuration profiles. . . 78
5.5 Comparison of operational latency between baseline (parallel processing only) and full MAS (including collaborative features) across profiles. 80
5.6 Distribution of cluster sizes across configuration profiles. 82
5.7 Multi-dimensional performance comparison across configuration profiles. . . 83

Acronyms

AI Artificial Intelligence.

ANN Artificial Neural Network.

CNN Convolutional Neural Network.

CPS Cyber-Physical System.

DBSCAN Density-Based Spatial Clustering of Applications with Noise.

ERP Enterprise Resource Planning.

ESTiG Escola Superior de Tecnologia e Gestão.

EW Early Warning.

FDD Fault Detection and Diagnosis.

FIPA Foundation for Intelligent Physical Agents.

FN False Negative.

FP False Positive.

GRU Gated Recurrent Unit.

IMHM Intelligent Machinery Health Management.

IoT Internet of Things.

IPB Instituto Politécnico de Bragança.

JSN Job Sequence Number.

JSON JavaScript Object Notation.

LSTM Long Short-Term Memory.

MAD Median Absolute Deviation.

MAS Multi-Agent System.

MES Manufacturing Execution System.

ML Machine Learning.

OPC UA OPC Unified Architecture.

RF Random Forest.

RNN Recurrent Neural Network.

RUL Remaining Useful Life.

SPADE Smart Python Agent Development Environment.

SPC Statistical Process Control.

SVM Support Vector Machine.

TN True Negative.

TP True Positive.

XMPP Extensible Messaging and Presence Protocol.

Chapter 1

Introduction

This chapter introduces the research context, motivations, and objectives of this dissertation on multi-agent systems for fault diagnosis in automotive assembly lines. The advent of Industry 4.0 has transformed manufacturing through intelligent automation and advanced analytical techniques, yet contemporary diagnostic approaches face significant limitations when addressing the complexity of modern production environments. This work proposes a novel Multi-Agent System architecture that integrates multiple machine learning techniques in a collaborative framework to overcome these limitations.

Section 1.1 establishes the industrial and technological context that motivates this research, examining how current diagnostic approaches fall short in addressing the demands of modern automotive manufacturing. Section 1.2 presents the main objective and specific contributions of this work, detailing the multi-agent architecture developed and its application to geometric defect detection in automotive body shop operations. Finally, Section 1.3 outlines the organization of this dissertation, providing a roadmap for the chapters that follow.

1.1 Contextualization

The advent of Industry 4.0 has fundamentally transformed manufacturing processes through the integration of cyber-physical systems, artificial intelligence, and machine learning techniques. This paradigm shift has significantly enhanced automation capabilities and operational efficiency across production lines, particularly in the automotive industry where precision and quality control are paramount.

In modern manufacturing environments, the ability to perform accurate real-time diagnoses is essential for maintaining product quality and ensuring production continuity. However, contemporary assembly lines present increasingly complex challenges due to their sophisticated operations, where defects can emerge at various production stages and stem from multiple interrelated factors.

Traditional diagnostic approaches, typically based on isolated analyses or single-method evaluations, have proven inadequate for addressing this complexity. As highlighted by [1], fault detection in automotive manufacturing is currently limited to simple boundary checking and limit checking techniques, which are slow to react to changes and fail to identify complex failures due to their inability to consider correlations between multiple variables or temporal changes. Furthermore, [2] emphasize that single-method fault detection techniques lack the versatility needed for diverse industrial applications, as they focus on specific sectors or methodologies and may only detect evident process characteristics while leaving behind latent defects. Additionally, the analysis of complex nonlinear signals often requires manual intervention by trained operators, limiting scalability and introducing potential human error [1].

Multi-Agent Systems (MAS) offer a promising solution to these challenges by enabling the integration of diverse diagnostic techniques within a collaborative framework. In a MAS architecture, each autonomous agent employs a distinct machine learning algorithm to analyze data from the assembly line and generate independent diagnostic assessments. Through agent interaction and collaboration, the system synthesizes these individual diagnoses to identify the most reliable conclusion, thereby leveraging the complementary

strengths of different analytical approaches.

This collaborative diagnostic strategy is particularly valuable in automotive body shop operations, where geometric defects such as gap (spacing between adjacent parts) and flush (alignment of surfaces) must be detected with high precision on Body-in-White structures. These defects, if undetected during the welding and assembly phases, can compromise both aesthetic quality and structural integrity of vehicles. The body shop represents a critical control point where such defects must be identified and addressed before the painting process.

By implementing a MAS-based diagnostic system, manufacturers can achieve superior detection accuracy while simultaneously improving product quality, increasing productivity, and reducing production costs through early defect identification and targeted intervention. As demonstrated by [1], advanced approaches combining multiple analytical perspectives can successfully diagnose and locate multiple classes of faults under real-time working conditions, outperforming established single-method fault detection and isolation techniques.

1.2 Objectives

The main objective of this work is to develop and validate a Multi-Agent System architecture for collaborative fault diagnosis in automotive body shop assembly lines, specifically targeting the detection of geometric defects in Body-in-White structures.

To achieve this main objective, the following specific objectives were established:

- **Design a multi-agent architecture** that enables distributed analysis and collaborative decision-making across multiple measurement stations in the body shop, allowing autonomous agents to work independently while maintaining coordination capabilities;
- **Develop a diagnostic framework** that supports the integration of multiple machine learning techniques, allowing different analytical approaches to be combined

within the multi-agent system and enabling each agent to employ algorithms appropriate for its diagnostic task;

- **Implement coordination mechanisms** that enable agents to identify relationships between measurement points, organize collaborative groups, and communicate effectively to share diagnostic information across the distributed system;

- **Create diagnostic fusion strategies** that synthesize individual diagnoses from multiple agents into consolidated assessments, establishing criteria for evaluating reliability and selecting the most trustworthy diagnosis when multiple agents provide different conclusions;

- **Validate the proposed system** using real production data from automotive body shop operations, demonstrating the system's capability to detect gap and flush defects in Body-in-White structures and evaluating its performance compared to individual machine learning techniques operating in isolation.

This approach aims to increase precision and reliability in the detection of geometric faults in automotive body shop operations by leveraging the complementary strengths of multiple analytical techniques coordinated through a distributed multi-agent architecture. The system addresses key limitations of traditional single-method approaches, including their inability to adapt to varying operational conditions, their lack of robustness against individual algorithm failures, and their limited capacity to exploit correlations between multiple measurement points across different stations in the production line.

1.3 Document Structure

This dissertation is organized into six chapters that progressively develop the proposed multi-agent diagnostic system from theoretical foundations to practical implementation and validation.

Chapter 1: Introduction establishes the research context, motivations, and objectives. It examines the limitations of current diagnostic approaches in automotive manufacturing and introduces the multi-agent paradigm as a promising solution for collaborative fault diagnosis.

Chapter 2: State of the Art provides a comprehensive review of relevant literature organized into four main areas: industrial diagnosis systems and their evolution from traditional to Industry 4.0 approaches; machine learning techniques applied to industrial diagnosis and their individual limitations; multi-agent systems in industrial applications and their distinctive properties; and efforts towards integrating multiple diagnostic techniques with multi-agent architectures for manufacturing diagnosis, identifying gaps that this dissertation addresses.

Chapter 3: Case Study describes the automotive body shop environment where the system was applied, detailing the measurement stations, data collection infrastructure, and characteristics of the geometric defects (gap and flush) that the system must detect in Body-in-White structures.

Chapter 4: Multi-Agent System Architecture presents the core contribution of this work: a detailed specification of the proposed multi-agent architecture. This chapter describes the hierarchical organization of agents (Point Agents, Station Agents, and Inter-Station Agent), communication protocols including distributed leader election mechanisms, correlation and clustering algorithms for identifying related measurement points, the extensible diagnostic algorithm framework supporting multiple machine learning techniques, and the hierarchical report structure with adaptive aggregation strategies for combining individual diagnoses into reliable assessments.

Chapter 5: Experiment Setup and Results documents the implementation of the proposed architecture, the machine learning algorithms integrated into the system, the experimental methodology used for validation with real body shop production data, and a comprehensive analysis of results demonstrating the system's diagnostic performance compared to individual techniques operating in isolation.

Chapter 6: Conclusion and Future Work synthesizes the main contributions of

this research, discusses the implications of the results for industrial practice, acknowledges limitations of the current work, and proposes directions for future research to extend and enhance the multi-agent diagnostic capabilities.

Throughout these chapters, the dissertation progressively builds from the theoretical foundations and state-of-the-art review toward the practical implementation and validation of a novel multi-agent system that addresses real industrial challenges in automotive body shop quality control.

Chapter 2

State of the Art

This chapter presents the state of the art in the development of Multi-Agent Systems for fault diagnosis in automotive assembly lines. The analysis is organized into four complementary sections: Section 2.1 examines the evolution of industrial diagnostic systems and the technical challenges in modern manufacturing environments; Section 2.2 analyzes machine learning techniques applied to industrial diagnostics and their limitations; Section 2.3 introduces Multi-Agent Systems and their applications in manufacturing; and Section 2.4 examines efforts towards integrating MAS and ML for diagnosis, identifying gaps that this dissertation aims to address.

2.1 Industrial Diagnosis Systems

Fault Detection and Diagnosis (FDD) systems aim to identify abnormal states in production processes and prevent future failures, and are essential for improving quality and performance in sectors such as semiconductors, automobiles, and chemical industries [3]. The evolution of these systems reflects a progression from techniques based on statistical pattern verification to contemporary approaches that exploit advanced sensing technologies, artificial intelligence, and non-destructive inspection.

Historically, FDD techniques were based on statistical process control (SPC) methods,

pre-established control limits, and predominantly manual inspection [3]. These conventional approaches have significant limitations in complex industrial processes, failing to identify latent characteristics due to the nonlinear dynamics of production systems. The deterministic nature of rule-based systems prevents adaptation to operational variations and hinders the identification of emerging failure modes.

With Industry 4.0, diagnostic systems have evolved through the integration of Zero Defect and Zero Waste strategies with non-destructive inspection technologies (NDITs) [4]. NDITs enable in-process or in-line inspection policies, increasing control by minimizing waste and optimizing production costs. Automation via robotic platforms equipped with multimodal sensors offers more comprehensive and consistent coverage compared to manual inspection. Recent developments include the fusion of NDITs with emerging technologies such as augmented reality, facilitating human-robot collaboration and improving quality assurance [4].

Modern systems face significant technical challenges. The increasing integration and complexity of production equipment present new challenges for data-driven monitoring [5]. Technologies such as Industrial Internet of Things (IIoT), Cyber-Physical Systems (CPS), and Artificial Intelligence, supported by cloud computing infrastructures, contribute to real-time diagnostic capabilities. Signal processing is compromised by stochastic noise and variability inherent in industrial environments, hindering robust extraction of discriminative features [6]. The integration of multiple subsystems and operating regimes increases the likelihood of concurrent failure modes. The fusion of heterogeneous data streams with deep learning architectures shows potential, but the optimization of network topologies and hyperparameters is a practical limitation, aggravated by challenges of explainability and generalization [5], [6].

The distinction between Industry 4.0 systems and conventional paradigms manifests itself through the pervasive incorporation of digital technologies, IoT, AI, and big data infrastructures [7]. Intelligent machinery health management (IMHM) encompasses prognostic maintenance, condition monitoring, remaining useful life (RUL) estimation, and

intelligent anomaly diagnosis. However, most methods assume the availability of complete, balanced, and abundant data, a premise that is often violated in real operational scenarios [7]. Predictive maintenance (PdM), although powered by IoT, faces barriers to achieving technological maturity, requiring advanced data science capabilities, integration of heterogeneous legacy systems, real-time massive data processing, cybersecurity guarantees, and adaptive models resilient to incomplete or noisy data [8].

This evolution reveals a trajectory of increasing sophistication accompanied by complex technical challenges, establishing the context for exploring advanced machine learning techniques as a means of overcoming the obstacles identified.

2.2 Machine Learning for Industrial Diagnosis

The limitations of traditional approaches have motivated the adoption of machine learning techniques for diagnosis in manufacturing. The ability to learn complex patterns, adapt to variations, and identify nonlinear relationships makes it particularly well suited to the challenges of modern production systems.

Fernandes et al. [9] conducted a systematic review of 44 studies (2015-2021) and found that most use artificial neural networks (ANNs), decision trees, hybrid models, or latent variable models. ANNs (12 studies) stand out for modeling complex relationships; decision trees (11 studies) offer interpretability; hybrid models (8 studies) combine techniques; and latent variable models (6 studies) reduce dimensionality.

Unsupervised approaches such as autoencoders and One-Class SVM demonstrate effectiveness in detecting anomalies without prior knowledge of the types of faults [10]. Autoencoders identify anomalies when the reconstruction error exceeds a threshold; One-Class SVM delimits the region of normal data, classifying external observations as anomalous.

Advantages include high performance, the ability to discover nonlinear relationships, and computational efficiency [9]. ANNs handle large datasets well; SVMs exhibit high

accuracy on small-to-medium datasets [11]. Automatic feature extraction reduces complexity while retaining relevant information [10].

However, individual machine learning techniques face significant limitations. Vithi and Chibaya [11] identified a strong dependence on high-quality data through a bibliometric review (2000-2024). Noise, missing values, or inaccurate measurements degrade performance, which is problematic in variable industrial environments. The limitation of labeled data is critical, as manual labeling is costly and requires expertise [12]. Failures are rare events, resulting in unbalanced datasets that create bias in the majority class.

Significant computational requirements restrict real-time applications or limited devices [11]. Limited adaptability is a concern: concept drift degrades performance when data distributions change [9]. Interpretability challenges reduce operator confidence and hinder validation [11]. Implementation complexities include integration with legacy systems, data infrastructure, and lifecycle management [11], [12].

No single machine learning technique is universally better. Performance depends on the characteristics of the problem, the nature of the data, and operational requirements. This observation motivates hybrid and ensemble models that combine the strengths of multiple techniques [12] The need to effectively integrate multiple techniques, with coordination, information sharing, and distributed decision-making, provides the motivation for more sophisticated architectures, a topic explored in the next section.

2.3 Multi-Agent Systems in Industrial Applications

The need to overcome limitations of individual machine learning methods, particularly in coordinating multiple analytical techniques, adaptability in dynamic environments, and distribution of processing capabilities, motivates the exploration of more sophisticated computational architectures. Multi-Agent Systems (MAS) emerge as a promising paradigm through their distributed, collaborative, and modular nature.

MASs are autonomous computational entities endowed with cognitive and social capabilities [13]. An agent is characterized by: (i) autonomy in local decision-making without

continuous centralized supervision, (ii) perception of the environment through sensors, (iii) capacity for action, and (iv) abilities to communicate and cooperate with other agents through interaction protocols.

In Industry 4.0 industrial cyber-physical systems, MASs distribute intelligence and data analysis capabilities across heterogeneous computational layers, supporting monitoring, diagnosis, prediction, and optimization tasks [14]. Agent-based modular architectures balance analytical capabilities across levels of the production system, from edge computing devices to cloud computing infrastructures, enabling low-latency local decisions while maintaining global analysis capabilities.

The distinctive properties of MAS for industrial applications (illustrated in Figure 2.1) include [13], [15]:

- **Decentralization** – eliminates single points of failure by distributing computational load and decisions across multiple nodes, providing operational robustness where the failure of one agent does not compromise the entire system;
- **Modularity** – facilitates the addition, removal, or modification of agents without architectural redesign, aligning with heterogeneous processes where different stations are represented by specialized agents that encapsulate domain knowledge;
- **Flexibility** – enables dynamic adaptation to changes in operating conditions or process requirements;
- **Scalability** – enables expansion through the incremental addition of new agents without restructuring the base system.

Crucially, MASs demonstrate native compatibility with emerging technologies, including machine learning [13], which is fundamental for integrating advanced analytical capabilities into distributed architectures where different agents employ distinct techniques adapted to their responsibilities. Practical implementations in automotive plants have demonstrated feasibility for zero-defect manufacturing strategies in multi-stage systems [14].

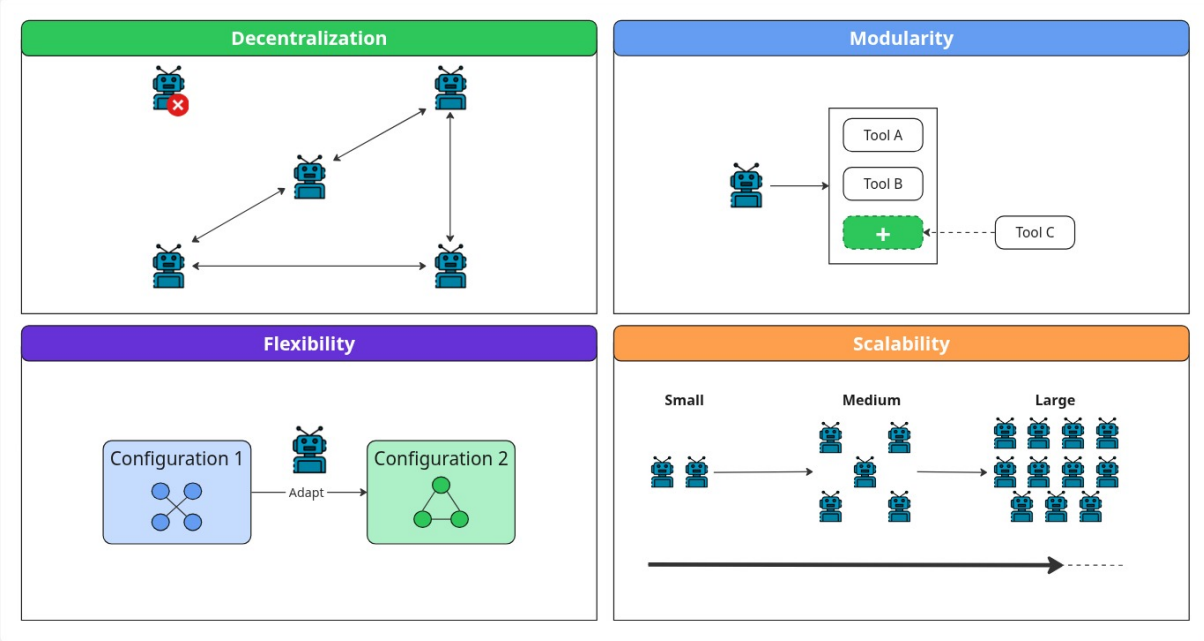

Figure 2.1: Properties of MAS for industrial applications.

MASs applications in manufacturing cover multiple domains [15]. In production scheduling, for example, multi-agent technology creates responsive systems where productive resources (represented by agents) dynamically negotiate task allocation and coordinate actions without centralized rescheduling [16]. This application manifests itself through the encapsulation of entities (machines, tools, products), organization (hierarchical or heterarchical structures), coordination/negotiation (consensus protocols), and learning (improvement through experience). Other applications include flexible manufacturing frameworks, distributed process control, and logistics optimization. Observed benefits include greater responsiveness to disruptions, robustness, ease of integration with legacy systems, and the ability to implement advanced personalization strategies [15], [16].

However, analyses identify emerging challenges, notably the integration of multiple machine learning techniques into multi-agent architectures for collaborative diagnosis [15]. Although MAS have proven successful in domains such as scheduling and control, applications that systematically integrate multiple machine learning techniques distributed across agents, with explicit mechanisms for collaboration and diagnosis fusion, remain limited. As discussed in Section 2.2, individual methods have limitations that could be

addressed through collaborative architectures where techniques complement each other via intelligent coordination.

The lack of studies integrating MAS with multiple machine learning techniques for collaborative diagnosis constitutes a research opportunity that motivates this dissertation. The next section examines existing efforts towards this integration, analyzing both centralized multi-technique approaches and multi-agent applications, and identifying specific gaps that remain.

2.4 Towards Integration of MAS and ML for Diagnosis

This section explore work that approaches the integration of multiple diagnostic techniques in industrial contexts, allowing us to identify specific gaps that this dissertation addresses. The analysis is structured around two complementary perspectives: first, approaches that integrate multiple machine learning techniques in centralized architectures; second, applications of MAS to diagnosis with limited exploration of multiple collaborative techniques.

The integration of multiple machine learning techniques for diagnosis has been explored through ensemble methods and hybrid approaches. Advanced methodologies combine optimized signal processing transforms (bicoherence, spectral kurtosis, cyclic coherence) with deep blending ensemble learning, integrating multiple deep learning models [17]. Ensemble training on composed datasets demonstrates improvements over isolated approaches, with efficient transfer between different components and failure modes. Hybrid approaches integrating supervised and unsupervised paradigms have been applied to discrete manufacturing systems, combining discrete event systems with data-driven techniques [18]. These methods integrate signals from programmable controllers, Petri models, and machine learning algorithms. Rule-based classifiers (Random Forest, Extreme Random Forest) achieve high performance metrics (accuracy/recall: 0.96); unsupervised

techniques via principal component analysis achieve detection rates of 98% with false alarms below 3%, using significantly reduced training sets.

These approaches, although effective, have three critical architectural limitations. First, they operate in a centralized manner, creating single points of failure and failing to take advantage of distributed processing. Second, the fusion of techniques is static, determined during development without dynamic adaptation to variable operating regimes or types of emerging anomalies. Third, they do not incorporate intelligent coordination mechanisms where techniques actively collaborate in solving ambiguous or conflicting diagnoses, or adapt their participation in the final decision based on relative confidence in different operating contexts.

The application of MAS to industrial diagnostics has been explored primarily in the contexts of predictive maintenance and Industry 4.0. The integration of AI techniques, including machine learning and deep learning, with multi-agent architectures aims to improve diagnostic accuracy through models such as SVMs, RFs, CNNs, and RNNs for complex industrial data analysis [19]. AI algorithms accelerate anomaly identification by exploiting operational data, sensors, and historical trends, minimizing downtime. AI-driven predictive maintenance approaches emphasize prognosis, predicting remaining useful life and enabling strategic maintenance scheduling. However, these applications face practical challenges of data quality, heterogeneous system integration, technical skill gaps, and explainability requirements [19].

A more specific multi-agent methodology combines supervised and semi-supervised methods to identify both known and novel anomalies in industrial processes [20]. The approach aims to build a decision support tool for operators to identify and manage abnormal situations in real time. Recognizing that supervised methods, although accurate, do not diagnose new faults, and that semi-supervised methods detect new anomalies but do not reveal root causes, the methodology combines both in a parallel-serial architecture exploiting complementarities. Supervised methods classify known faults with high accuracy; semi-supervised methods detect anomalous patterns not previously observed. The approach provides interpretable explanations and has been validated through two case

studies: the Tennessee Eastman benchmark process and real data from a heat recovery system in a thermomechanical pulp mill [20].

Despite these advances, three significant gaps remain. First, documented applications focus predominantly on predictive equipment maintenance and fault diagnosis in machines or continuous processes, not specifically addressing geometric defects in products along multi-stage assembly lines. Second, although they integrate MAS with machine learning techniques, they do not systematically exploit collaboration between multiple heterogeneous techniques distributed across specialized agents. In work combining supervised and semi-supervised methods [20], integration occurs within a defined structure without mechanisms for dynamic coordination or negotiation between agents employing different techniques to reach consensus on complex diagnoses. Third, there is no evidence of explicit adaptive fusion mechanisms where multiple agents, each using a distinct technique adapted to their context, collaborate dynamically to produce consensus diagnoses that are more robust than individual predictions.

The analysis of related works reveals gaps in the literature consulted, as illustrated in Figure 2.2. Although there are approaches demonstrating the effectiveness of combining multiple machine learning techniques in centralized architectures [17], [18], and works applying MAS to industrial diagnostics [19], [20], the effective integration of MAS with multiple machine learning techniques collaborating autonomously for the diagnosis of defects in manufactured products, with explicit protocols for coordination, negotiation, and decision fusion, remains insufficiently explored. The identified works do not specifically address the diagnosis of geometric defects (gaps, flushes) in products along multi-stage lines, nor do they systematically explore fusion mechanisms where multiple agents employing different techniques actively collaborate.

This dissertation proposes to address these gaps by developing a Multi-Agent System for collaborative diagnosis of geometric defects in automotive components in body shop lines. The approach allows multiple agents, each employing different machine learning techniques, to generate independent diagnoses that are subsequently integrated through coordination and adaptive fusion mechanisms to produce consolidated assessments that

are more reliable than individual diagnoses. The system leverages the modularity, scalability, and robustness characteristics of multi-agent architectures identified in Section 2.3, while overcoming limitations of individual machine learning methods discussed in Section 2.2, through distributed collaboration between heterogeneous techniques that complement each other in detecting defects in different operational contexts.

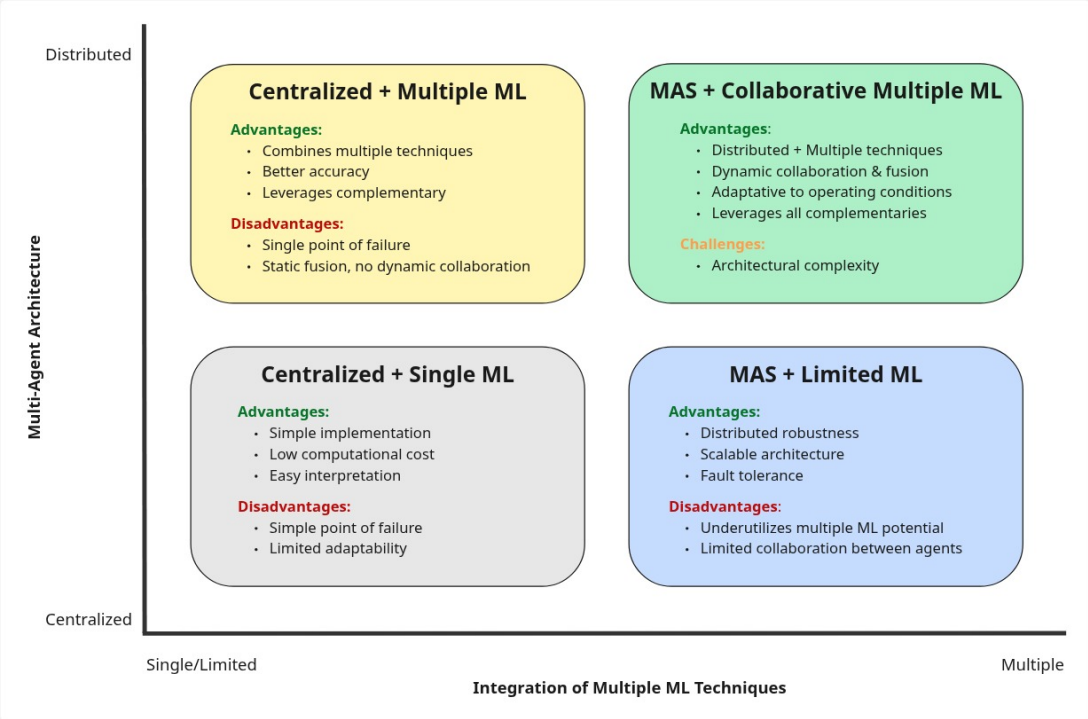

Figure 2.2: Research landscape for diagnostic approaches in industrial applications.

Chapter 3

Case Study

This chapter presents the case study that serves as the basis for the development of this work, describing the context of the automotive assembly line and the characteristics of the data used for the development of the proposed Multi-Agent System.

3.1 Automotive Assembly Line Context

The automotive industry is characterized by highly complex manufacturing processes and rigorous quality requirements, where the dimensional accuracy of assembled parts is one of the main factors determining the quality of the final product. During the vehicle assembly process, it is essential that the dimensions measured at critical points on the body are within pre-established limits, ensuring that the final product meets the design specifications and quality requirements expected by customers. Any deviation observed in the dimensional values may indicate potential fitting problems in subsequent assembly stages, compromising not only the vehicle's aesthetics, but also its functionality, safety, and durability.

Late detection of dimensional defects can result in costly rework at advanced stages of production, compromised customer-perceived quality, problems with sealing and noise in moving components, difficulties in subsequent surface finishing processes, and possible rejection of complete units, resulting in significant losses. For this reason, it is necessary

to perform systematic surface alignment inspections at various stages throughout the production line, allowing for early detection of non-conformities and preventing their propagation to later stages of the process.

The case study focuses on the body shop area of an automotive assembly line, where the basic structure of the vehicle body is assembled. At this stage of the production process, known as body-in-white, only the metal structure of the body is assembled, without the application of coatings or installation of internal components. The dimensional inspection process considered in this work involves two sequential automated measurement stations, positioned at strategic points on the assembly line. Figure 3.1 illustrates the sequential positioning of the inspection stations along the production flow.

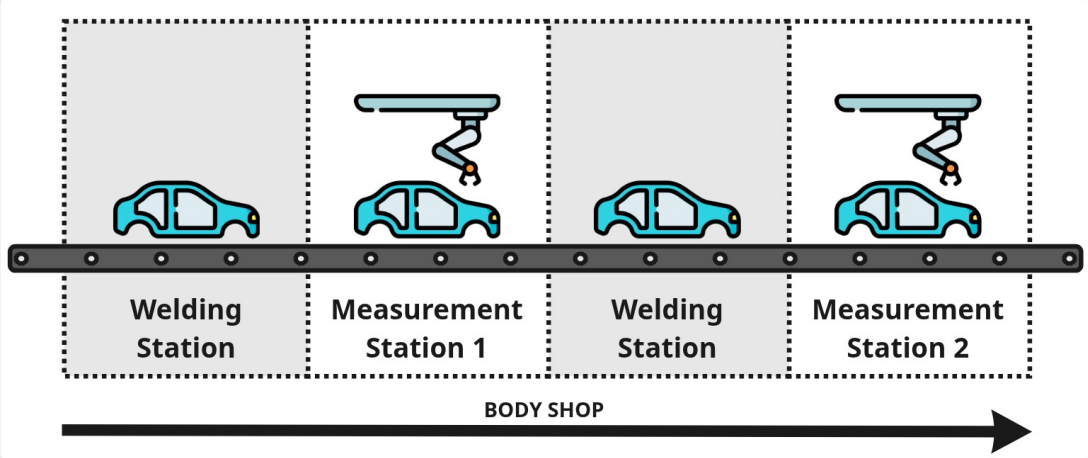

Figure 3.1: Sequence of inspection stations analyzed in the body shop area.

3.1.1 Automated Measurement System

The inspection system consists of robotic cells equipped with advanced laser triangulation optical sensors. These sensors perform high-precision automated measurements, capturing geometric parameters between adjacent parts. Two main parameters are measured by the inspection system: gap, which represents the horizontal distance between two adjacent surfaces of assembled components, and flush, which measures the height difference between two adjacent surfaces, indicating whether there is a misalignment between components

that should be aligned in the vertical plane. Gap values outside specifications may indicate problems such as misalignment in assembly, component deformation, or errors in fastener positioning, while flush deviations may result from improper fastening, deformation during the assembly process, or dimensional inadequacy of individual components. Figure 3.2 schematically illustrates the concepts of gap and flush.

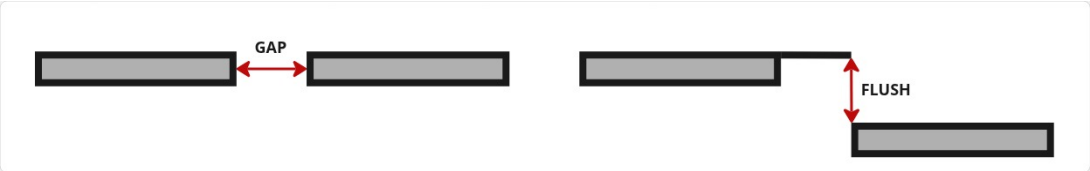

Figure 3.2: Schematic representation of the gap and flush parameters.

3.1.2 Inspection Stations

The first station takes measurements on vehicles that already have key components assembled on the structure, such as doors and side panels. This station takes measurements at 42 different points, checking gap and flush parameters between different body elements. The second station examines vehicles at a more advanced stage of assembly, with the bodywork practically complete in terms of the body-in-white structure. At this station, measurements are taken at 20 specific points, focusing on critical areas and final dimensional alignment checks.

At each station, the vehicle is positioned precisely in the measuring cell and the robots execute pre-programmed trajectories, positioning the laser sensors at the defined measuring points. The measured values are automatically processed and compared with the specified limits, with all records being stored for later analysis.

3.1.3 Challenges and Opportunities

Despite the sophisticated automated measurement system, there are significant challenges in effectively using the collected data. Considering 42 points at the first station and 20 points at the second, dozens of measurements are generated for each vehicle produced,

and with continuous production, thousands of records are generated daily. The current system operates primarily in a reactive manner, identifying problems after they occur, without the predictive capacity to anticipate defects. The analyses performed typically focus on specific cases or isolated problems, failing to cover the process holistically, and there is no automated analysis of the relationships between measurements from different stations or between different points on the same vehicle, making it difficult to understand how defects propagate.

The complexity and non-linearity of the relationships between the measurement points distributed across the two stations make it difficult to understand them through manual analysis, and a large volume of historical data remains unexplored, representing potential knowledge that has not been extracted. These challenges create significant opportunities for improvement through the implementation of advanced automated analysis techniques. The application of multiple machine learning techniques coordinated through a multi-agent architecture can enable the automatic processing of all data collected from measurement points, extracting insights that would be impractical through manual analysis.

An intelligent system can develop the ability to anticipate problems at the second station based on patterns observed at the first station, identify nonlinear relationships and interdependencies between multiple measurement points, and perform automatic classification of defect types. The use of multiple machine learning algorithms operating in parallel, combining their predictions for more reliable results, offers robustness through diversity, where the failure or limitation of an individual technique does not compromise the overall diagnosis. This approach aligns with the principles of Zero Defect Manufacturing, which seeks not only to detect defects, but to prevent their occurrence and propagation, contributing to cost reduction by decreasing rework through prevention and early detection of defects.

3.2 Data Sources and Description

This section describes the structure and characteristics of the data used in this work, which comes from the automated measurement system of the two inspection stations. The data is structured in JSON (JavaScript Object Notation) format, chosen for its flexibility, readability, and broad compatibility with analysis tools. Each record represents the complete measurements taken on a specific vehicle at one of the inspection stations.

3.2.1 Data Structure

The main fields for each register include:

- **Vehicle ID:** Unique vehicle identification number on the production line, enabling complete traceability of each unit and allowing measurements from different stations to be correlated for the same vehicle;
- **TIMESTAMP:** Exact date and time when the measurement was taken, enabling precise temporal ordering and analysis of trends over time;
- **MEASUREMENT_DATA:** Array containing all measurement points taken on the vehicle, with multiple points distributed across the inspection stations.

Each element of the MEASUREMENT_DATA array represents a specific measurement point and has the following structure:

- **Name:** Unique alphanumeric identifier for the measurement point, containing coded information about the vehicle region, components involved, and side;
- **Measured Values:** Object containing the values actually measured by the sensors:
  - *Gap:* Measured gap value in millimeters (may be null if not applicable);
  - *Flush:* Measured flush value in millimeters (may be null if not applicable).

Each measurement point has multiple limits that define quality zones:

- **US (Upper Specification) and LS (Lower Specification):** Upper and lower specification limits, where values measured within this range are considered to be in the ideal quality zone;
- **UT (Upper Tolerance) and LT (Lower Tolerance):** Upper and lower tolerance limits, which define intermediate alert zones between specification and rejection limits, indicating measurements that require attention but are not yet critical;
- **UR (Upper Rejection) and LR (Lower Rejection):** Upper and lower rejection limits, where values measured above UR or below LR are considered critical non-conformities.

Listing 3.1 presents a simplified representation of the data structure, with elided fields for brevity.

Listing 3.1: Generic structure of measurement data

```
{
   "VEHICLE_ID": "number",
   "TIMESTAMP": "datetime",
   "MEASUREMENT_DATA": [
      {
         "Name": "string",
         "D": {"Gap": "float", "Flush": "float"},
         "US": {"Gap": "float", "Flush": "float"},
         "LS": {"Gap": "float", "Flush": "float"},
         "UR": {"Gap": "float", "Flush": "float"},
         "LR": {"Gap": "float", "Flush": "float"},
         "UT": {"Gap": "float", "Flush": "float"},
         "LT": {"Gap": "float", "Flush": "float"}
      }
   ]
}
```

3.2.2 Interpretation of Limits and Classification Zones

The system uses multiple levels of thresholds to classify measurements into different quality zones, as illustrated in Figure 3.3. This multi-level structure allows for granular quality management, differentiating between minor deviations that can be monitored and critical deviations that require immediate action.

Figure 3.3: Measurement classification zones based on specified limits.

The ideal specification zone, delimited between LS and US, contains values considered compliant and representing the desired quality. The alert zone, located between US and UT (upper side) or between LT and LS (lower side), contains values that are outside the ideal specification but have not yet reached critical levels, generating alerts for monitoring. The rejection zone, corresponding to values above UR or below LR, indicates serious non-compliance and may require immediate intervention or rework.

3.2.3 Relevant Features for the Multi-Agent System

The collection of data presents features that support and justify the multi-agent approach proposed in this work. The availability of thousands of inspected vehicles results in an adequate number of historical examples, while specification limits provide natural references for conformity assessment that can be used by different diagnostic algorithms. Each vehicle has several measurements between stations, providing rich context for pattern identification through multiple analytical perspectives.

The sequential structure of the process, with two distinct stations, allows for the analysis of defect propagation along the assembly line, where patterns observed at the first station can be correlated with results at the second station. The JSN field enables this correlation between measurements from the two stations for the same vehicle, which is essential for understanding how dimensional characteristics evolve through the production

process. Actual production data captures natural variations in the process, including normal manufacturing fluctuations and occasional anomalous events, providing a realistic basis for model development and validation.

The dimensional complexity of the problem, with a total of 62 measurement points distributed across two stations, benefits from different algorithmic perspectives working in parallel, each potentially identifying distinct aspects of the quality status. The heterogeneity of the data, where some points measure only gap, others only flush, and others both, creates a scenario where different agents can specialize in different types of dimensional patterns. The need for reliability in an industrial environment justifies the combination of multiple models through coordination and fusion mechanisms, where the failure or limitation of an individual algorithm does not compromise the overall system.

3.2.4 Potential Diagnostic Algorithms

The characteristics of the available data and the requirements of the industrial context suggest several diagnostic approaches that could be coordinated through a multi-agent architecture. A first possible approach involves using pre-established limits set by engineers to categorize measurements into different quality zones. This type of deterministic algorithm could classify each measurement as being within the ideal specification, within an acceptable tolerance, or in a rejection zone due to violation of critical limits, providing an assessment based directly on known process specifications.

In addition, statistical anomaly detection techniques could identify atypical behavior in measurements even when they have not yet violated specification limits. This approach would be particularly valuable for early identification of degrading trends before they become critical nonconformities, considering the recent history of measurements to detect unusual deviations from normal process behavior. Robust methods using medians and absolute deviations could be more appropriate than techniques based on means and standard deviations, especially in industrial environments where occasional outliers are expected.

A third possible approach would be to analyze time trends to predict future violations of tolerance limits. Using recent measurement history to adjust trend models, it would be possible to project the future trajectory of measured values and estimate how many vehicles can be produced before a non-conformity occurs. This predictive capability would transform the system from reactive to proactive, enabling preventive interventions that avoid the production of defective units.

The complementarity of these approaches is evident when considering their different perspectives on the diagnostic problem. While threshold-based classification provides deterministic reference aligned with engineering specifications, anomaly detection captures unusual behaviors that may precede systematic failures, and prediction allows for the anticipation of problems before they manifest. Coordinating these approaches through a multi-agent architecture would allow each technique to contribute its specialized perspective, while fusion mechanisms would synthesize these perspectives into consolidated diagnoses that are potentially more robust than any individual technique could produce in isolation.

3.3 Summary

The case study presented offers a real and industrially relevant scenario for the development and validation of a Multi-Agent System for diagnosing geometric defects in automotive manufacturing processes. The body shop area represents a challenging environment where dimensional quality is critical to the structural and aesthetic integrity of the final product, large volumes of data are continuously generated through automated measurement systems, and complex relationships between the measurement points distributed across two sequential stations hinder traditional manual analysis.

The availability of structured real data provides a solid basis for implementing advanced automated diagnostic approaches. The sequential nature of the process, where vehicles are inspected at multiple stages identified by the same JSN, allows for defect

propagation analysis and inter-station correlations that are fundamental to a holistic understanding of dimensional quality.

The challenges identified in the current use of data, including massive volumes of information, fragmented analysis, and the lack of systematic correlation between measurements, provide clear motivation for the solution proposed in this work. The Multi-Agent System developed aims to transform raw data into actionable knowledge by coordinating multiple complementary analytical techniques, each contributing its specialized perspective to consolidated diagnoses that are more reliable than individual approaches.

The implementation of classification, anomaly detection, and prediction algorithms, coordinated through collaboration and adaptive fusion mechanisms, offers potential for early defect detection, anticipation of future non-conformities, and cost reduction by preventing rework and production interruptions. The alignment of this case study with the principles of Zero Defect Manufacturing reinforces its industrial relevance and potential impact on continuous quality improvement in automotive production processes.

Chapter 4

Multi-Agent System Architecture

This chapter presents the architecture of the multi-agent system developed for diagnosing defects in automotive assembly lines. The proposed architecture emphasizes modularity, extensibility, and distributed collaboration, allowing the system to dynamically adapt to operating conditions and integrate multiple analytical techniques in a cohesive manner. The chapter is structured into three main sections: Section 4.1 provides an overview of the hierarchical organization of the system, Section 4.2 details the components and operational mechanisms, and Section 4.3 describes the complete flow of data and operations that integrates all elements into a unified perspective.

4.1 Overview of the Architecture

The proposed multi-agent system for diagnosing defects in automotive assembly lines is structured in three distinct hierarchical layers, each performing specific functions in quality analysis and evaluation. Figure 4.1 presents a simplified view of this architecture, illustrating the relationships between the different types of agents and the organization of the system.

The base layer of the architecture consists of Point Agents, which are the fundamental units of analysis. Each Point Agent is responsible for diagnosing a specific measurement

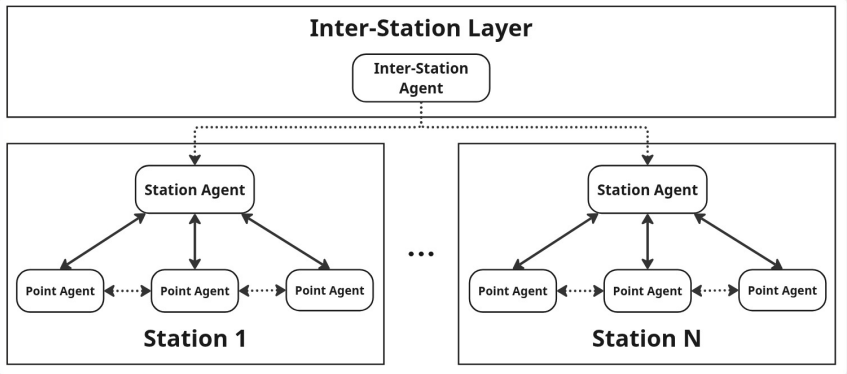

Figure 4.1: Overview of the multi-agent system architecture, showing the hierarchical organization in three layers: Inter-Station Layer, Station Agents, and Point Agents.

point, applying multiple algorithms to the collected data and producing detailed assessments. These agents operate independently but retain the ability to collaborate with other agents that are monitoring correlated points.

Above the Point Agents are the Station Agents, one per station, who act as local coordinators within each quality control station. Each Station Agent manages the group of Point Agents associated with the measurement points at their station, distributing data, coordinating analyses, and aggregating diagnostic results. This intermediate layer provides an initial consolidation of information, transforming multiple individual diagnoses into station-level assessments.

At the top of the hierarchy operates the Inter-Station Agent, which provides a cross-sectional view of multiple measurement stations. This agent identifies correlations between points at different stations along the assembly line, allowing the propagation of defects to be tracked through the various stages of the production process. The inter-station layer complements local analyses with a systemic view that would be impossible to obtain through isolated evaluations of each station.

Communication in the system flows both vertically and horizontally. Vertically, data and commands flow down from Station Agents to Point Agents, while diagnostic reports flow up from Point Agents to Station Agents. Horizontally, the Inter-Station Agent communicates with multiple Station Agents to collect information and distribute inter-station correlation clusters. Additionally, Point Agents belonging to the same correlation cluster

communicate with each other to coordinate the aggregation of diagnostics.

Hierarchical architecture provides multiple operational advantages. The distribution of responsibilities allows scalability, as new stations or measurement points can be incorporated by adding appropriate agents without restructuring the existing system. The specialization of each type of agent in specific tasks reduces individual complexity and facilitates maintenance. The layered organization allows failures to be contained, preventing local problems from compromising the overall system.

The following subsections detail each aspect of this architecture, describing the specific responsibilities of each type of agent, the communication protocols implemented, the correlation and clustering algorithms, the diagnostic mechanisms, the reporting structure, and finally the complete operational flow that integrates all these components.

4.2 Detailed Components and Operation

This section describes, in logical and functional terms, the components that make up the multiagent architecture and how they operate and interact. The objective is to establish boundaries of responsibility, immediate dependencies, and informational artifacts produced and consumed by each component, in order to support a coherent reading of the subsequent parts of the chapter. The emphasis is on defining roles, interfaces, and products, abstracting from implementation choices and methodological details that will be covered later.

The internal organization is as follows: §4.2.1 describes the types of agents and their direct responsibilities, as well as the artifacts that each one emits and consumes; §4.2.2 formalizes communication interfaces, specifying message structures and interaction semantics; §4.2.3 presents the correlation and cluster formation/maintenance strategy; §4.2.4 details the diagnostic algorithms executed at the measurement point level; and §4.2.5 defines the structure of system reports and their role in decision support.

4.2.1 Agents: Types and Responsibilities

The proposed multi-agent system consists of three distinct types of agents, each designed to perform specific functions within the fault diagnosis architecture. The choice of a distributed architecture based on multiple agents, as opposed to a centralized approach, is based on several operational and architectural benefits. This distribution allows for greater scalability, since new measurement points can be incorporated by creating new Point Agents without the need to restructure the system. In addition, the distributed architecture provides greater robustness, as the failure of an individual agent does not compromise the overall functioning of the system. The specialization of each type of agent in specific tasks results in greater modularity and facilitates system maintenance and evolution. This subsection describes the fundamental responsibilities of each type of agent and their operational behaviors through state diagrams.

Station Agent

The Station Agent acts as the primary interface between the measurement system and the agent-based diagnostic framework. It operates at the station level, managing the complete measurement data lifecycle for a specific quality control station on the assembly line.

The Station Agent is responsible for receiving all measurement points from units passing through the station and validating and pre-processing this data. It maintains a historical database of measurements for analysis purposes and creates and manages Point Agents, with each agent dedicated to analyzing a specific measurement point. Using historical data, the agent calculates intra-station correlations between measurement points and organizes correlated agents into groups called clusters, a process that is performed periodically within specific measurement windows. This correlation analysis allows the identification of patterns where multiple measurement points exhibit related anomalous behavior, indicating potential systemic problems that require attention. Additionally, the Station Agent can receive inter-station correlation clusters from the Inter-Station Agent,

subsequently distributing both the measurements and the correlation clusters to the respective Point Agents. Finally, it aggregates and summarizes the Cluster Reports received to generate comprehensive Station Reports.

Figure 4.2 shows the Station Agent state diagram, representing its workflow through different processing phases.

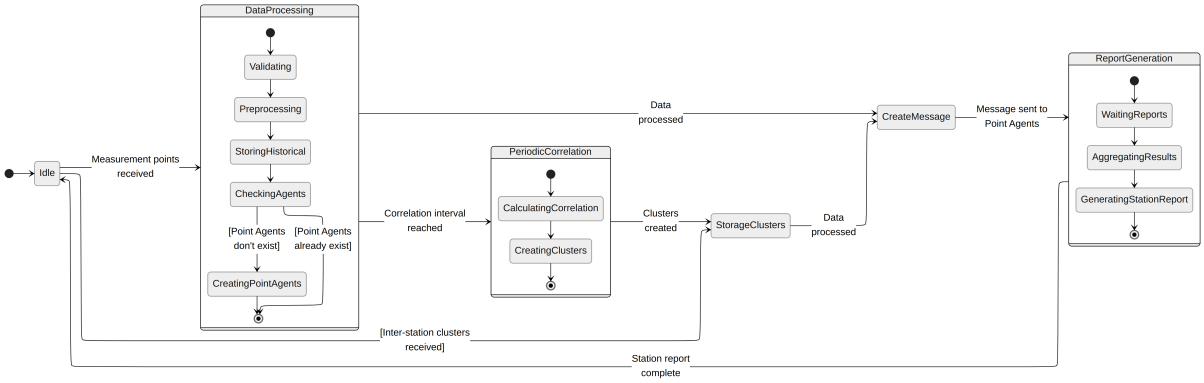

Figure 4.2: State diagram of the Station Agent showing data processing flows, correlation analysis, and report generation.

Point Agent

The Point Agent represents the fundamental analytical unit of the system, responsible for the diagnosis of individual measurement points. Each Point Agent operates independently, while maintaining collaborative capabilities with correlated agents.

The core responsibilities of the Point Agent include receiving and buffering data from individual measurement points, applying multiple diagnostic algorithms to detect anomalies and defects. It generates Point Reports containing detailed diagnostic results and communicates with agents belonging to its correlation cluster. It participates in leader election processes within clusters, assuming different responsibilities depending on the outcome of this election. When elected as cluster leader, the agent aggregates the Point Reports of cluster members to generate Cluster Reports, which it then forwards to the Station Agent. When operating as a cluster member, it transmits its Point Report to the

cluster leader agent. This hierarchical structure within clusters allows for efficient communication and reduces message overload in the system, since only the leader communicates directly with the Station Agent.

The operational behavior of the Point Agent is represented in Figure 4.3, which shows the transitions between data analysis states and communication roles inside the cluster.

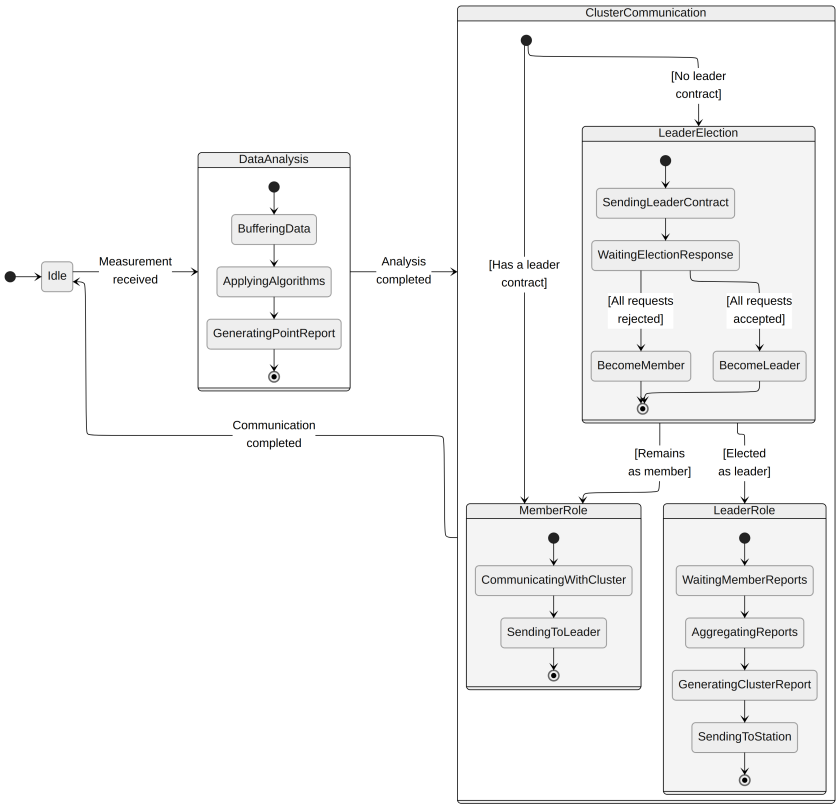

Figure 4.3: State diagram of the Point Agent illustrating the data analysis flow and cluster communication mechanisms, including leader election and role-specific behaviors.

Inter-Station Agent

The Inter-Station Agent operates at a systemic level, providing a global perspective on quality patterns across multiple measurement stations. This agent enables the identification of correlations between stations that may indicate systemic manufacturing problems, allowing the identification of defects which have their origin in earlier stages of the production process.

The Inter-Station Agent is responsible for collecting data from measurement points at all stations on the assembly line and calculating correlations between measurement points at different stations. Based on the correlations identified, it creates inter-station clusters, a process that is performed periodically within specific measurement windows. Finally, it distributes the information from the inter-station clusters to the respective Station Agents. This cross-sectional analysis capability is particularly relevant in automotive contexts, where defects observed in final assembly may be related to problems that occurred during the body shop phase, requiring a holistic view of the production process.

Figure 4.4 shows the state diagram of the Inter-Station Agent, highlighting its cyclic correlation analysis process.

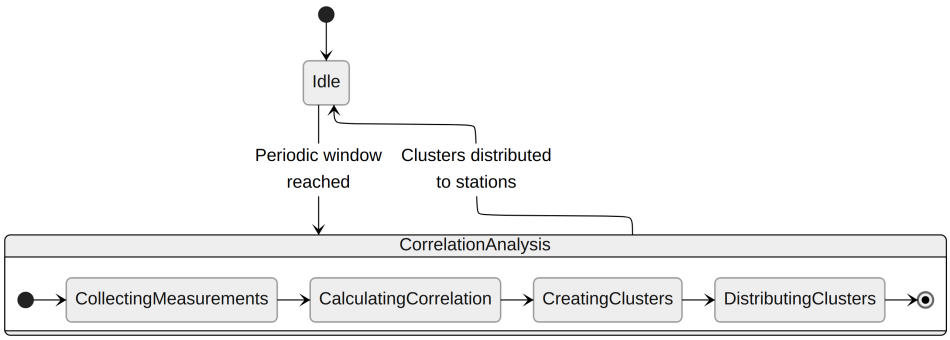

Figure 4.4: State diagram of the Inter-Station Agent, highlighting its cyclic correlation analysis process.

These three types of agents work synergistically to create a comprehensive and distributed fault diagnosis system. Station Agents provide localized analysis and coordination, Point Agents deliver detailed diagnostic assessments with collaborative capabilities, and the Inter-Station Agent ensures systemic consistency through correlation analysis between stations. This hierarchical and collaborative architecture enables both granular and holistic quality monitoring throughout the entire automotive assembly process. The operationalization of this collaboration between agents depends, however, on well-defined communication mechanisms, a topic that will be addressed in the next subsection.

4.2.2 Communication Protocols

The success of a multi-agent system depends on communication mechanisms that allow coordination and collaboration between different agents. This subsection describes the communication protocols implemented in the system, detailing the interactions between the three types of agents and the specific mechanisms for leadership election inside the clusters.

General System Communication

The system implements a well-defined set of communication rules that govern interactions between agents. Figure 4.5 illustrates the general communication flow and the main points of interaction in the system.

The communication protocol follows a hierarchical structure where the Station Agent coordinates local operations within its station. When the Station Agent receives data from the measurement station, it processes and stores this information in the historical database. If new measurement points are detected, the agent dynamically creates the respective Point Agents. Periodically, after a specific number of measurements (correlation window), the Station Agent calculates the intra-station correlations and creates clusters of correlated agents.

Simultaneously, the Inter-Station Agent collects measurements from all stations in the system and, also periodically, calculates correlations between points from different stations, creating inter-station clusters. These clusters are then distributed to the respective Station Agents, providing a cross-sectional view of the production process.

Once the measurements have been processed and the clusters defined (both intra- and inter-station), the Station Agent distributes the corresponding measurement data and information about the clusters to each Point Agent. This distribution allows each Point Agent to be aware not only of its own data, but also of its relationship with other correlated measurement points.

To complete the flow specification, the cluster leader election protocol is described

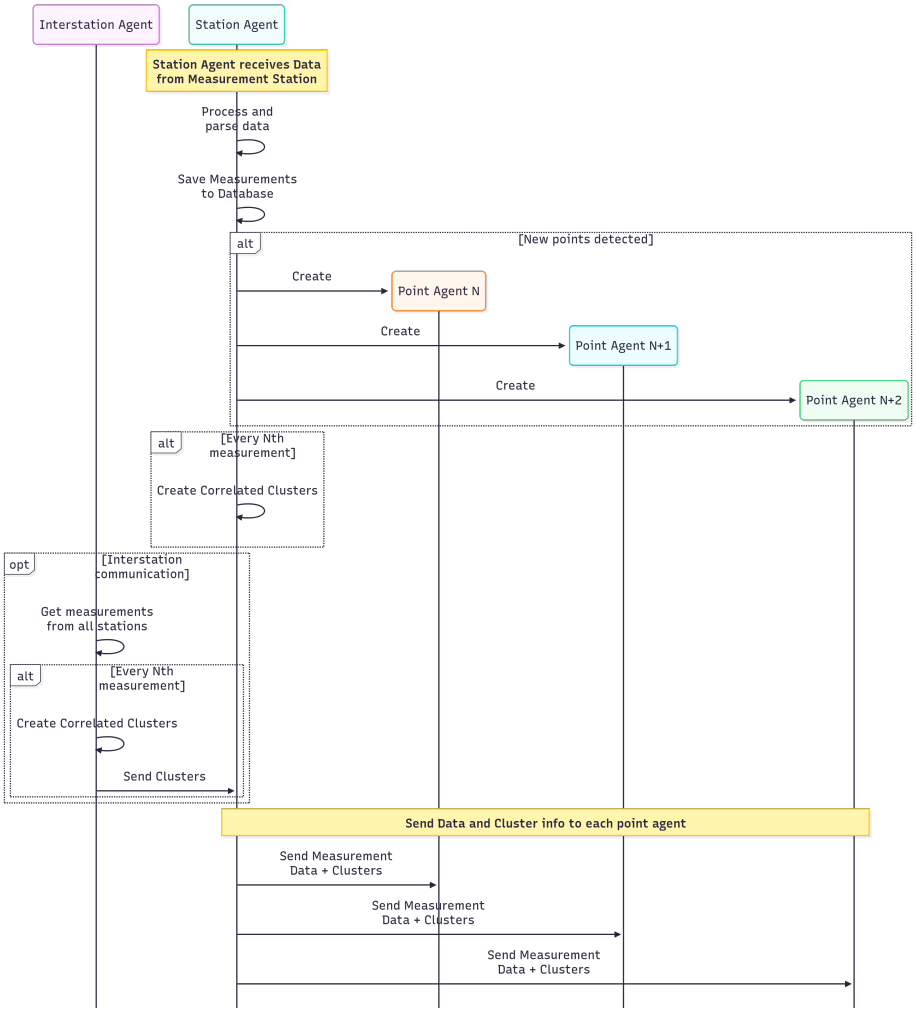

Figure 4.5: Sequence diagram of general system communication, showing interactions between the three types of agents.

below, which deterministically decides the aggregator agent and minimizes communication redundancies.

Cluster Leader Election Protocol

A critical aspect of the system is the leader election mechanism inside the clusters, which determines which Point Agent will be responsible for aggregating the members' diagnostics and communicate with the Station Agent. This protocol ensures efficient communication and avoids system overload. Figure 4.6 shows the detailed sequence diagram of this protocol.

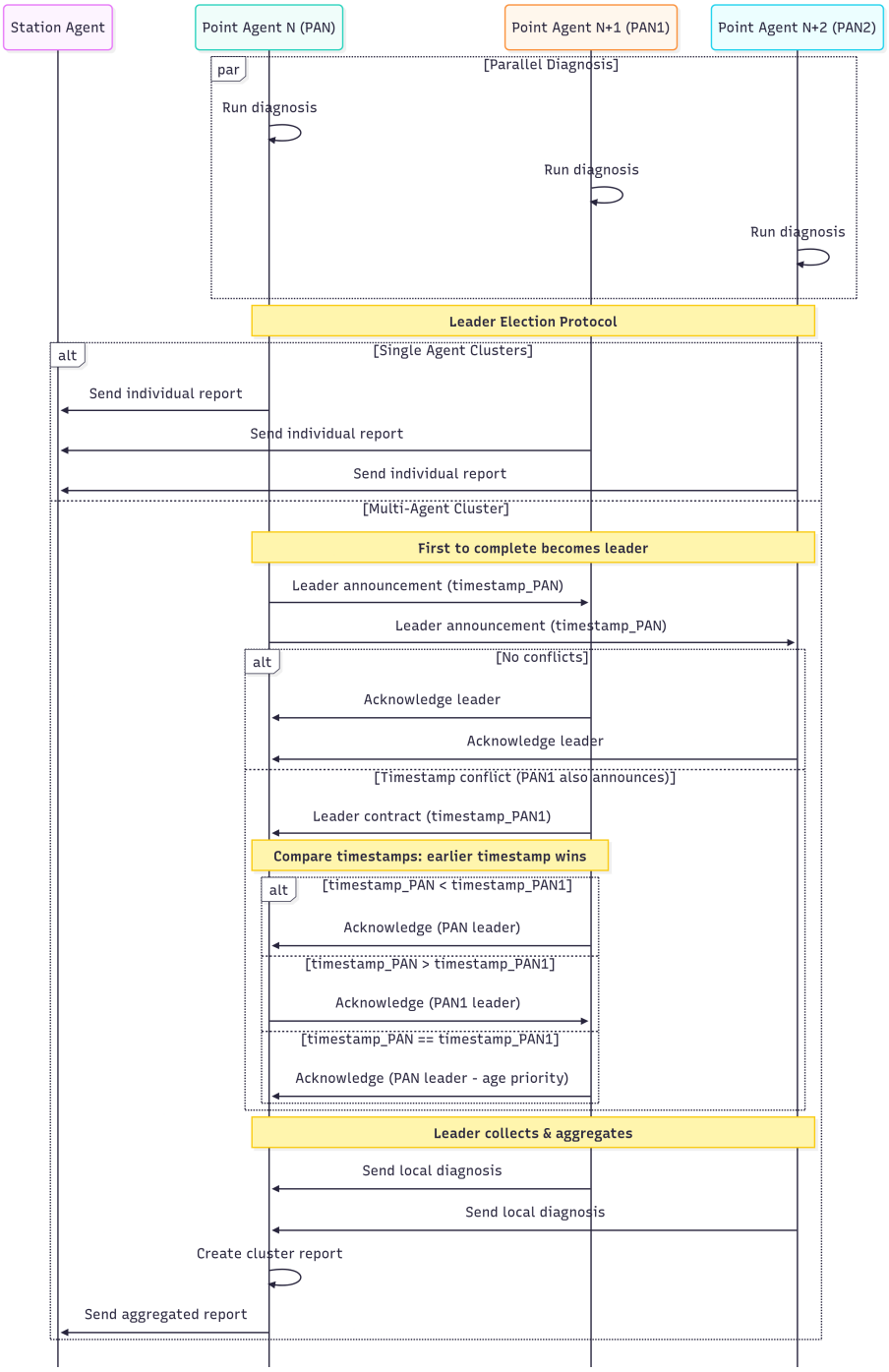

Figure 4.6: Sequence diagram of the leader election protocol and communication inside the cluster.

After receiving the data and cluster information from the Station Agent, all Point Agents run their diagnostic algorithms in parallel, generating the respective Point Reports. The subsequent communication protocol then distinguishes between two main scenarios based on the composition of the cluster.

In the case of clusters composed of a single agent, when the Point Agent completes its diagnosis and identifies that it is the only member of its cluster, it proceeds directly to transform its Point Report into a Cluster Report (which in this case contains only one individual report) and communicates it directly to the Station Agent, without the need for aggregation with other reports.

For multi-agent clusters, the protocol implements a leader election mechanism based on timestamps. The first agent to complete its diagnosis becomes a candidate for leader and announces its intention to the other cluster members through a leadership contract. This contract contains essential information that allows cluster members to validate and recognize leadership: a unique contract identifier, the identifier of the agent proposing itself as leader, and the contract creation timestamp, which serves as a tiebreaker in conflict situations.

In non-conflict situations, where only one agent announces their candidacy, the remaining members recognize them as leader and subsequently send them their individual reports. The leader then aggregates all reports from the cluster and sends the consolidated report to the Station Agent.

Nonetheless, situations of conflict may happen where multiple agents complete their diagnoses at the same time and announce their candidacy for leadership. In these cases, the protocol uses timestamps to resolve the conflict in a deterministic way. The agent with the oldest timestamp is elected leader. If the timestamps are identical, which can occur in systems with precise time synchronization, the protocol applies a tiebreaker criterion based on the agent's seniority, always ensuring a unique resolution of the conflict.

This distributed election mechanism has several advantages. First, it does not require a central coordinator for the election, distributing responsibility among the agents themselves. Second, it promotes efficiency by electing as leader the agent that completes its

diagnosis first, reducing the total processing time. Third, the protocol is deterministic and ensures that there will always be a single elected leader, avoiding situations of ambiguity or multiple leaders.

The implemented protocol is based on established principles of leader election in distributed systems [21], adapting the use of timestamps as a priority criterion [22]. In contrast to the classic Bully algorithm, which uses fixed process identifiers to determine priority, the adopted approach uses diagnostic completion timestamps as a dynamic criterion. This adaptation favors temporal efficiency by naturally electing the fastest agent as leader, which is particularly relevant in industrial contexts where latency minimization is critical. The use of timestamps for ordering events in distributed systems is a well-established principle that ensures consistency even in cases where clocks between agents are not perfectly synchronized.

Efficiency of Hierarchical Communication

The hierarchical communication structure within clusters provides significant benefits in terms of system efficiency and scalability. Considering a cluster composed of N agents, the hierarchical approach requires $N - 1$ messages from members to the leader plus one message from the leader to the Station Agent, totaling N messages. In contrast, in a non-hierarchical architecture where all agents communicate directly with the Station Agent, N direct messages would be required, which at first glance seems equivalent.

However, the advantage of the hierarchical approach becomes evident when considering processing in the Station Agent. In direct communication, the Station Agent would receive N individual reports and would have to identify which ones belong to the same cluster in order to perform the aggregation. In the hierarchical approach, it receives a single report already aggregated by cluster, significantly reducing the processing load. Additionally, this structure allows the aggregation to be distributed among the leaders of different clusters, performing this work in parallel and improving the overall performance of the system.

For systems with multiple clusters, the efficiency improvements are even more pronounced. Considering C clusters with an average of N agents per cluster, the Station Agent receives only C aggregated reports instead of $C \times N$ individual reports, representing a reduction proportional to the average size of the clusters. This reduction in communication and processing volume at the Station Agent directly contributes to the scalability of the system, allowing it to function effectively even with a large number of measurement points.

4.2.3 Correlation and Clustering Strategy

The formation of clusters of correlated agents is a fundamental element of the architecture, allowing agents with related characteristics to collaborate in the diagnostic process. This subsection describes the algorithm used to identify correlations between measurement points and organize agents into cohesive groups.

Fundamentals of Correlation Analysis

The clustering process is based on statistical analysis of correlations between time series of measurements. The algorithm uses historical data stored by the Station Agent (or by the Inter-Station Agent, in the case of correlations between stations) to calculate a correlation matrix that quantifies the degree of linear relationship between each pair of measurement points.

The correlation is calculated using statistical coefficients, which can be obtained using methods such as Pearson, Spearman, or Kendall, depending on the nature of the data. These coefficients typically range from -1 to 1, where values close to 1 indicate a strong positive correlation, values close to -1 indicate a strong negative correlation, and values close to 0 indicate no linear correlation.

For each measurement point, the system considers separately different types of measurements that can be performed at the same physical point. This distinction is important

because it allows specific patterns to be identified for each type of measurement, recognizing that different aspects of quality may exhibit distinct correlation behaviors.

Clustering Algorithm

The clustering algorithm operates in several sequential phases, transforming the correlation matrix into cohesive groups of agents. Figure 4.7 visually illustrates the complete process, from collecting agent data to forming the final clusters. Algorithm 1 presents the detailed pseudocode for the process.

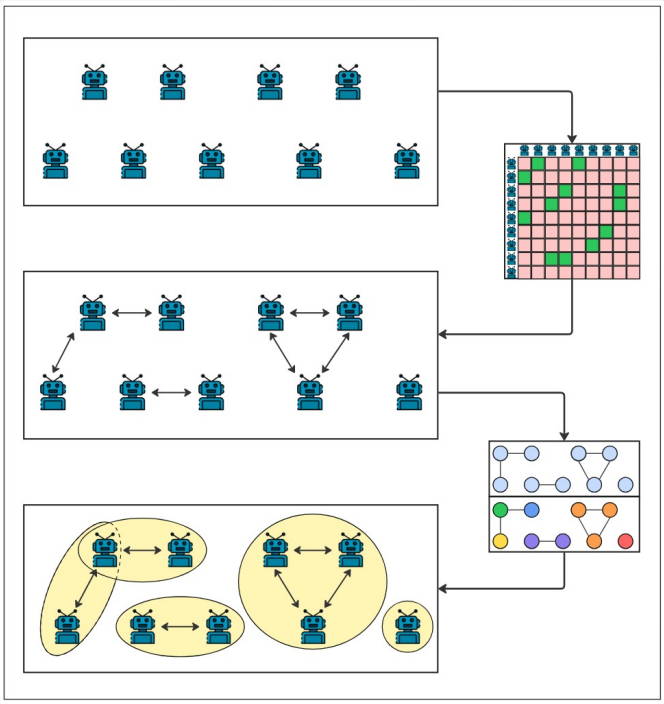

Figure 4.7: Overview of the correlation and clustering algorithm, showing the transformation of agent data into a correlation matrix, adjacency graph, and final clusters.

The first phase (lines 1-2) consists of calculating the correlation matrix R and transforming it into an adjacency matrix G_{adj}. The adjacency matrix is a binary representation where an edge exists between two nodes if and only if the absolute correlation between them exceeds the threshold τ. This threshold is a configurable system parameter, typically set to high values (e.g., 0.9) to ensure that only strong correlations result in cluster formation.

Algorithm 1 Correlation and Clustering Algorithm

Require: A: list of agents, M: historical measurements, τ: correlation threshold
Ensure: C: set of clusters

$R \leftarrow \text{ComputeCorrelationMatrix}(M, A)$
$G_{adj} \leftarrow \text{CreateAdjacencyMatrix}(R, \tau)$
$C \leftarrow \{\}$
$visited \leftarrow \{\}$
for each $a \in A$ **do**
    for each $type \in MeasurementTypes$ **do**
        $key \leftarrow (a.id, type)$
        if $key \notin G_{adj}$ **then**
            $C \leftarrow C \cup \{\text{Cluster}([Member(a.id, type, \{\})])\}$
            $visited[key] \leftarrow \text{true}$
        end if
    end for
end for
$G \leftarrow \text{BuildGraph}(G_{adj})$
$cliques \leftarrow \text{FindCliques}(G)$
for each $clique \in cliques$ **do**
    $grouping \leftarrow \text{GroupByAgent}(clique)$
    if HasAgentWithMultipleTypes($grouping$) **then**
        $separated\_clusters \leftarrow \text{SeparateByType}(clique)$
        for each $subcluster \in separated\_clusters$ **do**
            $members \leftarrow \text{CreateMembers}(subcluster, R)$
            $C \leftarrow C \cup \{\text{Cluster}(members)\}$
            Mark members as visited
        end for
    else
        $members \leftarrow \text{CreateMembers}(clique, R)$
        $C \leftarrow C \cup \{\text{Cluster}(members)\}$
        Mark members as visited
    end if
end for
for each $(key, state) \in visited$ **do**
    if $state = \text{false}$ **then**
        $(a.id, type) \leftarrow key$
        $C \leftarrow C \cup \{\text{Cluster}([Member(a.id, type, \{\})])\}$
    end if
end for
return C

The second phase (lines 3-13) deals with special cases where measurement points do not have sufficient data in the analysis window or do not show significant correlation with any other point. These points are immediately assigned to individual clusters, ensuring that all agents are included in some cluster, even if in isolation.

The third phase (lines 14-15) models the clustering problem as a graph clique detection problem. The graph G is constructed where each node represents a pair (measurement point, measurement type) and the edges represent significant correlations. Maximal cliques in this graph correspond to groups of points that are all correlated with each other, forming natural candidates for clusters. The system uses established algorithms for maximal clique detection, such as the Bron-Kerbosch algorithm [23], which has efficient implementations for sparse graphs typical of industrial applications.

The fourth phase (lines 16-30) processes each identified clique and handles a specific situation that may occur: the same agent may appear in the clique with multiple types of measurement. When this happens, the algorithm separates the clique into distinct sub-clusters by measurement type. This separation is necessary to prevent a single agent from having to participate simultaneously in multiple leader elections in different contexts, which would violate the consistency of the communication protocol. If there is no such overlap, the clique directly forms a cluster.

The final phase (lines 31-36) ensures completeness by processing any points that were not assigned to clusters in the previous phases, creating individual clusters for these cases.

Algorithm Properties and Considerations

The algorithm has several desirable properties for the industrial diagnostics context. First, it ensures that all agents are assigned to exactly one cluster per measurement type, avoiding ambiguities. Second, the use of maximum cliques ensures that within each cluster, all members have a significant correlation with each other, not just with a subset of the members.

The temporal complexity of the algorithm is dominated by the clique detection phase, which in the worst case is exponential in the number of nodes in the graph. However,

in practical applications, the number of strongly correlated points tends to be limited, resulting in sparse graphs where clique detection is efficient.

The parameterization of the correlation threshold τ allows adjusting the trade-off between sensitivity and specificity in cluster formation. High values of τ produce more cohesive clusters but potentially in greater numbers, while lower values may group points with weaker correlations. The appropriate choice of this parameter depends on the specific characteristics of the production process and must be determined experimentally.

4.2.4 Diagnosis Algorithms

The diagnosis capacity of Point Agents constitutes the functional core of the proposed multi-agent system. To ensure flexibility and extensibility, the system implements a modular architecture that allows the integration of multiple analysis and diagnosis algorithms without the need for structural changes in the agent code. This subsection describes the architecture for managing diagnosis algorithms, abstracting from the specific algorithms used, which will be detailed in the experimental chapter.

The system adopts a design pattern based on an abstract class hierarchy that defines common interfaces for different categories of algorithms. This approach allows new algorithms to be added to the system in a systematic and consistent manner, ensuring compatibility with the existing infrastructure of the agents. Figure 4.8 presents the class diagram illustrating the structure of the algorithm architecture. The design distinguishes three fundamental categories of algorithms, each appropriate for different types of diagnostic analysis.

Stateless Algorithms

This category includes algorithms that do not require prior training. They receive a measurement as input and, optionally, additional parameters as needed, apply predefined rules, and return a diagnostic result. The term “stateless” refers to the fact that these algorithms do not maintain internal state between executions, processing each call

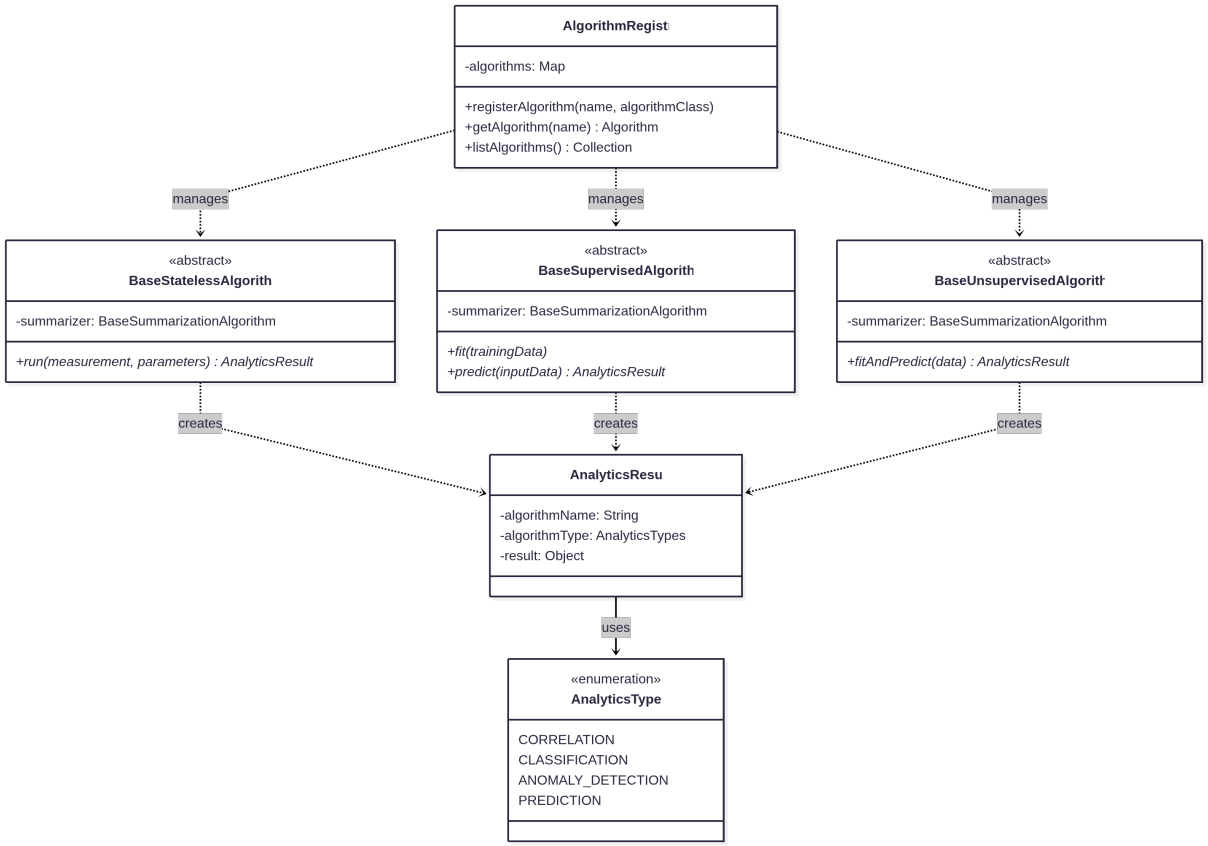

Figure 4.8: Architecture of the diagnostic algorithm system, showing the base abstract classes, the algorithm registry, and the analytical results structure.

independently based only on the parameters received.

Typical examples include tolerance limit checking, where the algorithm compares measured values with acceptable specification ranges. This approach allows for efficient processing and direct interpretation of results.

Supervised Algorithms

Supervised algorithms require a training phase where they learn patterns from labeled historical data, i.e., measurements for which the correct diagnosis is known. After training, the resulting model can be used to classify or predict the status of new measurements. This category is suitable for situations where there is a representative set of historical examples of defects and normal measurements. The process is divided into two distinct phases: the

training phase, performed periodically on accumulated data, and the prediction phase, performed by agents in real time on each new measurement.

Examples of applicable techniques include Support Vector Machines (SVM) for binary defective/non-defective classification, Random Forest for identifying multiple classes of defects, and Artificial Neural Networks for detecting complex patterns.

Unsupervised Algorithms

This category comprises algorithms that identify patterns and anomalies in data without the need for labeled examples. They operate on sets of measurements, discovering intrinsic structures in the data and identifying observations that deviate significantly from normal patterns. They are particularly valuable in scenarios where manual defect labeling is impractical or where new types of defects may emerge. The execution combines training and prediction in a single process, analyzing a set of recent measurements to simultaneously identify normal behavior and anomalies.

Representative techniques include Isolation Forest for anomaly detection, DBSCAN for identifying natural clusters and outliers, and Autoencoders for learning compact representations and detecting deviations.

Algorithm Registration and Management

The system implements a centralized registration mechanism that maintains a catalog of all available algorithms. This registry associates textual identifiers with concrete implementations of algorithms, allowing agents to dynamically instantiate the necessary algorithms through nominal references. The Station Agent is responsible for registering the algorithms available in the system during its initialization. Fundamental operations include adding new algorithms, retrieving implementations by name, and listing all available algorithms.

When a Point Agent is created, it receives a configuration that specifies which algorithms it should execute. The agent queries the registry to obtain the corresponding implementations and maintains instances of these algorithms for subsequent execution.

This configuration-based approach allows the diagnostic behavior of different agents to be customized according to the specific needs of each measurement point, while maintaining the homogeneity of the agent architecture.

Analytical Results Structure

All algorithms, regardless of category, produce results through a standardized structure that encapsulates diagnostic information. This structure contains three essential components: the identification of the algorithm that produced the result, allowing traceability; the classification of the type of analysis performed, facilitating further processing; and the result itself, whose specific format depends on the algorithm but is encapsulated consistently.

The standardization of results allows Point Agents to aggregate diagnostics from multiple algorithms in a uniform manner, regardless of the particularities of each analytical method. This uniformity also extends to serialization mechanisms, allowing results to be transmitted between agents and stored persistently while maintaining all relevant information.

This modular architecture decouples machine learning algorithm experts from the agent architecture, allowing each group to develop their components independently. Assembly lines with distinct characteristics can employ different sets of algorithms, each parameterized according to the specifics of the process. As new methods are developed by the scientific community, they can be integrated into the system through the implementation of appropriate interfaces, allowing the system to continuously benefit from technological progress.

The execution of diagnostic algorithms by each Point Agent generates multiple analytical results that need to be consolidated and presented in an understandable way. The next subsection describes how the system aggregates these results into structured reports that support decision-making.

4.2.5 System Reports and Decision Support

The production of diagnoses by individual agents is only the first step in the analysis process. For this information to be useful in operational decision-making, the system implements a hierarchical reporting architecture that progressively aggregates results from the most granular level to a consolidated view of the station. This subsection describes the structure of the reports and the aggregation and summarization mechanisms implemented.

The system structures the diagnostic information into three hierarchical levels, each appropriate for different analysis and decision-making needs. Figure 4.9 presents the class diagram illustrating this hierarchy and the relationships between the different types of reports.

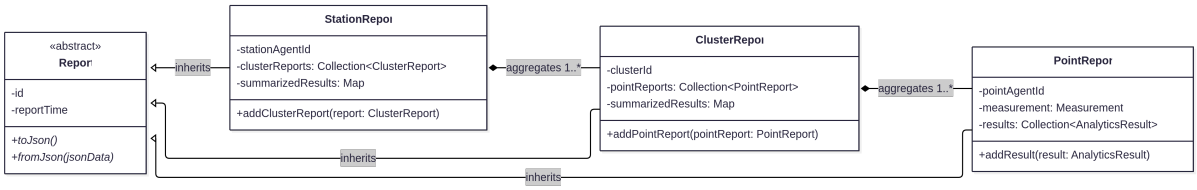

Figure 4.9: Hierarchical architecture of the reporting system

All report types share a common base structure that ensures consistency in the representation of information. This structure includes a unique report identifier, the generation timestamp, and standardized methods for serialization and deserialization, enabling persistence and efficient transmission between agents.

Point Report

The most fundamental level of the hierarchy is the Point Report, generated by each Point Agent after executing its diagnosis algorithms. This report encapsulates all relevant information for a single measurement point at a given moment, including the identification of the responsible agent, the measurement analyzed with its values and tolerance limits, and the complete collection of results produced by the various algorithms applied.

Each algorithmic result preserves not only the diagnosis or classification, but also metadata that identifies the algorithm used and the type of analysis performed. This

traceability is essential for auditing and understanding diagnostic decisions. The Point Report represents the primary source of diagnostic information in the system, forming the basis on which all subsequent aggregations are built.

Cluster Report

The second hierarchical level aggregates Point Reports from agents belonging to the same correlation cluster. The Cluster Report is generated by the cluster leader after collecting the individual reports from all members, containing the cluster identification, the Point Reports received, and summarized results representing the consensus of the individual diagnoses.

Aggregation at the cluster level capitalizes on previously identified correlations. Correlated measurement points tend to exhibit related defect patterns, and joint analysis of these points provides greater diagnosis reliability than isolated evaluation of each point.

Station Report

The highest level of the hierarchy is the Station Report, generated by the Station Agent after collecting all Cluster Reports. This report provides a holistic view of the quality status of the entire measurement station, aggregating information from all monitored points. It is typically used for high-level operational decisions, trend analysis, and communication with production management systems.

Report Aggregation and Summarization

The process of consolidating individual diagnoses into higher-level assessments operates in two distinct phases: **aggregation** consists of collecting reports from a lower level of the hierarchy, preserving all diagnostic information generated by individual agents; and **summarization** extracts consolidated insights from the aggregated reports, producing a synthetic assessment that identifies the predominant diagnosis, determines the degree of consensus among agents, and calculates the level of reliability of the conclusion.

Different types of diagnostic results require different summarization approaches: categorical results (classifications) require different techniques than probabilistic results (anomaly scores) or numerical results (quantitative predictions). To ensure this flexibility, the system implements an architecture where each diagnostic algorithm has an associated summarization strategy. As illustrated in Figure 4.10, the three categories of diagnostic algorithms (stateless, supervised, and unsupervised) have a summarization component responsible for consolidating multiple results of the same type of analysis.

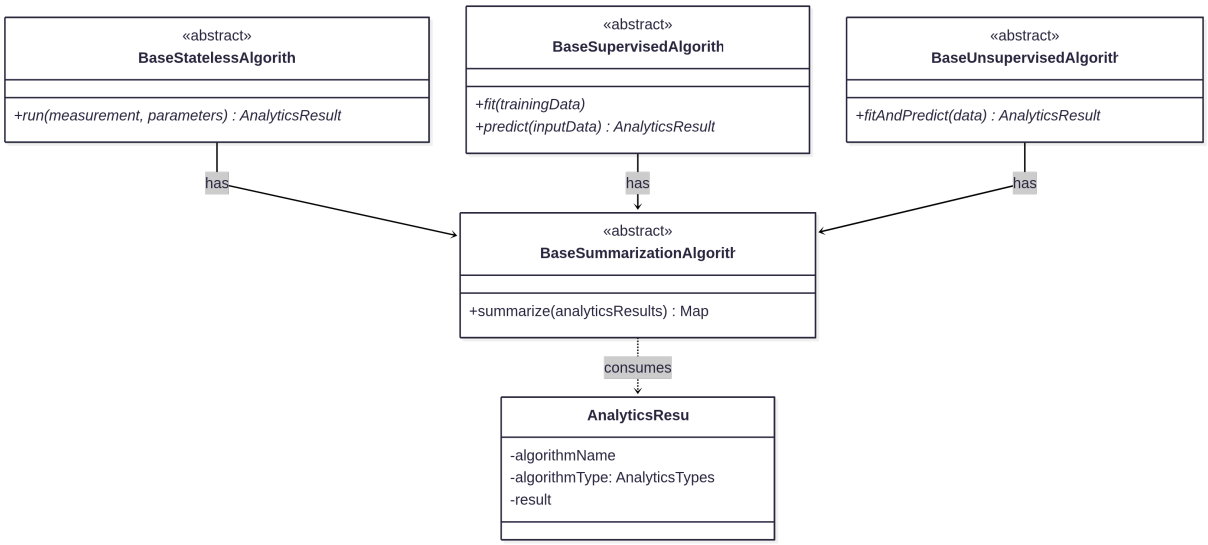

Figure 4.10: Summarization architecture

The abstract class `BaseSummarizationAlgorithm` defines the interface for all summarization strategies, receiving a collection of analytical results (`AnalyticsResult`) of the same type and producing a consolidated evaluation. This approach ensures that each type of diagnostic algorithm employs a summarization technique appropriate to the characteristics of the results it produces. Classification algorithms, for example, can use voting or consensus-based strategies, while anomaly detection algorithms employ probabilistic aggregations, and prediction algorithms apply statistical analyses. This architecture allows the summarization logic to be encapsulated and reused: multiple algorithms of the same type can share the same summarization strategy, avoiding code duplication.

Contextual Enrichment and Traceability

A key aspect of the reporting architecture is the preservation of complete traceability from individual diagnoses to consolidated assessment. Each Cluster Report maintains explicit references to the Point Reports that comprise it, and each Station Report maintains references to the Cluster Reports. This structure allows for both downward (from general to specific) and upward (from specific to general) navigation.

When a Station Report indicates an anomaly, operators can navigate through the hierarchy to identify which specific clusters contributed to this assessment, and within those clusters, which individual measurement points detected problems. This drill-down capability is essential for effective corrective action, allowing interventions to be targeted precisely at problem areas rather than requiring extensive manual investigation.

Reports also include temporal metadata that enables analysis of trends over time. Sequences of reports can reveal progressive degradation in quality, the effectiveness of adjustments made, or cyclical patterns related to production shifts or equipment maintenance.

The reporting architecture described completes the multi-agent diagnostic system, transforming raw measurements into structured and actionable assessments. The next section presents the complete operational flow of the system, integrating all the components and mechanisms described above.

4.3 Workflow and Data Flow

This section integrates the components described above, presenting the complete operational flow of the system through three complementary perspectives: the main data flow inside a station, the extension for correlations between multiple stations, and a concrete example of complete execution.

4.3.1 Main Data Flow (Intra-Station)

Intra-station flow begins when a unit passes through the inspection station and the measurement system transmits the data to the Station Agent. This agent validates the received data, checking for consistency and completeness, and stores it in the historical database. Validation identifies missing values, or inconsistencies in format, applying appropriate treatment strategies.

After validation and storage, the Station Agent checks whether it has reached the threshold configured for correlation recalculation (e.g., every 100 units processed). When this threshold is reached, it executes the correlation and clustering algorithm described in §4.2.3, calculating the correlation matrix over the historical data window and identifying clusters of correlated points. This periodic update allows the system to adapt to gradual changes in correlation patterns resulting from adjustments in the production process.

Once the measurements have been processed and the clusters defined (or updated), the Station Agent distributes the corresponding measurement data and information about the clusters to which it belongs to each Point Agent. The Point Agents then run their diagnostic algorithms in parallel, each independently processing its specific measurement point. This concurrent execution is the main mechanism for system efficiency.

After completing the diagnosis, each Point Agent checks the composition of its cluster. Agents in single clusters directly transform their Point Report into a Cluster Report and send it to the Station Agent. In multi-agent clusters, the leader election protocol based on completion timestamps is initiated. The first agent to complete announces its candidacy for leadership; the others recognize it as leader and send it their reports. The leader aggregates the reports received, applying the summarization strategies described in §4.2.5, and transmits the consolidated Cluster Report to the Station Agent.

The Station Agent waits for Cluster Reports from all active clusters, then performs a second aggregation phase that consolidates the reports at the station level. The resulting Station Report represents the complete assessment of the quality status of the inspected unit, ready for consultation by operators or integration with production management

systems.

4.3.2 Inter-Station Flow

In parallel with the intra-station flow, the Inter-Station Agent operates independently, collecting measurements from all stations in the production process. Periodically, also using a configurable window of historical data, it calculates correlations between measurement points from different stations using the same correlation algorithm, but applied to cross-station data.

When the Inter-Station Agent identifies significant correlations between points from different stations, it forms inter-station clusters and distributes this information to the relevant Station Agents. Each Station Agent incorporates the received inter-station clusters into the next data distribution to Point Agents, along with the intra-station clusters.

For Point Agents belonging to inter-station clusters, the Station Agent also attaches relevant Point Reports from previous stations, obtained through previously received Station Reports from those stations. This contextual information allows agents at later stations to consider patterns observed at preceding stations when performing their diagnostics.

Inter-station cluster processing follows the same leader election and aggregation protocol described for intra-station clusters. The fundamental difference is that cluster members reside at different stations, and the elected leader can be at any of them. The inter-station Cluster Report is sent to the Station Agent where the leader resides, who incorporates it into their Station Report. This cross-sectional analysis capability is particularly valuable for identifying defects that originate in earlier stages of the process but only manifest themselves at later stations.

4.3.3 Example of Complete Execution

To illustrate the integrated operation of the system, consider a quality control station that monitors 15 measurement points. The system is configured to recalculate correlations

every 50 units processed. In the following description, the notation Si.Pj is used to designate measurement point j at station i (for example, S1.P5 refers to point 5 at Station 1).

When unit number 50 passes through the station (Station 3), the Station Agent runs the correlation algorithm on the data from the last 50 units and identifies 5 intra-station clusters: three clusters with 4 agents each, one cluster with 2 agents, and one unitary cluster. Simultaneously, the Inter-Station Agent, which monitors 3 sequential stations (Stations 1, 2, and 3), identifies 3 inter-station clusters connecting points S1.P5 with S3.P2, points S1.P8, S2.P3, and S3. P7 crossing the three stations, and points S2.P12 with S3.P11. Note that three points from Station 3 (P2, P7, and P11) participate in inter-station clusters, each in a different cluster.

Station Agent 3 distributes the measurement data from the current unit and the definitions of the 8 clusters (5 intra-station and 3 inter-station) to the 15 Point Agents. For points involved in inter-station clusters, the agent attaches the contextual Point Reports from the previous stations: P2 receives the report from S1.P5, P7 receives the reports from S1.P8 and S2.P3, and P11 receives the report from S2.P12.

The 15 Point Agents run their diagnostic algorithms in parallel. Assuming that each agent runs 3 algorithms (limit validation, classification, and anomaly detection), there are 45 simultaneous algorithm executions. Each agent generates its Point Report containing the results of the three algorithms applied to its measurement data.

After completing the diagnostics, agents process the clusters in two different ways. In the five intra-station clusters, the first agent to finish announces their candidacy for leadership. The other members recognize the leader and send them their Point Reports. The leader aggregates the reports, applies the appropriate summarization strategy, and generates a Cluster Report. This process results in five intra-station Cluster Reports sent to the Station Agent.

In the three inter-station clusters, there is no leader election. The agent positioned at the most advanced station (Station 3) deterministically aggregates the diagnostics: P2 aggregates its diagnosis with the report from S1.P5, P7 aggregates with the reports from

S1.P8 and S2.P3, and P11 aggregates with the report from S2.P12. Each aggregation applies the summarization strategy and generates a Cluster Report, resulting in three inter-station Cluster Reports sent to the Station Agent.

Notice that points P2, P7, and P11 can also participate in intra-station clusters. In this scenario, each agent acts in two distinct roles: as a member of an intra-cluster (sending its Point Report to the elected leader) and as a deterministic aggregator of an inter-cluster (generating an independent Cluster Report). This multiple participation allows the system to capture both local correlations and correlations between stations for the same measurement points.

Station Agent 3 receives the 8 Cluster Reports (5 intra-station and 3 inter-station) and performs the final aggregation. It applies the appropriate summarization strategies to consolidate the diagnostics of all clusters, calculating reliability and consensus metrics. The resulting Station Report provides a complete assessment of the quality status of the inspected unit, integrating local analyses and the context of the entire production process.

This example demonstrates how the system coordinates multiple agents through distributed elections for intra-station clusters and deterministic aggregation for inter-station clusters. Parallelism in algorithm execution and report generation, combined with the fault containment inherent in distributed architecture, allows for efficient processing of large volumes of measurements while maintaining operational robustness.

The next chapter presents the specific implementation of this architecture applied to automotive assembly lines, detailing the technologies used, the concrete algorithms integrated, and the experimental results obtained with real production data.

Note: For information regarding source code availability, please refer to Appendix A.

Chapter 5

Experimental Setup and Results

This chapter presents the experimental setup used to validate the proposed multi-agent fault diagnosis system, as well as the results obtained by applying the system to real automotive production data. Section 5.1 describes the development environment, the implemented algorithms, the summarization techniques, and the experimental design adopted. Section 5.2 presents the quantitative results obtained through metrics that evaluate both the diagnostic quality and computational performance of the system. Finally, Section 5.3 discusses the results in the context of the established objectives and limitations identified in the literature.

5.1 Experimental Setup

This section presents the experimental setup used to validate the proposed multi-agent fault diagnosis system. First, we describe the development environment and the tools used to implement the architecture presented in Chapter 4. Next, we detail the implemented diagnosis algorithms and the summarization techniques used to consolidate diagnoses from multiple agents. Finally, the experimental design is presented, including the different system configuration profiles and the testing methodology adopted.

5.1.1 Environment and Tools

The implementation of the multi-agent system required the selection of appropriate technologies to support distributed communication, parallel data processing, and the execution of machine learning algorithms. This subsection presents the multi-agent framework used, the data processing and statistical analysis libraries, the storage infrastructure, and the hardware specifications of the experimental environment.

Multi-Agent Framework

The system was developed using the SPADE (*Smart Python Agent Development Environment*) [24] framework, a Python-based platform that implements the FIPA (*Foundation for Intelligent Physical Agents*) [25] standard. SPADE offers asynchronous communication between agents via the XMPP protocol [26], modular architecture for distributed systems, and support for multiple concurrent behaviors per agent, facilitating integration with data processing and statistical analysis libraries. The choice is justified by its compliance with international standards for multi-agent systems, ensuring interoperability, and by asynchronous communication that allows efficient parallel processing.

Libraries and Dependencies

The development used established Python libraries for data processing and implementation of machine learning algorithms:

- **Pandas** [27] and **NumPy** [28]: Efficient tabular data manipulation and numerical operations
- **Scikit-learn** [29]: Implementation of machine learning algorithms
- **NetworkX** [30]: Graph analysis in the detection of maximum clicks during the correlation-based clustering process
- **PyMongo** [31]: Communication interface with the MongoDB database

- **JSON and UUID**: Data serialization and message tracking to ensure persistence and traceability of communications between agents

Storage Infrastructure

Data storage was implemented with **MongoDB** [32], a document-oriented NoSQL database. MongoDB was chosen because of its ability to store documents with variable schemas, an essential feature given that different measurement points have different settings for limits and measurement types. Additionally, native support for capped collections allows automatic history management with periodic cleaning based on configurable thresholds, which is essential for operating in environments with storage restrictions. MongoDB's performance is suitable for real-time operations with moderate data volumes characteristic of industrial environments.

The infrastructure was implemented using **Docker** [33], a containerization platform that provides environment isolation, portability between different operating systems, and ease of replication of the experimental environment. Containerization allows the storage system to be easily reconfigured or scaled as needed, without impacting the host operating system, facilitating both development and potential deployment in industrial environments.

Hardware Specifications

The experiments were performed in an environment with the following specifications:

- **Processor**: 13th Gen Intel(R) Core(TM) i9-13900HX
- **RAM**: 16GB
- **Storage**: 1TB SSD
- **Operating System**: Windows 11
- **Virtualization**: Docker Engine v28.4.0

5.1.2 Diagnosis Algorithms

The system implements three complementary diagnostic algorithms, each responsible for a different perspective of defect analysis. This subsection describes the mechanics of each algorithm, with specific parameters presented later in Subsection 5.1.4.

Classification Algorithm

The classification algorithm categorizes measurements based on the tolerance limits described in Section 3.2.2 (Chapter 3). Using the hierarchical limit structure (LR, LT, LS, US, UT, UR), the algorithm classifies a measurement x into one of the following categories:

- **Within Specification**: $LS \leq x \leq US$ (ideal quality zone)
- **Within Tolerance**: $LT < x < LS$ or $US < x < UT$ (alert zone)
- **Rejected**: $x \leq LR$ or $x \geq UR$ (critical rejection zone)
- **Not Classified**: missing value or undefined limits

Because it is based on established limits rather than statistical inference, the algorithm produces deterministically correct results according to engineering specifications.

Anomaly Detection Algorithm

The anomaly detection algorithm uses the Modified Z-Score with MAD (Median Absolute Deviation) to identify measurements with atypical behavior compared to recent history. This robust approach is less sensitive to outliers than methods based on standard deviation, making it more suitable for non-normal distributions common in industrial processes [34], [35].

The modified z-score is calculated as:

$$z_{MAD} = \frac{0.6745 \times (x - M)}{MAD} \tag{5.1}$$

where x is the measured value, M is the median of the historical window of size w (configurable), and MAD is defined as:

$$MAD = \text{median}(|x_i - M|) \tag{5.2}$$

The value 0.6745 normalizes the MAD for consistency with the standard deviation in normal distributions. A measurement is considered anomalous if:

$$|z_{MAD}| > \theta_{MAD} \tag{5.3}$$

where θ_{MAD} is the configurable threshold. The algorithm implements proximity gating to suppress alarms for anomalies far from the tolerance limits. The normalized distance to the nearest limit is:

$$d_{norm} = \frac{\min(|UR - x|, |x - LR|)}{UR - LR} \tag{5.4}$$

The anomaly is suppressed if $d_{norm} > \rho$ and $|z_{MAD}| < z_{huge}$, where ρ is the proximity ratio and z_{huge} is the override threshold, both of which are configurable. When $MAD = 0$ (identical historical values), the algorithm automatically uses traditional z-score with standard deviation as a fallback:

$$z_{std} = \frac{x - \mu}{\sigma} \tag{5.5}$$

where μ and σ are the mean and standard deviation of the historical window, with the same threshold condition $|z_{std}| > \theta_{std}$ applied.

Prediction Algorithm

The prediction algorithm uses linear regression to estimate when a measurement point will violate tolerance limits based on the observed trend. The model fits a straight line to historical data:

$$\hat{y} = ax + b \tag{5.6}$$

where a is the slope, b is the intercept, x is the time index, and $\hat{y}$ is the predicted value. To calculate when a limit will be reached:

$$x_{limite} = \frac{L - b}{a} \tag{5.7}$$

where L is the limit value (UR or LR). The number of vehicles remaining until the violation is:

$$n_{remaining} = \lceil x_{limit} - x_{current} \rceil \tag{5.8}$$

The algorithm implements directional prediction: if $a > 0$ (increasing trend), only the upper limit UR is considered; if $a < 0$ (decreasing trend), only LR is considered. When $|a| < \epsilon_{slope}$, the algorithm does not produce a prediction, indicating insufficient trend. Predictions beyond a maximum horizon h_{max} are classified as "beyond horizon."

5.1.3 Summarization Techniques

The system implements distinct summarization strategies for each category of diagnostic algorithm. The need for multiple strategies arises from the different characteristics of the results produced: classification algorithms generate discrete categories, anomaly detection algorithms produce continuous probabilities, and prediction algorithms return numerical values. Each type of result requires specific consolidation approaches to produce reliable unified assessments.

This subsection presents the main techniques implemented, explaining the motivation for each approach, its distinctive characteristics, and the scenarios where they are most appropriate. For each algorithm category, the system offers multiple strategies that represent different trade-offs between computational simplicity and analytical sophistication.

Summary of Classifications

Classification algorithms assign each measurement to predefined categories, such as "compliant," "alert," or "critical." When multiple agents classify correlated points, the system needs to consolidate these individual classifications into a final classification. Two voting strategies are implemented for this purpose.

The **Simple Voting** treats all clusters equally, regardless of the number of agents they contain. This democratic approach is appropriate when each cluster represents an equally valid perspective of the process. For each candidate classification c, the proportion of clusters that chose it is calculated as:

$$p(c) = \frac{\text{count}(c)}{n} \tag{5.9}$$

where n is the total number of clusters. The classification with the highest proportion is chosen as the final result, and this proportion serves as a confidence metric. For example, if 4 out of 5 clusters classify a measurement as "compliant," the confidence would be 0.8.

Weighted Voting recognizes that larger clusters containing more agents can provide more robust evidence. Each cluster is given a weight proportional to its size, subject to configurable limits w_{min} and w_{max} that prevent a single very large cluster from completely dominating the decision. The weighted proportion for each classification is:

$$p_w(c) = \frac{\sum_{i:d_i=c} w_i}{\sum_{j=1}^{n} w_j} \tag{5.10}$$

where w_i represents the weight of cluster i and d_i its classification. This strategy is particularly useful when clusters of very different sizes participate in the decision, allowing evidence from multiple correlated points to have greater influence.

The choice between simple and weighted voting depends on the characteristics of the production process. In processes where correlations between points are homogeneous, simple voting is appropriate. When clusters have significantly different sizes and larger clusters represent more reliable evidence, weighted voting is preferable.

Anomaly Summarization

Anomaly detection algorithms produce probabilities or scores indicating the likelihood that a measurement is anomalous. Consolidating these probabilities from multiple agents requires approaches that preserve the probabilistic nature of the information while producing a clear final decision.

The **Majority Strategy** converts each probability into a binary decision (normal/anomalous) by comparing it to a decision threshold $\theta_{decision}$, typically set to 0.5. The proportion of agents indicating an anomaly is then calculated as:

$$p_{anomaly} = \frac{\text{number of agents with } p_i \geq \theta_{decision}}{n} \tag{5.11}$$

where n is the total number of agents and p_i is the anomaly probability reported by agent i. Formally, this can be expressed as:

$$p_{anomaly} = \frac{1}{n} \sum_{i=1}^{n} \begin{cases} 1, & \text{if } p_i \geq \theta_{decision} \\ 0, & \text{otherwise} \end{cases} \tag{5.12}$$

This approach is simple and interpretable: if most agents detect an anomaly, the final result is "anomalous." It is appropriate when clear binary decisions are needed and uncertainty in individual probabilities is not critical. For example, if 7 out of 10 agents report anomalies, then $p_{anomaly} = 0.7$.

The **Weighted Probability** Strategy preserves probabilistic information by calculating the average of the individual probabilities:

$$p_{avg} = \frac{1}{n} \sum_{i=1}^{n} p_i \tag{5.13}$$

where p_i is the anomaly probability from agent i. This average can optionally be adjusted by a confidence factor based on cluster size, giving greater weight to larger clusters. This strategy is more sophisticated than majority voting because it considers the magnitude of probabilities, not just their direction. For example, three clusters with probabilities

$\{0.9, 0.85, 0.8\}$ yield $p_{avg} = 0.85$, while three clusters with probabilities $\{0.55, 0.52, 0.51\}$ yield $p_{avg} = 0.53$. Both constitute a "majority," but the weighted strategy captures that the first case shows much stronger evidence of an anomaly.

Confidence in the aggregate decision is quantified through the variance of the individual probabilities. First, the variance is computed:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (p_i - p_{avg})^2 \tag{5.14}$$

The confidence metric is then derived by inverting the normalized variance:

$$\text{confidence} = 1.0 - \min(1.0, \sigma^2) \tag{5.15}$$

Low variance (high agreement between agents) results in high confidence, approaching 1.0. High variance (discordant agents) reduces confidence, signaling that the decision requires further validation. This metric is particularly valuable for identifying ambiguous situations where automatic diagnosis may be insufficient. For instance, if all agents report probabilities near 0.9, $\sigma^2 \approx 0$ and confidence ≈ 1.0; if agents report diverse values like $\{0.1, 0.5, 0.9\}$, variance is high and confidence is low.

Prediction Summarization

Prediction algorithms generate numerical values, such as the estimated number of units until tolerance violation. Consolidating predictions from multiple agents requires transforming value distributions into severity and urgency assessments.

The **Threshold-Based Strategy** compares descriptive statistics of predictions against predefined thresholds to classify the situation into severity levels. Specifically, it considers the minimum predicted value (n_{min}) and the mean of the predictions (n_{mean}), comparing them with critical thresholds ($\tau_{crit,min}$, $\tau_{crit,mean}$) and warning thresholds ($\tau_{warn,min}$, $\tau_{warn,mean}$):

$$\text{level} = \begin{cases} \text{Critical} & \text{se } n_{min} < \tau_{crit,min} \text{ ou } n_{mean} < \tau_{crit,mean} \\ \text{Warning} & \text{se } n_{min} < \tau_{warn,min} \text{ ou } n_{mean} < \tau_{warn,mean} \\ \text{Normal} & \text{otherwise} \end{cases} \tag{5.16}$$

This approach is conservative, considering both the most pessimistic prediction (minimum) and the central trend (average). For example, if a cluster predicts a violation in 5 units but the average is 50, the system flags an alert based on the minimum, promoting preventive action. The thresholds are configurable according to the organization's maintenance policies.

The **Weighted Scoring Strategy** uses a more sophisticated cumulative scoring system, considering multiple criteria simultaneously: proximity of the minimum prediction to critical thresholds, proportion of predictions below each threshold, and variability between predictions. A raw score S_{raw} is calculated and optionally adjusted by a confidence factor based on cluster size:

$$S_{adjusted} = S_{raw} \times (0.5 + 0.5 \times c_{cluster}) \tag{5.17}$$

where $c_{cluster}$ is the normalized confidence of the cluster. This strategy produces a continuous scale of severity rather than discrete categories, allowing for more granular prioritization of interventions. It is appropriate when resources for preventive maintenance are limited and decisions need to be prioritized accurately.

The choice between threshold-based and scoring strategies depends on operational needs. Thresholds are simpler and produce clear categories for decision-making, suitable when response protocols are well defined for each level. Scoring is more sophisticated and appropriate when multiple factors must be balanced and precise prioritization is required.

These summarization strategies are designed to be configurable and extensible. New criteria, weights, or aggregation functions can be incorporated as specific requirements emerge, maintaining the modular architecture of the system. The selection of specific strategies is done through configuration profiles, allowing different production contexts to

choose the most appropriate approaches for their processes and operational policies.

5.1.4 Experiment Design

This subsection describes the experimental design used to validate the proposed multi-agent system. The system configuration profiles, the baseline system for comparison, and the testing methodology adopted, including the selected evaluation metrics, are presented.

Multi-Agent System: Configuration Profiles

The multi-agent system was evaluated under three distinct configuration profiles, each optimized for different operational scenarios and exploring trade-offs between computational resource usage and analytical capacity. Table 5.1 presents a detailed comparison of the main parameters of each profile.

The **Base Settings** profile uses conservative parameters suitable for historical data of approximately 100 vehicles, representing a typical deployment scenario with moderate resources. With storage of 25MB for measurements and 512MB for reports, a window of 50 measurements for anomaly detection, and simple consensus strategies without weighting, this profile prioritizes operational stability and ease of initial deployment. The summarization strategies (simple voting for classification, majority for anomaly, threshold for prediction) minimize computational complexity while maintaining basic diagnostic effectiveness.

The **Low Storage** profile was optimized for environments with severe resource constraints, suitable for historical data of approximately 20 vehicles. It uses minimal storage (5MB measurements, 10MB reports), reduced windows (20 for anomaly, 10 for prediction), aggressive cleanup at 60% capacity, reduced timeouts (60s), and frequent processing every 5 cars. The thresholds were adjusted to be slightly more conservative (MAD threshold 3.5, proximity ratio 0.40) to compensate for the smaller amount of available historical data. This profile sacrifices long-term analysis capability in favor of minimal memory footprint, being suitable for embedded devices or edge computing scenarios with limited

Table 5.1: Comparison of Multi-Agent System Configuration Profiles

Parameter	Base Settings	Low Storage	High Storage
Estimated Capacity	∼100 cars	∼20 cars	∼5000 cars
Storage (Measurements)	25 MB	5 MB	2 GB
Storage (Reports)	512 MB	10 MB	5 GB
Max Documents (Measurements)	24,000	4,800	1,200,000
Max Documents (Reports)	1,000,000	2,000	10,000,000
Anomaly Window Size	50	20	100
Prediction Window Size	25	10	35
MAD Threshold	3.0	3.5	3.0
Proximity Ratio	0.30	0.40	0.25
Max Prediction Horizon	800	300	3000
Confidence Threshold	0.6	0.6	0.7
Anomaly Threshold	0.4	0.4	0.45
Weighted Voting	No	No	Yes
Classification Strategy	Consensus	Consensus	Consensus
Anomaly Strategy	Majority	Majority	Weighted Prob.
Prediction Strategy	Threshold	Threshold	Weighted
Point Agent Buffer	5	3	25
Message Timeout (s)	180	60	300
Max Connection Pool	75	20	300
Processing Milestone	10	5	25
Cleanup Threshold	70%	60%	95%
Cleanup Check Interval	every 5 units	every 2 units	every 100 units

Notes: Estimated capacities calculated based on: (i) average measurement document size (∼1KB), (ii) number of points per vehicle (62 total), (iii) hierarchical report overhead, and (iv) 20% safety margins for MongoDB metadata and indexes.

resources.

The **High Storage** profile was configured for long-term analysis and maximum analytical capacity, suitable for historical data of approximately 5000 vehicles. It uses extensive storage (2GB measurements, 5GB reports), a wide window of 100 measurements for anomaly detection, conservative cleanup only at 95% capacity, a pool of 300 connections for high throughput, and batch processing every 25 cars. The parameters were adjusted to exploit the larger amount of available data: extended prediction horizons (3000 cars), more stringent thresholds (MAD 3.0, proximity 0.25), and sophisticated summarization strategies including weighted voting by cluster size for classification, weighted probability with confidence adjustment for anomaly detection, and weighted scoring with confidence adjustment for prediction. This profile maximizes analytical capacity while maintaining adequate responsiveness for real-time operation.

The assignment of summarization strategies by profile reflects the trade-offs between simplicity and sophistication: profiles with limited resources (Base and Low) use deterministic techniques based on counting and direct comparison, while the High profile employs weighted probabilistic techniques that exploit the largest amount of historical data to produce potentially more refined diagnostics.

Baseline System

To establish a benchmark for comparison, a baseline system was implemented that runs the same three diagnostic algorithms (classification, anomaly detection, and prediction) used by the multi-agent system, but without the collaboration, consensus, and hierarchical reporting mechanisms characteristic of the MAS architecture.

The baseline system processes each measurement point completely independently and in parallel. When a vehicle passes through the inspection station, the measurement data is distributed simultaneously for parallel processing of all points. Each point independently executes the three diagnostic algorithms on its local data. There is no communication or exchange of information between measurement points, analysis of correlations between points, or cluster formation. Each algorithm generates an individual diagnosis without

aggregation or consolidation, and the results are stored directly in the database without generating structured reports.

The baseline system uses exactly the same algorithmic parameters as the multi-agent system for each corresponding experimental profile, including historical window sizes, detection thresholds, prediction horizons, and classification limits defined by engineers. This parametric standardization ensures that differences in results are attributable solely to the presence or absence of the collaborative mechanisms of the multi-agent architecture, and not to differences in the configuration of the underlying algorithms.

Unlike the multi-agent system, which produces consolidated diagnostics through voting, consensus, and hierarchical reports, the baseline generates only individual, disaggregated diagnostics. There is no summarization at the station level, clusters, or synthesis of multiple algorithmic perspectives. This intentional simplicity allows us to evaluate the specific added value of the collaborative mechanisms of the multi-agent system.

Test Methodology

The experiments were conducted following a systematic methodology that allows for fair comparison between the baseline system and the different configurations of the proposed multi-agent system. Figure 5.1 illustrates the overview of the case study, showing the two measurement stations in the production flow and the architecture of the multi-agent system superimposed on the physical process.

The four systems were tested: one baseline system and three multi-agent system configurations (Base Settings, Low Storage, and High Storage), totaling four complete executions. Each system was given a time window of 12 hours to process as many vehicles as possible from a dataset of real production data. The data originates from two sequential measurement stations on the automotive assembly line, as described in Chapter 3. Measurement Station 1 performs 42 distinct measurements per vehicle, while Measurement Station 2 executes 20 specific measurements, totaling 62 measurement points per processed vehicle.

The experimental methodology follows a sequential processing approach where vehicles

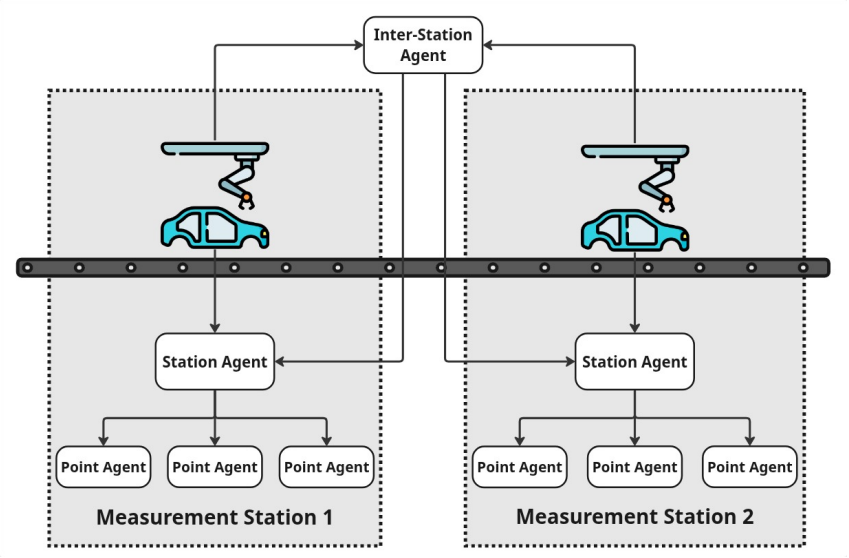

Figure 5.1: Overview of the case study with the multi-agent system architecture.

are diagnosed one at a time. For each vehicle, the system first receives measurement data from Station 1, performs the complete diagnosis for that station, then receives data from Station 2 for the same vehicle, and performs the corresponding diagnosis. Only after completing the full diagnostic cycle for vehicle n does the system proceed to process vehicle $n + 1$. This sequential approach simulates realistic operational deployment where vehicles progress through the assembly line in order.

Each system configuration was executed independently with a 12-hour time limit. The performance metric of interest is the number of vehicles successfully diagnosed within this time window, which directly reflects the computational efficiency and scalability of each configuration. Systems with more efficient algorithms, optimized parameters, and appropriate resource allocation can process more vehicles in the same time period.

Between each system execution, the environment was completely reset: all MongoDB collections were removed and recreated, Docker services were restarted, and caches and buffers were cleared, ensuring total independence between executions. Each system started with an empty database and progressively accumulated historical data as vehicles were processed.

All vehicles were processed sequentially without discarding initial data. This decision

is justified because the algorithms do not require prior supervised training: anomaly and prediction algorithms use historical windows that are populated progressively as new vehicles are processed, reaching full operational capacity after processing w vehicles, where w is the configured window size. During the initial phase (first w vehicles), the algorithms operate with partial historical data, gradually improving their diagnostic capability as the window fills.

The experiments were conducted in single-run execution for each system. Multiple repetitions were not considered necessary given that processing is completely deterministic for fixed input data, there are no random or stochastic components in the implemented algorithms, and the 12-hour time window provides sufficient operational time to characterize system performance under realistic conditions. The variability observed in results reflects the real variability of the production process and the different computational characteristics of each configuration profile.

All experiments were executed on the same physical machine specified in Subsection 5.1.1, using isolated Docker containers for MongoDB services. This uniformity eliminates variability related to hardware differences, operating system configuration, or state of other concurrent processes. The 12-hour execution window was enforced by system monitoring that gracefully terminated each experiment at the time limit, recording the total number of vehicles successfully diagnosed.

Evaluation Metrics

The evaluation of the systems focused on metrics that prove the added value of multi-agent architecture and its operational feasibility. Five main categories of metrics were used.

The classification algorithm based on tolerance limits was used as a reference (ground truth) for validating the other algorithms. This choice is justified because the classification is deterministic and based directly on established engineering specifications, the tolerance limits are defined by experts and represent objective acceptance criteria used in the actual production line, and violations of these limits definitively indicate the need for

intervention, regardless of complementary statistical analyses.

1. Diagnostic Validation with Reference Anchor

Using the classification algorithm as ground truth, the performance of the anomaly detection algorithm was calculated using recognized statistical metrics:

- **Sensitivity (Recall)**: Ability to detect cases that violate tolerance limits

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{5.18}$$

  where TP (true positives) represents anomalies correctly identified when the classification indicates rejection, and FN (false negatives) represents undetected limit violations.

- **Specificity**: Ability to avoid generating false alarms for cases within specification

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{5.19}$$

  where TN (true negatives) represents correctly identified normal cases, and FP (false positives) represents anomalies detected without violating thresholds.

- **Precision**: Proportion of detected anomalies that correspond to actual problems

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.20}$$

2. Early Warning Capability

To assess whether the system can anticipate problems before they violate critical limits, cases were analyzed where the anomaly detection algorithm identified atypical patterns while the classification algorithm still indicated compliance. Confirmation was performed through temporal analysis in sliding windows of 10 consecutive vehicles. The following were quantified:

- **Confirmed early warnings**: Anomaly followed by rejection within the window

- **Accuracy of Early Warnings (EW)**:

$$\text{Accuracy EW} = \frac{\text{Confirmed EW}}{\text{Confirmed EW} + \text{False EW}} \tag{5.21}$$

- **Average Lead Time**: Average number of vehicles between detection and violation

$$\text{Average Lead Time} = \frac{1}{n}\sum_{i=1}^{n}(t_{violation,i} - t_{detection,i}) \tag{5.22}$$

3. Complementarity between Algorithms

To quantify the value of combining multiple algorithms, each measurement was categorized into scenarios: diagnostic reinforcement (both identify problem), early warning (only anomaly detects), sudden violation (only classification detects), or normal operation (both indicate normality). The percentage distribution shows the proportion in which each algorithm contributes unique versus redundant information.

4. Operational Latency

To assess industrial viability, baseline time (parallel processing without collaboration, measured until completion of the last point), complete MAS time (end-to-end latency including clustering and consensus), and additional time were measured:

$$\text{Overhead} = \frac{T_{MAS} - T_{baseline}}{T_{baseline}} \times 100\% \tag{5.23}$$

Statistics include mean, median, and 95th percentile, with outliers above the 99th percentile removed.

5. Formation and Effectiveness of Clusters

To validate the fact that the architecture identifies significant correlations, the total number of clusters formed (configurable Pearson threshold) and the average size of the clusters were analyzed:

$$\bar{s} = \frac{1}{N}\sum_{i=1}^{N}|C_i| \tag{5.24}$$

where $|C_i|$ is the size of cluster i, and high confidence rate:

$$\text{HC Rate} = \frac{\text{Clusters with High Confidence}}{\text{Total Clusters}} \tag{5.25}$$

Methodological Limitations It is important to explicitly acknowledge the limitations of this experimental approach. Only the classification algorithm has validation based on established engineering specifications, while anomaly and prediction algorithms lack complete external validation against human inspections or alternative systems, being evaluated primarily through consistency with classification results. Early warnings, in particular, cannot be fully validated as they represent predictions about future events that may or may not materialize depending on interventions taken.

The 12-hour execution window, while providing a controlled comparison of system efficiency, represents a limited operational timeframe. This duration may not capture long-term variations such as seasonal supplier changes, progressive equipment degradation, or systematic process adjustments occurring over weekly or monthly scales. The time-limited execution inherently favors configurations optimized for throughput over those potentially offering superior diagnostic quality with longer processing times. Different system configurations might exhibit different performance characteristics under varying time pressures, and the relative rankings observed within the 12-hour window might shift under different temporal constraints.

The sequential processing methodology (complete diagnosis of vehicle n before starting vehicle $n+1$) accurately reflects realistic deployment constraints but prevents evaluation of alternative processing strategies such as pipeline parallelization where multiple vehicles could be at different diagnostic stages simultaneously. The throughput metric (vehicles diagnosed per time unit) captures computational efficiency but does not directly measure diagnostic quality or clinical accuracy, creating potential trade-offs between speed and thoroughness that are not fully explored.

No manual inspection by quality specialists was conducted to confirm diagnoses in ambiguous cases, validate early warnings against operators' tacit knowledge, or adjudicate

disagreements between algorithms, limiting validation to automated metrics. The baseline system is an artificial construction for comparison purposes, not an actual alternative system previously in operation. Comparisons demonstrate the specific value of multi-agent collaboration but do not establish absolute superiority against other possible industrial approaches such as expert systems or centralized deep learning.

The absence of multiple repetitions, while justified by deterministic processing and the substantial volume of data, prevents formal analysis of statistical variability through confidence intervals or significance tests. Single-run execution means that potential system-level variations (such as database performance fluctuations, memory management edge cases, or timing-dependent race conditions) remain uncharacterized, though the deterministic algorithmic core minimizes such concerns.

Finally, the results are specific to these measurement stations of this particular manufacturer. Generalization to other assembly lines, other manufacturers, or other types of industrial processes requires additional validation with appropriate data. These limitations are inherent to experimental studies with real industrial data where complete ground truth and controlled repetitions are impractical. The limitations do not invalidate the obtained results but appropriately contextualize the scope of possible conclusions and guide appropriate interpretation of the findings.

5.2 Results

This section presents the experimental results obtained from the evaluation of the multi-agent system using three configuration profiles: Base Settings, High Storage, and Low Storage. The results are organized to answer the key validation questions established in the experimental methodology: diagnostic accuracy, early warning capability, complementarity between algorithms, computational performance, and effectiveness of cluster formation. All analyses use the Classification algorithm as a ground truth reference for validation, given its deterministic nature based on engineering specifications.

5.2.1 Diagnostic Validation with Classification Anchor

The diagnostic performance of the Anomaly Detection algorithm was evaluated using the Classification results as ground truth, calculating standard metrics of Sensitivity (ability to detect real rejections), Specificity (ability to avoid false alarms when measurements are within specification), and Accuracy (proportion of detected anomalies that correspond to real problems). Figure 5.2 shows the validation metrics across the three configuration profiles.

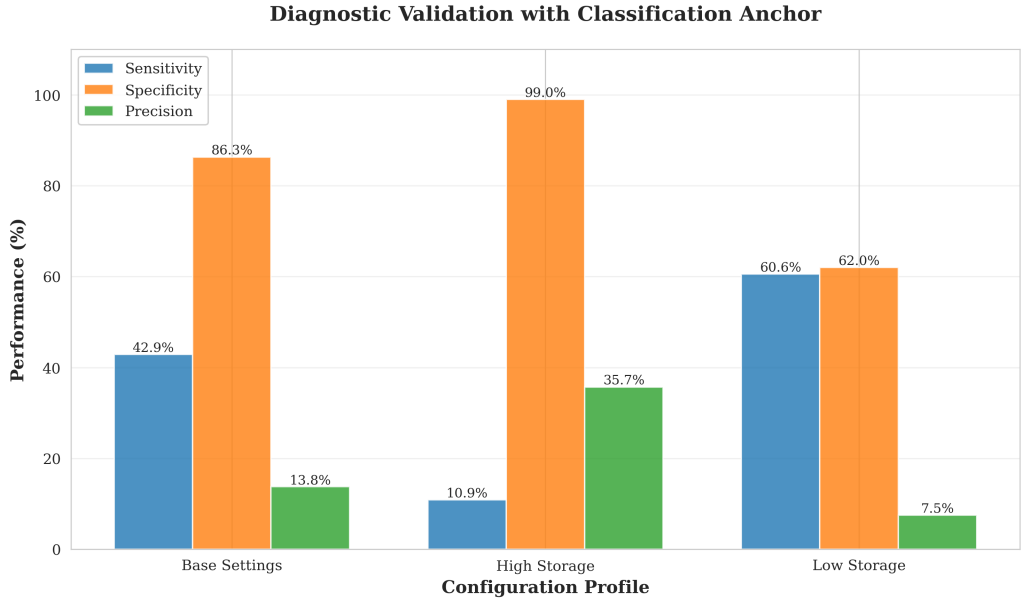

Figure 5.2: Diagnostic validation metrics comparing the performance of Anomaly Detection against the Classification anchor across three configuration profiles.

The results reveal fundamental trade-offs between detection sensitivity and false alarm rate. The High Storage profile achieved the highest Specificity (99.0%) and Accuracy (35.7%), indicating that when this configuration detects an anomaly, there is a 35.7% probability that it corresponds to an actual specification violation. However, this conservative approach resulted in the lowest Sensitivity (10.9%), meaning that approximately 89% of actual rejections were not flagged as anomalies before violating limits.

In contrast, the Low Storage profile achieved the highest Sensitivity (60.6%), detecting approximately 60% of rejections before or at the time of limit violation. This aggressive

detection came at the cost of substantially lower Precision (7.5%), indicating that 92.5% of anomaly alerts in this configuration were false positives that did not correspond to immediate specification violations. The Base Settings profile provided intermediate performance across all metrics, with 42.9% Sensitivity, 86.3% Specificity, and 13.8% Accuracy.

Table 5.2 presents the complete validation statistics, including the total number of rejected measurements and the elements of the confusion matrix for each profile.

Table 5.2: Detailed validation metrics for Anomaly Detection using Classification as anchor

Profile	**Rejected**	**TP**	**FN**	**TN**	**FP**
Base Settings	6,695	2,873	3,822	113,215	18,042
High Storage	3,744	408	3,336	70,485	709
Low Storage	11,388	6,900	4,488	138,421	85,397

These results demonstrate that no single configuration dominates across all metrics. Instead, each profile represents a valid operating philosophy suited to different industrial contexts, a point elaborated upon in Section 5.3.1.

5.2.2 Early Warning Capability

A critical capability of the proposed system is early warning detection, where the Anomaly Detection algorithm identifies potential problems before they result in specification violations. To validate this capability, temporal analysis was performed on sequential measurement data, identifying cases where an anomaly was detected while the Classification indicated "Within Specification," followed by rejection within a window of 10 vehicles. Figure 5.3 shows the number of confirmed early warnings and their average lead time across profiles.

The Low Storage profile generated 5,919 confirmed early warnings, representing 36.2% of all analyzable sequences where both algorithms provided valid diagnoses. These warnings provided an average lead time of 4.8 vehicles, offering approximately 4-5 production cycles for intervention before specification violation. However, the early warning accuracy

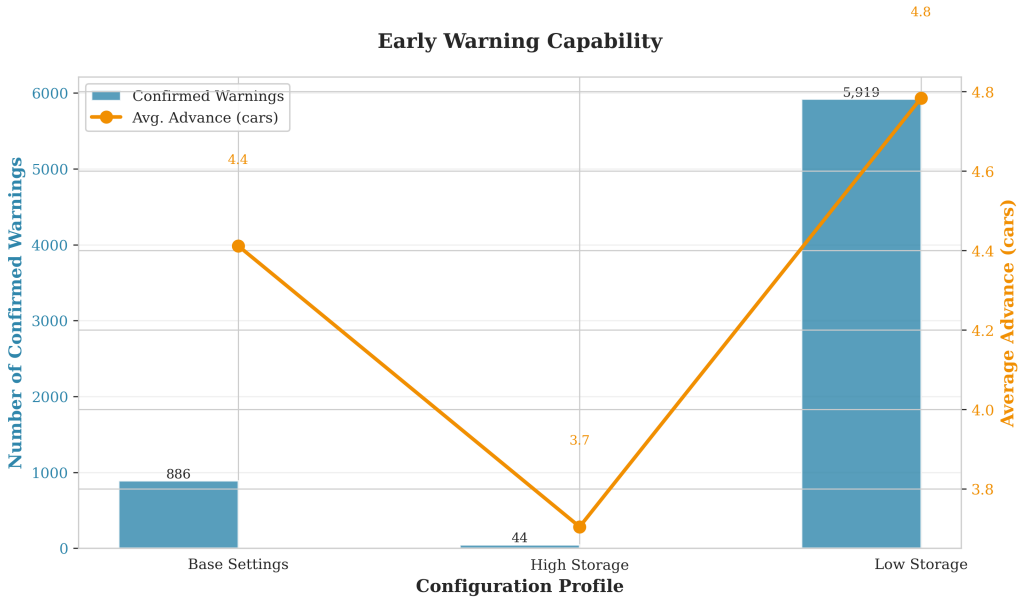

Figure 5.3: Early warning capability through configuration profiles.

for this profile was 7.0%, meaning that 93% of anomaly alerts during normal operation did not result in subsequent rejections within the observation window.

The Base Settings profile generated 886 confirmed early warnings (13.0% of sequences) with slightly lower accuracy (4.9%) and 4.4 vehicles of average lead time. The High Storage profile, consistent with its conservative parameterization, produced only 44 confirmed early warnings (1.0% of sequences) but achieved marginally higher accuracy (6.0%) with 3.7 vehicles of average lead time.

The lead time of 4-5 vehicles is relatively consistent across all profiles, suggesting that this represents an intrinsic characteristic of defect progression dynamics in the body shop process rather than a configuration-dependent parameter. Whether this lead time is operationally sufficient depends on the production cycle time and intervention procedures available at each manufacturing facility.

Table 5.3 provides complete statistics on early warning performance, including false early warnings (anomalies that did not result in rejections within the observation window).

The low accuracy of early warnings across all profiles (4.9–7.0%) represents a significant limitation that must be considered when deploying such systems in production

Table 5.3: Early warning capability statistics across configuration profiles

Profile	Confirmed	False	Precision	Avg. Lead Time	Max. Lead Time
Base Settings	886	17,156	4.9%	4.4 vehicles	10 vehicles
High Storage	44	665	6.0%	3.7 vehicles	9 vehicles
Low Storage	5,919	78,576	7.0%	4.8 vehicles	10 vehicles

environments, as discussed in Section 5.3.2.

5.2.3 Complementarity between Algorithms

To evaluate the value of combining multiple diagnostic algorithms, each measurement was categorized into one of four scenarios based on the agreement between the Classification and Anomaly Detection algorithms: (1) Reinforcement, where both algorithms identify a problem; (2) Early Warning, where Anomaly Detection signals a problem while Classification indicates normal operation; (3) Classification Only, where Classification identifies a violation without prior anomaly detection; and (4) Both OK, where both algorithms indicate normal operation. Figure 5.4 shows the distribution of these scenarios across the profiles.

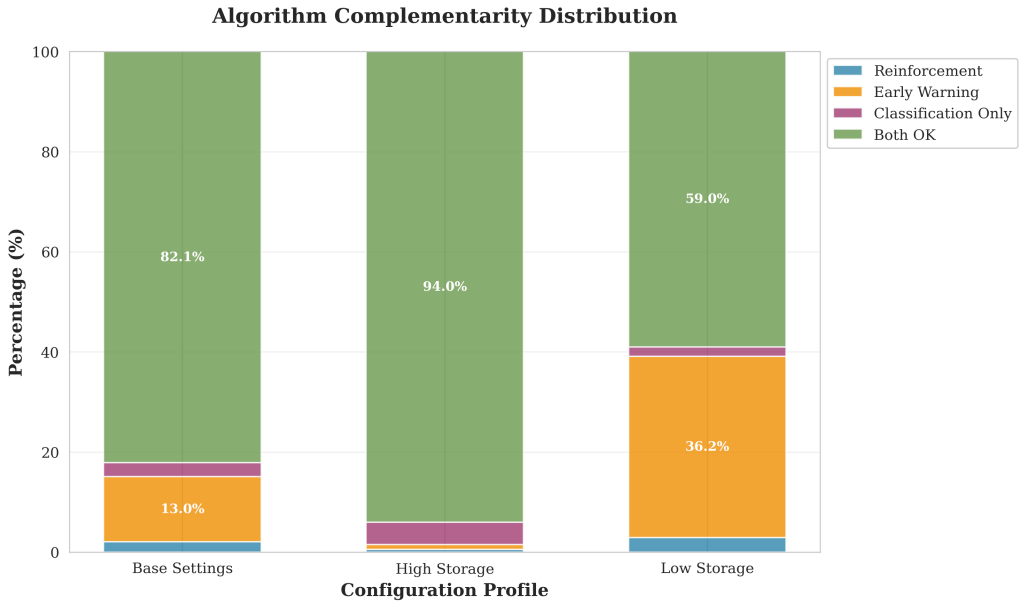

Figure 5.4: Distribution of complementarity scenarios across configuration profiles.

The complementarity distribution reveals dramatically different operational characteristics across profiles. The High Storage profile exhibited 94.0% of "Both OK" scenarios, indicating that the vast majority of measurements were classified as normal by both algorithms. Only 1.0% of cases represented early warning scenarios, 4.4% were detected only by Classification, and 0.5% showed reinforcement between algorithms. This distribution reflects the conservative parameterization of High Storage, which prioritizes avoiding false alarms over maximizing detection coverage.

As a notable contrast, the Low Storage profile showed 59.0% "Both OK," with substantial complementarity manifested as 36.2% early warning scenarios, 2.9% reinforcement, and 1.9% detections by Classification alone. This aggressive profile generates frequent anomaly alerts during normal operation, consistent with its design philosophy of maximizing sensitivity even at the cost of numerous false positives.

The Base Settings profile achieved intermediate distribution with 82.1% "Both OK," 13.0% early warnings, 2.8% Classification only, and 2.1% reinforcement. Table 5.4 shows the absolute counts for each scenario.

Table 5.4: Detailed counts of complementarity scenarios across profiles

Profile	**Total**	**Both OK**	**Early Warn**	**Class Only**	**Reinforcement**
Base Settings	137,887	113,215 (82.1%)	17,918 (13.0%)	3,822 (2.8%)	2,932 (2.1%)
High Storage	74,973	70,485 (94.0%)	730 (1.0%)	3,336 (4.4%)	422 (0.5%)
Low Storage	235,095	138,421 (59.0%)	85,131 (36.2%)	4,488 (1.9%)	7,055 (2.9%)

These results demonstrate that the complementarity between algorithms is highly dependent on the parameterization strategy. The 36.2% early warning rate of the Low Storage profile, while reflecting numerous false positives as shown in Section 5.2.2, also indicates that the system is actively attempting predictive detection rather than purely reactive classification. The implications of these different operating modes are discussed in Section 5.3.1.

5.2.4 Computational Performance and Latency

Computational performance was evaluated by measuring end-to-end latency from measurement reception to station report generation. Baseline latency represents the time for parallel execution of algorithms at the point level (simulating a non-collaborative system), while full MAS latency includes correlation analysis, clustering, leader election, and hierarchical report aggregation. Figure 5.5 shows the latency comparison across profiles.

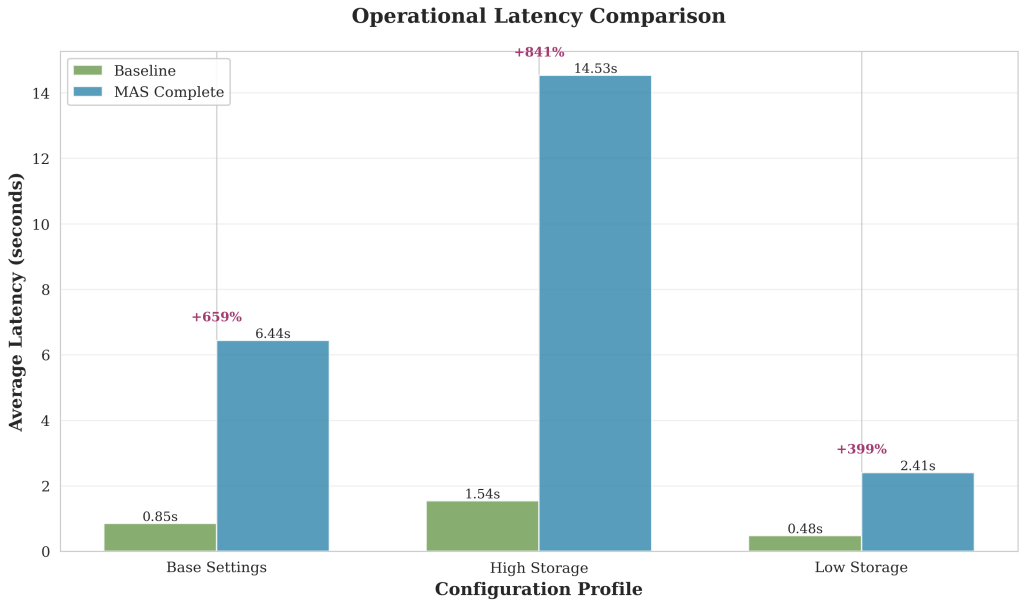

Figure 5.5: Comparison of operational latency between baseline (parallel processing only) and full MAS (including collaborative features) across profiles.

The Low Storage profile demonstrated the best computational performance with an average latency of 2.41 seconds for complete MAS processing compared to a baseline of 0.48 seconds, representing 399% additional time. The Base Settings profile required 6.44 seconds of average MAS latency versus 0.85 seconds baseline (+659%), while the High Storage profile exhibited 14.53 seconds of average MAS latency compared to 1.54 seconds baseline (+841%).

The substantial increase in latency in High Storage is attributable to several factors: larger historical windows (100 measurements for anomaly detection vs. 20 in Low Storage), more sophisticated aggregation strategies (weighted probability vs. simple majority),

and confidence-weighted calculations across multiple hierarchical levels. Additionally, the High Storage profile processes larger correlation matrices due to its extended historical retention.

Table 5.5 presents detailed latency statistics including median and 95th percentile values.

Table 5.5: Detailed latency statistics across configuration profiles

Profile	**Baseline (seconds)**			**Full MAS (seconds)**		
	Mean	Median	P95	Mean	Median	P95
Base Settings	0.85	0.82	1.45	6.44	5.80	12.81
High Storage	1.54	1.52	2.89	14.53	13.59	26.47
Low Storage	0.48	0.46	0.91	2.41	2.49	4.52

Whether these latencies are industrially acceptable depends on the production cycle time. For typical automotive body shops, where cycle times can range from under a minute to several minutes depending on production volume and automation level, the diagnostic latency may represent an acceptable fraction of total cycle time, particularly if processing occurs in parallel with mechanical assembly operations. For higher throughput lines, the Low Storage profile with lower latency (2.4 seconds) would be more appropriate, while lower throughput lines can accommodate the additional processing time of the High Storage profile (15 seconds).

The additional time beyond the baseline represents the cost of collaborative features: correlation analysis identifies relationships between measurement points, clustering organizes related agents, consensus mechanisms aggregate multiple perspectives, and hierarchical reports provide multi-level diagnostic insights. These features provide value beyond simple parallel processing, although whether this value justifies the computational cost depends on the operational context, as discussed in Section 5.3.3.

5.2.5 Cluster Formation and Effectiveness

Multi-agent architecture identifies correlations between measurement points and organizes them into clusters for collaborative diagnosis. Clustering statistics provide insight into the degree of collaboration actually achieved in practice. Figure 5.6 shows the distribution of cluster sizes across the three profiles.

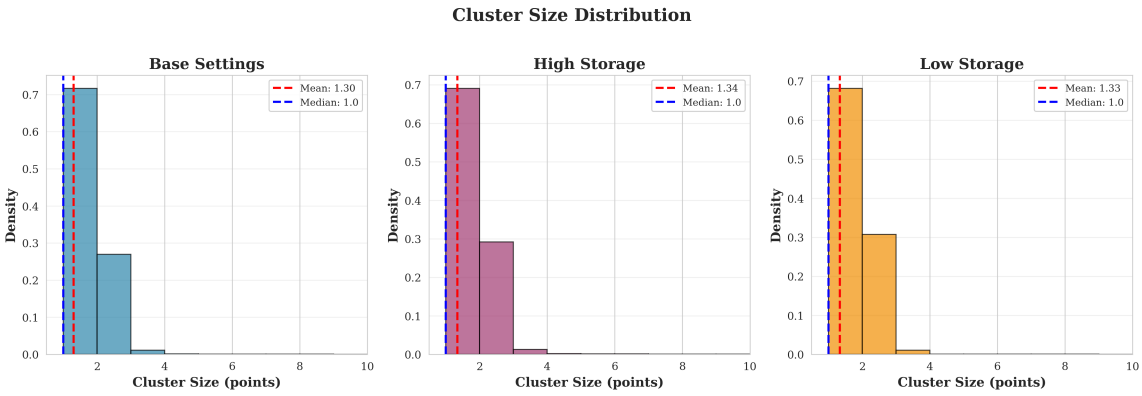

Figure 5.6: Distribution of cluster sizes across configuration profiles.

The cluster size distributions are remarkably similar across all three profiles, with average sizes of 1.30 (Base Settings), 1.34 (High Storage), and 1.33 (Low Storage). These small average sizes indicate that approximately 70% of clusters consist of single measurement points, with the remaining 30% comprising clusters of 2-3 points. Clusters larger than 4 points are extremely rare across all configurations.

This limited clustering suggests that strong correlations (above the Pearson threshold of 0.7 used in all profiles) between measurement points are relatively uncommon in the dataset. Possible explanations include: (1) measurement points genuinely exhibit independent behavior due to localized manufacturing processes, (2) high process noise masks underlying correlations, or (3) the correlation threshold of 0.7 is too conservative for this application. The implications of limited clustering for the value of multi-agent architecture are discussed in Section 5.3.3.

Despite the small cluster sizes, the system achieved high confidence ratings in most cases. Table 5.6 shows the distribution of confidence levels in cluster reports.

Table 5.6: Distribution of confidence levels in cluster reports

Profile	Total Clusters	High Conf.	Medium Conf.	Low Conf.
Base Settings	210,864	186,356 (88.4%)	18,471 (8.8%)	6,037 (2.9%)
High Storage	116,701	102,503 (87.8%)	10,670 (9.1%)	3,528 (3.0%)
Low Storage	364,354	321,016 (88.1%)	32,792 (9.0%)	10,546 (2.9%)

Approximately 88% of cluster reports achieved a "High Confidence" rating across all profiles, indicating strong internal consistency within clusters even when cluster sizes are small. This suggests that when measurement points cluster together, they provide mutually reinforcing diagnostic information.

5.2.6 Comparison of Configuration Profiles and Trade-offs

To provide a holistic comparison of the three configuration profiles, Figure 5.7 presents a radar chart comparing normalized performance across five key dimensions: Sensitivity (detection capability), Accuracy (alarm reliability), Early Warning Accuracy (predictive capability), Cluster Efficiency (high confidence rate), and Speed (inverse of latency).

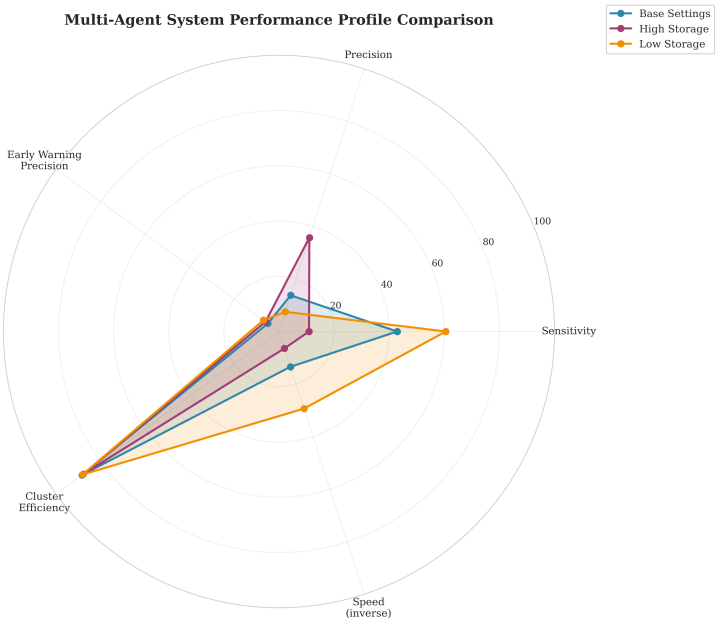

Figure 5.7: Multi-dimensional performance comparison across configuration profiles.

The radar chart visualizes the fundamental trade-offs between profiles. High Storage exhibits the strongest performance in Accuracy (outer region) but the weakest in Sensitivity (inner region), reflecting its conservative detection philosophy. Low Storage shows the opposite pattern, with strong Sensitivity and Speed but weak Accuracy. Base Settings occupies intermediate positions across most dimensions, although it does not achieve the best performance on any single metric.

Table 5.7 provides a comprehensive numerical comparison across all key metrics evaluated in this study.

Table 5.7: Comprehensive comparison of configuration profiles across all evaluated metrics

Metric	**Base**	**High**	**Low**
Validation Metrics			
Sensitivity (%)	42.9	10.9	**60.6**
Specificity (%)	86.3	**99.0**	62.0
Precision (%)	13.8	**35.7**	7.5
Early Warning			
Confirmed Warnings	886	44	**5,919**
EW Precision (%)	4.9	6.0	**7.0**
Avg. Lead Time (vehicles)	4.4	3.7	**4.8**
Complementarity			
Early Warning Rate (%)	13.0	1.0	**36.2**
Reinforcement Rate (%)	2.1	0.5	**2.9**
Both OK Rate (%)	82.1	**94.0**	59.0
Performance			
Baseline Latency (s)	0.85	1.54	**0.48**
MAS Latency (s)	6.44	14.53	**2.41**
Additional Time (%)	659	841	**399**
Clustering			
Avg. Cluster Size	1.30	**1.34**	1.33
High Confidence (%)	88.4	87.8	88.1

The comprehensive comparison confirms that each profile excels in different dimensions. High Storage achieves the best Specificity (99.0%) and Accuracy (35.7%), making it suitable for contexts where false alarms are particularly costly. Low Storage achieves

the best Sensitivity (60.6%), more early warnings (5,919), and lower latency (2.41s), suitable for resource-constrained environments prioritizing detection coverage. Base Settings provides consistent intermediate performance without extreme strengths or weaknesses.

These results demonstrate that the multi-agent architecture successfully supports diverse operational philosophies through configuration alone, without requiring architectural modifications. The implications of these findings for industrial deployment and research contributions are examined in the following discussion section.

5.3 Results Discussion

This section interprets the experimental results presented in Section 5.2, contextualizing them within the objectives established in Chapter 1 and the limitations identified in the literature review (Chapter 2). The discussion addresses four key aspects: the fundamental trade-offs between configuration profiles, the limitations of individual diagnostic algorithms, the value provided by the collaborative multi-agent architecture, and the positioning of this work relative to the state of the art.

5.3.1 Trade-offs Between Configuration Profiles

The three configuration profiles evaluated represent fundamentally different operational philosophies for fault diagnosis, each optimized for distinct industrial scenarios. The results confirm clear trade-offs between accuracy and sensitivity, detection coverage, and alarm reliability.

The **High Storage** profile exemplifies a precision-oriented strategy with 99.0% Specificity and 35.7% Precision (approximately 1 in 3 alerts corresponds to an actual violation). This high reliability comes at the cost of coverage: a Sensitivity of 10.9% means that 89% of actual violations are not preceded by warnings. This profile is appropriate for environments where false alarms incur significant operational costs: 24/7 continuous production lines where unscheduled interventions are extremely costly, facilities with strict alarm response procedures, and highly automated plants where alarm fatigue must be minimized.

The **Low Storage** profile implements the opposite philosophy, prioritizing detection coverage. The sensitivity of 60.6% detects approximately 60% of rejections, substantially higher than the other profiles. The cost is evident in the accuracy of 7.5% (approximately 12 false alarms per genuine problem), which can lead to alarm fatigue. However, it offers compensating advantages: lower latency overhead (+399%, 2.41s), minimal storage requirements (5MB measurements, 10MB reports), and computational simplicity. It is suitable for safety-critical applications where missing a fault is unacceptable, edge computing deployments with limited resources, and scenarios where defect costs dramatically exceed inspection costs.

The **Base Settings** profile occupies intermediate positions with 42.9% Sensitivity, 86.3% Specificity, and 13.8% Precision, providing reasonable coverage while maintaining acceptable reliability. It is appropriate as a default setting for initial deployment when operational priorities are uncertain.

The existence of three dramatically different profiles achieved only by configuration, without architectural modifications, demonstrates key flexibility of the system: organizations can adapt behavior by adjusting parameters rather than implementing separate systems. However, this flexibility introduces complexity into configuration management, suggesting a need for automated parameter tuning approaches for future work.

5.3.2 Limitations of Individual Diagnostic Algorithms

A critical finding across all profiles is the relatively low accuracy of anomaly detection, ranging from 7.5% (Low Storage) to 35.7% (High Storage). Even in the best case, approximately 64% of alerts are false positives. This limitation reflects fundamental constraints of the simple algorithms employed.

The **Classification** algorithm provides deterministic results based on engineering specifications, but operates purely reactively, only confirming violations after the fact. Using Classification as a validation anchor introduces circularity: Anomaly Detection is validated against Classification, but this only confirms consistency with limit violations,

not whether anomalies represent genuine precursors versus random fluctuations. Overcoming this limitation would require a prospective study where operators act on early warnings and the effectiveness of interventions is explicitly tracked.

Anomaly detection using Modified Z-Score with MAD offers robustness against outliers and computational efficiency suitable for industrial deployment. However, it assumes that deviations from history indicate problems, an assumption that is violated when normal process variation is high. The proximity gating mechanism was introduced to reduce false positives, but its effectiveness is limited—even with a proximity ratio of 0.4, accuracy remains at 7.5%. The fundamental problem is the difficulty in distinguishing significant changes from random fluctuations within normal operating ranges.

Linear regression assumes linear trends that will continue uninterrupted, assumptions that are often violated in real processes where adjustments, maintenance, or material changes cause abrupt changes. The algorithm also faces challenges in distinguishing stable processes near limits versus degrading processes. Addressing this requires incorporating process knowledge, integrating maintenance schedules, supplier batch changes, or environmental variations.

A fundamental shared limitation is the inexistence of supervised learning on labeled data. The system operates entirely on unsupervised or rule-based principles. This choice reflects industrial reality: defects are rare events (4.85-4.99% rejection rate), and labeled data is rarely available in sufficient quantities. However, in production deployment, labeled data would naturally accumulate as operators investigate alerts. Semi-supervised or active learning approaches could gradually improve accuracy.

The observed cluster sizes (average 1.30-1.34) indicate that strong correlations between points are uncommon at the Pearson threshold of 0.7 employed. Approximately 70% of agents operate in isolation. Contributing factors include: the threshold may be too conservative for industrial data with high noise, Pearson correlation captures only linear relationships, or the data genuinely reflects independent localized processes. The high confidence rate (88%) despite small clusters suggests that when points cluster, they provide mutually reinforcing information. Limited clustering has a positive implication:

computational complexity scales linearly with cluster size, so small clusters enable efficient processing.

5.3.3 Quantitative Analysis of Collaborative Overhead

Given the limitations discussed above, particularly the low accuracy of individual algorithms and limited clustering, a critical question arises: do the quantitative benefits of collaboration justify the computational overhead introduced?

Trade-off: Latency vs Cycle Time

The comparison between baseline latency (parallel processing at the point level) and full MAS latency (including collaboration) provides a direct quantitative assessment of this trade-off. Table 5.8 summarizes the observed computational overhead:

Table 5.8: Computational overhead analysis by configuration profile

Profile	**Overhead (%)**	**Total Latency (s)**	**% Cycle Time***
Low Storage	399	2.41	2.7-4.0%
Base Settings	512	7.83	8.7-13.1%
High Storage	841	14.53	16.1-24.2%

* Considering typical cycle time of 60-90 seconds

The additional overhead of 399-841% appears substantial in relative terms, but the absolute values (2.41-14.53 seconds) must be contextualized against typical production cycle times. For automotive body shops, where cycle times can range from less than a minute to several minutes depending on production volume and automation level, even the highest latency observed may represent an acceptable fraction of the total cycle time. This technical feasibility is particularly evident when diagnostic processing occurs in parallel with mechanical assembly operations.

Informational Value of Collaboration

The complementarity analysis quantifies the informational value added by collaboration. Early warning rates of 13.0-36.2% (varying by profile) represent cases where Anomaly Detection identified potential problems before specification violations occurred. Although the low accuracy of early warning (4.9-7.0%) indicates a high false positive rate, confirmed warnings (886-5,919 depending on profile) with an average lead time of 4-5 vehicles demonstrate genuine quantifiable predictive capability. This informational gain would not exist in a system based solely on reactive classification.

Quantitative Parameterization of Trade-offs

The three configuration profiles (Base Settings, Low Storage, High Storage) demonstrate quantitative parameterization of the sensitivity-specificity trade-off without architectural modification:

- **Low Storage:** 60.6% sensitivity, 7.5% precision (high sensitivity, low precision - maximizes detection coverage)
- **Base Settings:** 42.9% sensitivity, 13.8% precision (intermediate equilibrium)
- **High Storage:** 10.9% sensitivity, 35.7% precision (lower sensitivity, higher precision - minimizes false alarms)

This variation of 49.7 percentage points in sensitivity and 28.2 points in precision through parametric adjustment validates the operational flexibility of the architecture. The ability to reconfigure behavior without changing code or retraining models reduces the cost of adapting to different operational contexts.

5.3.4 State of the Art Positioning

The proposed multi-agent system addresses several gaps identified in the literature review (Chapter 2) while acknowledging limitations related to cutting-edge research.

Distinct application domain. Existing multi-agent systems for industrial diagnostics focus predominantly on predictive equipment maintenance (machine health, vibration analysis, temperature monitoring), while this work monitors product quality (gap, flush) during assembly. This represents a domain with distinct characteristics: observable defects at specific inspection points versus continuous monitoring, ground truth defined by engineering specifications versus eventual equipment failure, and interventions aimed at process adjustments versus maintenance scheduling.

Flexible architecture versus hardcoded. Previous work combining multiple diagnostic techniques employs centralized architectures where integration is hardcoded. Ensemble learning approaches run multiple models and vote on results, but adding a new model requires modifying the structure and retraining. The proposed architecture enables new algorithms by implementing a standard interface and registering them in the catalog, without modifying existing agents or retraining models.

Multi-station hierarchical structure. Existing systems do not systematically exploit explicit correlation analysis between stations. Related work treats each station independently, or uses black-box deep learning models that implicitly capture correlations but do not make them explicit. The proposed architecture explicitly models intra- and inter-station correlations, organizing agents into clusters based on measured relationships, providing hierarchical reports that aggregate diagnostics at point, cluster, and station levels.

Interpretability vs Sophistication. This work adopts deliberately simple algorithms (MAD z-score, linear regression), suitable for industrial deployment where interpretability and operator confidence are critical. Deep learning techniques (LSTM, transformers) achieve higher accuracy in benchmarks, but require large labeled datasets that are rarely available, substantial computational resources, and provide black-box predictions. Correlation analysis uses Pearson, capturing only linear relationships. Sophisticated measures (mutual information, Granger causality) could identify nonlinear relationships, but are computationally expensive and may identify spurious dependencies.

Pragmatic validation. The methodology uses Classification as an anchor, lacking

ground truth labels for "true anomalies" versus "false alarms." State-of-the-art evaluation uses datasets labeled by experts. The compromise adopted (retrospective validation using specification violations) is pragmatic given resource constraints but limits the strength of conclusions about predictive capability.

Practical Application vs Simulation. This work demonstrates the practical application of MAS principles to real industrial problems. Much multi-agent research uses simulated environments. The challenges of noisy industrial data, limited historical availability, integration with existing infrastructure, and operator acceptance constraints present practical obstacles not captured in simulation. The hierarchical organization (point, cluster, station, inter-station) with explicit clustering represents an architectural pattern applicable beyond quality control, potentially supporting distributed sensor networks, environmental monitoring, or smart grid applications.

The fair comparison is not "MAS versus nothing" but "MAS versus simpler alternatives." Independent parallel processing would eliminate clustering and hierarchical reporting, reducing overhead but eliminating the ability to leverage correlations. A centralized pipeline would simplify architecture but create a single point of failure. Ensemble learning frameworks do not naturally accommodate multi-station hierarchical structure or real-time streaming features.

The fundamental contribution is to demonstrate the viability of multi-agent architectures for industrial diagnostics, with specific strengths (extensibility, transparency, distributed resilience) that justify complexity in contexts prioritizing long-term evolution and interpretability over pure diagnostic accuracy.

Chapter 6

Conclusions

This chapter presents the conclusions of the work carried out, summarizing the main contributions, discussing the implications of the results obtained, acknowledging the limitations identified, and proposing directions for future research. Section 6.1 summarizes the work carried out and the main contributions. Section 6.2 discusses the practical implications of the results for the automotive industry. Section 6.3 acknowledges the limitations of the current work. Finally, Section 6.4 proposes directions for future work that could extend and improve the capabilities of the developed system.

6.1 Summary of Contributions

This dissertation developed a Multi-Agent System that integrates multiple machine learning techniques for collaborative diagnosis of geometric defects in automotive assembly lines. The central motivation for the work, established in Chapter 1, was the realization that traditional diagnostic approaches, based on isolated methods or single-technique evaluations, are inadequate for dealing with the complexity of modern production processes.

The work developed generated multiple contributions from both a scientific and practical point of view, in line with the specific objectives established in Section 1.2:

1. **Hierarchical multi-agent architecture:** Design of a three-layer structure (Point

Agents, Station Agents, and Inter-Station Agent) that enables distributed analysis and collaborative decision-making, as detailed in Chapter 4. This organization provides scalability and specialization.

2. **Extensible diagnostic framework:** Development of a modular architecture that supports the integration of multiple machine learning techniques through abstract interfaces for stateless, supervised, and unsupervised algorithms (Section 4.2.4). This structure facilitates the incorporation of scientific advances without fundamental restructuring.

3. **Coordination mechanisms:** Implementation of correlation and clustering algorithms to identify points with related behaviors, and communication protocols including distributed leader election based on timestamps (Sections 4.2.3 and 4.2.2). The hierarchical communication structure significantly reduces message volume and distributes the processing load.

4. **Diagnostic fusion strategies:** Creation of a hierarchical reporting architecture on three levels (Point, Cluster, and Station) with aggregation and summarization mechanisms that synthesize individual diagnostics into consolidated assessments (Section 4.2.5). The inclusion of confidence metrics and preservation of complete traceability support informed decision-making and effective corrective action.

5. **Industrial validation:** Demonstration of the system's ability to detect gap and flush defects using real data from automotive body shop operations, comprising two sequential stations with 62 distributed measurement points (Chapter 3).

Overall, this work contributes to advancing knowledge at the intersection of Multi-Agent Systems, Machine Learning, and Industrial Diagnostics. The research addresses gaps identified in the literature review (Section 2.4), demonstrating that distributed collaboration between heterogeneous techniques can leverage the modularity, scalability, and robustness characteristics of multi-agent architectures to overcome limitations of individual machine learning methods.

6.2 Practical Implications for Industry

The results obtained in this work have significant implications for industrial practice, particularly in the context of automotive manufacturing and more broadly for Industry 4.0.

From an operational standpoint, the system transforms large volumes of data into actionable knowledge through automated and distributed analysis. The hierarchical reporting structure supports different levels of decision-making, from localized corrective actions to long-term strategic assessments.

The system's ability to identify correlations between measurement points at different stations through the Inter-Station Agent represents an important advance in the holistic understanding of the production process. This cross-sectional perspective allows tracking how defects observed in advanced stages may be related to problems originating in earlier stages, facilitating the identification of root causes and preventive interventions. The integrated analysis of multiple sequential stations aligns with Zero Defect Manufacturing principles, transforming reactive systems that identify problems after they occur into proactive systems capable of anticipating and preventing defects.

The modular and extensible architecture provides essential adaptability for dynamic industrial environments. New measurement points, diagnostic algorithms, or configurations can be incorporated without modifying existing code. This flexibility is particularly relevant in environments where multiple products are manufactured on the same infrastructure.

The diversity of analytical techniques provides operational robustness. The multi-agent approach offers intelligent redundancy where the failure of one algorithm does not compromise the overall diagnosis. Fusion with confidence metrics allows ambiguous situations that require additional validation to be flagged.

From a strategic standpoint, implementation has demonstrated technical feasibility in a real industrial environment, processing continuous production data and integrating with existing measurement infrastructure.

Potential benefits include cost and waste reduction through early identification, increased productivity, and improved quality. Although this work did not conduct a quantitative return on investment analysis, the proposed architecture lays the foundation for industrial deployment.

6.3 Limitations of the Study

Despite its contributions, this study has limitations that should be acknowledged and considered when interpreting the results.

Scope of experimental validation. Although the system was validated with real automotive production data, the application was limited to a single type of process (body shop) in a single organization. Specific characteristics of this context, including typical defect patterns, process variability, and available measurement infrastructure, may influence system performance. Generalization to other industrial contexts, manufacturing sectors, or defect types requires additional validation.

Coverage of machine learning techniques. The selection of algorithms integrated into the system, although representing different analytical categories (stateless, supervised, unsupervised), is not exhaustive. More sophisticated techniques such as deep learning, particularly architectures designed for sequential or temporal data, have not been systematically explored. The investigation of whether and how such techniques could be integrated into the multi-agent architecture, and whether they would provide superior diagnostic benefits, remains an open question.

Diagnostic fusion strategies. The strategies implemented, although effective for the algorithms tested, are based on relatively straightforward approaches. More sophisticated fusion techniques, such as Dempster-Shafer evidence theory, Bayesian fusion, or meta-learning, where a model learns to optimally combine predictions from base models, have not been explored and constitute an opportunity for future work.

Characterization of operational performance. The evaluation focused on diagnostic accuracy metrics. Practical implementation aspects such as scalability for very

large numbers of measurement points, memory consumption in large-scale scenarios, and latency optimization for different hardware architectures were not systematically characterized. Although the distributed architecture suggests favorable scalability, quantitative validation of this property would be valuable.

Interpretability and explainability. The interpretability of the diagnoses produced, particularly when multiple algorithms contribute to a consolidated conclusion, represents a recognized but not fully addressed challenge. While the system preserves traceability, allowing the identification of which agents and algorithms contributed to each diagnosis, explaining why a particular conclusion was reached in terms understandable to human operators requires further development. Integration of Explainable AI (XAI) techniques could significantly improve trust and adoption of the system.

Temporal validation and real deployment. Validation was limited to historical data analyzed retrospectively. Operational deployment in a real production environment, processing data in real time and supporting current operational decisions, would introduce additional challenges not fully explored, including management of communication failures, temporal synchronization between agents, and recovery of states after interruptions.

Enterprise integration. Aspects of integration with broader enterprise systems, including manufacturing execution systems (MES), enterprise resource planning (ERP), and predictive maintenance frameworks, were not addressed. While the proposed architecture is conceptually compatible with such integrations, practical implementation would require development of appropriate interfaces and validation of interoperability.

6.4 Directions for Future Work

Based on the contributions made and the limitations identified, several promising directions for future work emerge.

Validation in Diverse Industrial Contexts

Applying and validating the proposed architecture in additional manufacturing sectors, such as aerospace, electronics, pharmaceuticals, or food, would allow us to identify general principles that are widely applicable versus adaptations needed for specific contexts. This comparative research would refine the understanding of when and how the architecture provides the most value, characterizing types of processes, data characteristics, and operational requirements that benefit most from the collaborative multi-agent approach.

Exploration of Advanced Algorithms

Integrating more sophisticated machine learning techniques is a promising direction. Deep learning architectures designed for sequential data, such as LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Units), could capture complex temporal dependencies. Transfer learning would allow leveraging pre-trained models, potentially reducing labeled data requirements. Meta-learning could enable rapid adaptation to new types of defects with limited data. Reinforcement learning could enable agents to learn optimized collaboration strategies through experience, dynamically adapting behavior to observed patterns.

Advanced Fusion Strategies

The development of more sophisticated diagnostic fusion strategies represents a rich area for research. Dempster-Shafer evidence theory could provide a robust mathematical framework for combining diagnoses with different levels of specificity and confidence. Bayesian fusion would allow formal incorporation of prior knowledge about defect probabilities. Meta-learning for fusion, where a model learns to optimally combine predictions based on data characteristics and operational context, could adapt the strategy dynamically. Investigation of adaptive weighting based on historical performance of algorithms under different conditions would be a valuable contribution.

Collaborative Learning among Agents

An ambitious extension would be to allow agents not only to collaborate on diagnostic fusion but also to learn from each other. Federated learning could enable distributed model training while preserving data privacy. Agents could share knowledge about identified patterns, model updates, or successful strategies, accelerating collective learning. Imitation learning mechanisms could allow less experienced agents to learn from better-performing agents.

Integration with Predictive Maintenance

Integrating the diagnostic system with predictive maintenance frameworks would be a natural extension. In addition to identifying product defects, the system could monitor the health of production equipment, predicting failures based on degradation patterns. This integration would enable holistic coordination between product quality and equipment health, optimizing maintenance scheduling to minimize impact on production while preventing failures that compromise quality.

Implementation Optimization

Systematic investigation of practical aspects including latency optimization, efficient management of computational resources, fault tolerance strategies, and integration with existing infrastructures would facilitate large-scale deployment. Development of containerized frameworks, orchestration through platforms such as Kubernetes, and integration with industry standards such as OPC UA would contribute to widespread industrial adoption.

Cost-Benefit Studies

Quantitative studies comparing the proposed system with conventional approaches would provide valuable evidence for organizational decision-making. Metrics such as reduction in rework, decrease in quality costs, increase in throughput, and return on investment would

demonstrate economic value. Longitudinal studies in real environments documenting the evolution of indicators, the effectiveness of diagnostic-based interventions, and user feedback would provide insights into sustainable practical benefits.

6.5 Final Considerations

This work has made a significant contribution to the advancement of industrial diagnostic systems through the development of a Multi-Agent System architecture that integrates multiple machine learning techniques into a collaborative and distributed framework. Validation with real automotive production data has demonstrated the viability of the architecture for complex industrial contexts where diagnostic accuracy is critical to product quality and operational efficiency.

The limitations identified and the directions proposed for future work adequately contextualize the contributions made and outline promising avenues for subsequent research that can extend and enhance the capabilities demonstrated.

The convergence of Multi-Agent Systems with advanced Machine Learning techniques represents a fertile area of research with significant potential to transform diagnostic and quality control practices in Industry 4.0. This work lays solid foundations on which future developments can build, contributing to the realization of visions of intelligent, adaptive, and autonomous manufacturing that characterize the future of industrial production.

Bibliography

[1] R. Iqbal, T. Maniak, F. Doctor, and C. Karyotis, "Fault detection and isolation in industrial processes using deep learning approaches," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 3077–3084, May 2019, ISSN: 1941-0050. DOI: 10.1109/TII.2019.2902274

[2] Y. Seid Ahmed, A. A. Abubakar, A. F. M. Arif, and F. A. Al-Badour, "Advances in fault detection techniques for automated manufacturing systems in industry 4.0," *Frontiers in Mechanical Engineering*, vol. Volume 11 - 2025, 2025, ISSN: 2297-3079. DOI: 10.3389/fmech.2025.1564846 [Online]. Available: https://www.frontiersin.org/journals/mechanical-engineering/articles/10.3389/fmech.2025.1564846

[3] Y.-J. Park, S.-K. S. Fan, and C.-Y. Hsu, "A review on fault detection and process diagnostics in industrial processes," *Processes*, vol. 8, no. 9, 2020, ISSN: 2227-9717. DOI: 10.3390/pr8091123 [Online]. Available: https://www.mdpi.com/2227-9717/8/9/1123

[4] J. Lario, J. Mateos, F. Psarommatis, and Á. Ortiz, "Towards zero defect and zero waste manufacturing by implementing non-destructive inspection technologies," *Journal of Manufacturing and Materials Processing*, vol. 9, no. 2, 2025, ISSN: 2504-4494. DOI: 10.3390/jmmp9020029 [Online]. Available: https://www.mdpi.com/2504-4494/9/2/29

[5] W. Yan, J. Wang, S. Lu, M. Zhou, and X. Peng, "A review of real-time fault diagnosis methods for industrial smart manufacturing," *Processes*, vol. 11, no. 2,

2023, ISSN: 2227-9717. DOI: 10.3390/pr11020369 [Online]. Available: https://www.mdpi.com/2227-9717/11/2/369

[6] F. Arellano-Espitia, M. Delgado-Prieto, V. Martinez-Viol, J. J. Saucedo-Dorantes, and R. A. Osornio-Rios, "Deep-learning-based methodology for fault diagnosis in electromechanical systems," *Sensors*, vol. 20, no. 14, 2020, ISSN: 1424-8220. DOI: 10.3390/s20143949 [Online]. Available: https://www.mdpi.com/1424-8220/20/14/3949

[7] A. Saeed, M. A. Khan, U. Akram, W. J. Obidallah, S. Jawed, and A. Ahmad, "Deep learning based approaches for intelligent industrial machinery health management and fault diagnosis in resource-constrained environments," *Scientific Reports*, vol. 15, no. 1, p. 1114, Jan. 2025, ISSN: 2045-2322. DOI: 10.1038/s41598-024-79151-2 [Online]. Available: https://doi.org/10.1038/s41598-024-79151-2

[8] M. Compare, P. Baraldi, and E. Zio, "Challenges to iot-enabled predictive maintenance for industry 4.0," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4585–4597, May 2020, ISSN: 2327-4662. DOI: 10.1109/JIOT.2019.2957029

[9] M. Fernandes, J. M. Corchado, and G. Marreiros, "Machine learning techniques applied to mechanical fault diagnosis and fault prognosis in the context of real industrial manufacturing use-cases: A systematic literature review," *Applied Intelligence*, vol. 52, no. 12, pp. 14 246–14 280, Sep. 2022, ISSN: 1573-7497. DOI: 10.1007/s10489-022-03344-3 [Online]. Available: https://doi.org/10.1007/s10489-022-03344-3

[10] E. H. Abualsauod, "Machine learning based fault detection approach to enhance quality control in smart manufacturing," *Production Planning & Control*, vol. 0, no. 0, pp. 1–9, 2023. DOI: 10.1080/09537287.2023.2175736 eprint: https://doi.org/10.1080/09537287.2023.2175736. [Online]. Available: https://doi.org/10.1080/09537287.2023.2175736

[11] N. L. Vithi and C. Chibaya, "Advancements in predictive maintenance: A bibliometric review of diagnostic models using machine learning techniques," *Analytics*, vol. 3, no. 4, pp. 493–507, 2024, ISSN: 2813-2203. DOI: 10.3390/analytics3040028 [Online]. Available: https://www.mdpi.com/2813-2203/3/4/28

[12] A. Singh and S. Bobde, "A review on predicting failure in industrial machines: Its methods, challenges and future direction," in *2025 1st International Conference on AIML-Applications for Engineering & Technology (ICAET)*, Jan. 2025, pp. 1–6. DOI: 10.1109/ICAET63349.2025.10932305

[13] T. Pulikottil, L. A. Estrada-Jimenez, H. Ur Rehman, F. Mo, S. Nikghadam-Hojjati, and J. Barata, "Agent-based manufacturing — review and expert evaluation," *The International Journal of Advanced Manufacturing Technology*, vol. 127, no. 5, pp. 2151–2180, Jul. 2023, ISSN: 1433-3015. DOI: 10.1007/s00170-023-11517-8 [Online]. Available: https://doi.org/10.1007/s00170-023-11517-8

[14] J. Queiroz, P. Leitão, J. Barbosa, E. Oliveira, and G. Garcia, "Agent-based distributed data analysis in industrial cyber-physical systems," *IEEE Journal of Emerging and Selected Topics in Industrial Electronics*, vol. 3, no. 1, pp. 5–12, Jan. 2022, ISSN: 2687-9743. DOI: 10.1109/JESTIE.2021.3100775

[15] T. Pulikottil, L. A. Estrada-Jimenez, H. U. Rehman, J. Barata, S. Nikghadam-Hojjati, and L. Zarzycki, "Multi-agent based manufacturing: Current trends and challenges," in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2021, pp. 1–7. DOI: 10.1109/ETFA45728.2021.9613555

[16] R. S. Ahluwalia, "A review of multi agent-based production scheduling in manufacturing system," *Recent Patents on Engineering*, vol. 15, no. 5, pp. 5–22, 2021, ISSN: 1872-2121/2212-4047. DOI: 10.2174/1872121114999200423112955 [Online]. Available: https://www.eurekaselect.com/article/106052

[17] U. I. Inyang, I. Petrunin, and I. Jennions, "Diagnosis of multiple faults in rotating machinery using ensemble learning," *Sensors*, vol. 23, no. 2, 2023, ISSN: 1424-8220. DOI: 10.3390/s23021005 [Online]. Available: https://www.mdpi.com/1424-8220/23/2/1005

[18] J. Cohen, B. Jiang, and J. Ni, "Machine learning for diagnosis of event synchronization faults in discrete manufacturing systems," *Journal of Manufacturing Science and Engineering*, vol. 144, no. 7, p. 071 006, Dec. 2021, ISSN: 1087-1357. DOI: 10.1115/1.4052762 eprint: https://asmedigitalcollection.asme.org/manufacturingscience/article-pdf/144/7/071006/6808233/manu_144_7_071006.pdf. [Online]. Available: https://doi.org/10.1115/1.4052762

[19] S. Baroud, N. Yahaya, and A. M. Elzamly, "Cutting-edge ai approaches with mas for pdm in industry 4.0: Challenges and future directions," *Journal of Applied Data Sciences*, vol. 5, no. 2, pp. 455–473, 2024. DOI: 10.47738/jads.v5i2.196 [Online]. Available: https://bright-journal.org/Journal/index.php/JADS/article/view/196

[20] M. El Koujok, A. Ragab, H. Ghezzaz, and M. Amazouz, "A multiagent-based methodology for known and novel faults diagnosis in industrial processes," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3358–3366, May 2021, ISSN: 1941-0050. DOI: 10.1109/TII.2020.3011069

[21] H. Garcia-Molina, "Elections in a distributed computing system," *IEEE Transactions on Computers*, vol. 100, no. 1, pp. 48–59, 1982.

[22] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.

[23] C. Bron and J. Kerbosch, "Algorithm 457: Finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[24] SPADE Developers, *SPADE: Smart Python Agent Development Environment*, Accessed: 2024-10-07, 2023. [Online]. Available: `https://spade-mas.readthedocs.io/`

[25] Foundation for Intelligent Physical Agents, "FIPA Abstract Architecture Specification," FIPA, Tech. Rep. SC00001L, 2002. [Online]. Available: `http://www.fipa.org/specs/fipa00001/`

[26] P. Saint-Andre, *XMPP: The Extensible Messaging and Presence Protocol*, 2011. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc6120`

[27] W. McKinney et al., "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, Austin, TX, vol. 445, 2010, pp. 51–56.

[28] C. R. Harris, K. J. Millman, S. J. van der Walt, et al., "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, et al., "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[30] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, Pasadena, CA USA, vol. 836, 2008, pp. 11–15.

[31] MongoDB Inc., *PyMongo: Python Driver for MongoDB*, Accessed: 2024-10-07, 2023. [Online]. Available: `https://pymongo.readthedocs.io/`

[32] MongoDB Inc., *MongoDB: The Developer Data Platform*, Accessed: 2024-10-07, 2023. [Online]. Available: `https://www.mongodb.com/`

[33] Docker Inc., *Docker: Accelerated Container Application Development*, Accessed: 2024-10-07, 2023. [Online]. Available: `https://www.docker.com/`

[34] B. Iglewicz and D. C. Hoaglin, "How to detect and handle outliers," *The ASQC basic references in quality control: statistical techniques*, vol. 16, 1993.

[35] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, 2013.

Appendix A

Appendix: Implementation Availability

The implementation of the multi-agent system developed in this work is based on real industrial data and proprietary business logic from an automotive manufacturing company. Due to confidentiality agreements and intellectual property restrictions, the complete source code cannot be publicly disclosed.

However, the architectural descriptions, algorithm explanations, and implementation details provided throughout Chapter 5 contain sufficient information to enable researchers and practitioners to replicate the approach in similar industrial contexts.

Researchers interested in collaboration or further details may contact the author or supervisors for potential academic partnerships, subject to appropriate confidentiality agreements.